FEATURE EXTRACTION FROM NATURAL LANGUAGE TO AID REQUIREMENTS REUSE IN SOFTWARE PRODUCT LINES ENGINEERING

NOOR HASRINA BAKAR

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2016

University of Malaya

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: NOOR HASRINA BINTI BAKAR

(I.C/Passport No:

Registration/Matric No: WHA 110041

Name of Degree: DOCTOR OF PHILOSOPHY

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Feature Extraction From Natural Language To Aid Requirements Reuse In Software Product Lines Engineering

Field of Study: SOFTWARE ENGINEERING

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

University of Malaya

ABSTRACT

Software Product Lines Engineering (SPLE) is a systematic approach towards realising software reuse. Among important software assets to be reused includes architectural documents, test cases, source codes and also requirements. Requirements Reuse (RR) in SPLE is the process of systematically reusing previously defined requirements for an earlier software product and applying them to a new, slightly different product within similar domain. SRS documents are not easily accessible, therefore many researchers in this area opted to use other forms of requirements including product brochures, user manuals and software reviews when SRS is not available. Unfortunately, to extract reusable features from natural language requirements for reuse is not easy. This task if done manually can be very complicated, expensive, and error-prone on the results. Many research efforts in SPLE focused on issues related to architectures, designs and code reuse, but research on requirements reuse has received slightly less attention from researchers and practitioners. Results from an exploratory survey gathered among SE practitioners indicated that the main impediments for RR practice includes the unavailability of RR tools or models for adoption, the conditions of existing requirements to be reused (incomplete, poorly structured, or not kept updated), and the lack of awareness among software practitioners pertaining to the systematic RR. Additionally, a Systematic Literature Review (SLR) conducted for feature extraction approaches for RR in SPLE reveals that there is a mixture of automated and semi-automated approaches from data mining and information retrieval, with only some approaches coming with support tools. This SLR also reveals that most of the support tools proposed in the selected studies are not made available publicly and thus making it hard for practitioners' adoption. Motivated by these findings, this research proposes a process model for feature extractions from natural language requirements for reuse (FENL). FENL consists of four main phases: Accessing Requirements, Terms Extraction, Feature Identification and Formation of Feature Model. The proposed model is demonstrated through lab experiment and online software reviews are used as the input. In phase 1, software reviews are fetched from the Internet. Then, in phase 2, these reviews undergo text pre-processing stage. In phase 3, Latent Semantic Analysis (LSA) and *tfidf* term weighting are used in order to determine document relatedness. Then, linguistic tagging is applied to extract software features followed by applying simple clustering algorithms to form groups of common features. In phase 4, the common features that are grouped together are passed to the feature modelling process and manual feature diagram are constructed as the final output. The extraction results from the proposed semi-automated extraction is compared with the one obtained by the manual extraction procedure performed by teachers and software practitioner. Comparisons are made in terms of accuracy metrics (precision, recall and F-Measure), and time efficiency. The proposed approach obtained a recall of up to 85.95% (78.03% average) and a precision of up to 80.16% (58.63% average), when evaluated against the truth data set created manually. Additionally, when comparing with the related works, FENL results to obtain a comparable F-Measure.

University of Malaya

ABSTRAK

Kejuruteraan Perisian Rangkaian Produk (Software Product Lines Engineering - SPLE) merupakan pendekatan secara sistematik ke arah penggunaan semula perisian. Antara aset perisian penting yang akan diguna semula di dalam SPLE termasuklah kod, dokumen senibina, kes-kes pengujian, dokumen rekacipta dan juga dokumen keperluan. Penggunaan semula dokumen keperluan (RR) di dalam SPLE ialah satu proses sistematik untuk menguna semula keperluan perisian yg pernah digunakan pada masa yang lalu. Ini bertujuan untuk mengeluarkan perisian produk baru yang sedikit berbeza tetapi masih di dalam domain yang serupa. Dokumen SRS bukanlah mudah untuk diakses, makanya ramai penyelidik menggunakan dokumen keperluan di dalam bentuk berlainan seperti brosur produk, manual pengguna dan juga semakan perisian semasa tiada SRS. Malangnya, untuk mengestrak ciri-ciri yang boleh diguna semula daripada keperluan di dalam bahasa tabii bukanlah sesuatu yang mudah. Proses ini boleh menjadi rumit, mahal dan sangat terdedah kepada ralat. Terdapat pelbagai usaha penyelidikkan di dalam SPLE yang memfokuskan kepada isu-isu berkaitan dengan penggunaan semula senibina, rekabentuk dan kod, tetapi penyelidikan di dalam penggunaan semula keperluan kurang mendapat perhatian daripada kalangan penyelidik dan pengamal. Berdasarkan keputusan kaji selidik yang dijalankan ke atas golongan yang mempraktiskan Kejuruteraan Perisian, sebab utama RR tidak diamalkan di Malaysia adalah kerana ketiadaan peralatan atau model untuk memilih keperluan untuk penggunaan semula, keadaan keperluan yang tersedia yang akan digunasemula (tidak lengkap, struktur yang lemah atau tidak dikemaskini), dan juga kurang kesedaran di kalangan golongan yang mempraktis tentang penggunaan semula secara sistematik. Tambahan pula, kajian literatur bersistematik (SLR) yang telah dijalankan di dalam kajian ini telah menunjukkan bahawa terdapat gabungan di dalam pendekatan pengekstrakan secara automasi dan juga semi-automasi daripada aktiviti perlombongan data dan dapatan semula maklumat. Kajian literatur bersistematik yang dijalankan ini juga mendedahkan bahawa kebanyakan alat sokongan yang dicadangkan di dalam kajian terdahulu tidak disediakan secara terbuka, dan ini mengehadkan pengamal perisian daripada menggunakannya. Bermotivasikan dapatan di atas, tesis ini mencadangkan satu model proses untuk pengekstrakan ciri-ciri perisian daripada bahasa tabii untuk diguna semula, FENL. Model FENL mengandungi empat fasa utama: Capaian kepada Keperluan, Pengestrakan Terma, Pengenalpastian Ciri-ciri serupa, dan Pembentukan Permodelan ciri-ciri. Proses separa-automatik ini didemonstrasikan melalui eksperimen di dalam makmal dengan menggunakan ulasan yang dibuat terhadap perisian sebagai input kepada FENL. Di dalam fasa pertama, ulasan perisian diambil daripada Internet. Di dalam fasa kedua pula, ulasan-ulasan ini akan melalui proses pra-pemprosesan teks. Di dalam fasa ketiga, teknik Latent Semantic Analysis (LSA) dan pemberatan terma *tfidf* digunakan untuk menentukan dokumen yang berkaitan. Kemudian, teknik pengelompokan mudah digunapakai untuk mengumpulkan ciri-ciri yang berkaitan. Di dalam fasa keempat, terma-terma berkaitan yang telah dikelompokkan kini dipindahkan ke proses permodelan ciri-ciri dan rajah ciri-ciri (feature model) dilakarkan secara manual sebagai output terakhir proses. Keputusan yang diperolehi daripada kaedah semi automatik ini dibandingkan dengan keputusan yang diperolehi melalui kaedah pengesktrakan secara manual yang dibuat oleh guru-guru dan penganalisis sistem. Perbandingan dibuat dari segi ukuran dan juga keefisienan masa. Kaedah yang dicadangkan oleh penulis apabila dinilai mencapai ketepatan (precision) 85.95% (purata 78.03%) dan recall setinggi 80.16% (purata 58.63%), apabila dibandingkan dengan set data yang disediakan secara manual. Tambahan pula, apabila dibandingkan dengan kerja-kerja berkaitan, FENL mencapai keputusan yang setara.

University of Malaya

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Associate Prof. Dr. Zarinah Kasirun from the Department of Software Engineering, University of Malaya, for her undivided attention and support given to this research work. Her valuable guidance, ideas, motivation and dedicated quality time spent for every progress meeting throughout this PhD research was excellent. Not to forget, Dr. Norsaremah Salleh, my second advisor from the Department of Computer Science, Kulliyyah of ICT, International Islamic University of Malaysia, who has given me direction, assistance and encouragement in making this submission comes into reality.

Most importantly, I would like to dedicate a special thanks to my beloved husband, Rodzi, who has been there for me during my ups and downs in this journey. To my children, Adam, Elsa, Aiman and Jasmine, this achievement is for all of us, and thanks for being super good throughout these four years. To my late dad, if you are still around, I am sure this achievements will make you proud. To my mom, my sister and brother, my in-laws, thank you for taking care of the children when I had to go abroad few times to present my research papers at conferences. I will never forget your sacrifice and love.

To Dr. Hamid A. Jalab, I appreciate your guidance in showing me how to do clustering. To Dr. Azni Haslizan and Mdm Siti Hawa, thank you for always being there for me, listening to my research stories. To Arnie, you have dedicated your valuable time proofread all my papers and polished my grammar mistakes, how will I forget your valuable assistance.

This research was made possible by IIUM sponsorship for paid study leave that have been awarded to me, and some part of this research was also funded by FRGS grant awarded by the Ministry of Higher Education of Malaysia (2013 – 2015).

University of Malaya

TABLE OF CONTENTS

University of Malaya

University of Malaya

University of Malaya

University of Malaya

University of Malaya

University of Malaya

University of Malaya

LIST OF FIGURES

University of Malaya

University of Malaya

LIST OF TABLES

University of Malaya

University of Malaya

LIST OF SYMBOLS AND ABBREVIATIONS

RR	:	Requirements Reuse
SPLE	:	Software Product Lines Engineering
LSA	:	Latent Semantic Analysis
LDA	:	Latent Dirichlet Allocation
DA	:	Domain Analysis
DE	:	Domain Engineering
VM	:	Variability Modeling
SE	:	Software Engineering
RE	:	Requirements Engineering
NLP	:	Natural Language Processing
IR	:	Information Retrieval
SLR	:	Systematic Literature Review
SRS	:	Software Requirements Specification
UML	:	Unified Modeling Language
RQ	:	Research Question
HAC	:	Hierarchical Clustering
IDC	:	Incremental Diffusive Clustering
VSM	:	Vector Space Model
FENL	:	Feature Extraction from Natural Language
Tf-idf	:	Term Frequency Inverse Document Frequency
PICOC	:	Population, Intervention, Comparison, Outcome & Context
OOP	:	Object Oriented Programming

University of Malaya

University of Malaya

CHAPTER 1: INTRODUCTION

Background

There are many important software reuse activities that can help to expedite new software development process. One of them is the reuse of requirements. Requirements reuse if done systematically i.e. in the context of Software Product Lines Engineering (SPLE), will expedite the time to market and increase developer's productivity. SPLE is a systematic approach towards realising software reuse. SPLE refers to software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production(Pohl, Bockle, & Van der Linden, 2005). Software assets to be reused in SPLE consist of source codes, architectural documents, requirements, test cases, design documents and other relevant artefacts in software development. Requirements Reuse (RR) in SPLE is the process of systematically reusing previously defined requirements for an earlier software product and applying them to a new, slightly different product within a similar domain. There are mainly two forms of requirements that can be reused: model based requirements, or natural language (textual) requirements. Although various efforts have been made in the area of software product lines engineering, the process of reusing natural language requirements has not captured much attention from researchers or practitioners.

University of Malaya

1.1 Problems Statement

There are three main problems investigated in this research. Firstly, reusing requirements in the form of natural language if done manually can be very arduous, time consuming, labor intensives, decrease productivity and also prone to error on the results. (Weston, Chitchyan, & Rashid, 2009), (Niu & Easterbrook, 2008), (Ferrari, Spagnolo, & Dell'Orletta, 2013) and (Boutkova & Houdek, 2011) Secondly, the current state of RR practice among software practitioners is not known. Available resources on the RR practice mainly revolve around publications resulting from research proposals, and some are not empirically validated(Alves et al., 2008). Thirdly, based on the conducted systematic literature review, there are missing guidelines, taxonomy or even process model for overall process of feature extraction for the reuse of requirements within the software engineering body of works. However, this research will only propose a process model (no taxonomy or formal guidelines will be produced out of this thesis)

1.2 Research Questions

Seven research questions have been formulated based on the above mentioned problem statements:

Research Question#1: What approaches are available to extract natural language requirements in the context of requirements reuse?

Research Question#2: How are the existing approaches being validated?

To gather the answers for the first and second research questions, a Systematic Literature Review on the feature extraction from natural language requirements for requirements reuse has been conducted and the findings will be presented and discussed in Chapter 2 of this thesis.

Additionally, the following research questions will also be addressed:

Research Question#3: To what extent has RR been used?

Research Question#4: What are the factors that might hinder RR in practice?

University of Malaya

To answer the third and fourth research questions, a survey has been conducted among software practitioners in Malaysia to explore the state of requirements reuse practice. RQ3 and RQ4 is not limited to the SPLE community only because this exploratory survey is to find the gap in between the systematic reuse and the non systematic reuse. Additionally, SPLE is known to be as common practice in Malaysia, thus limiting to SPLE context only will eventually reduce the number of respondents in this survey. These survey findings will be discussed in Chapter 4.

Research Question#5: What is the proposed feature extraction process and how to demonstrate the solution?

The proposed solution is demonstrated through lab experiment and each of the phases involved is explained in Chapter 5 of the thesis. Additionally, results obtained from the experiment are presented in Chapter 6.

Research Question#6: How is the evaluation being done?

Research Question#7: How is the experiment result ?

1.3 Aims and Objectives

This research aims to investigate the current state of requirements reuse practice and propose a solution to this problem. Based on the research questions presented earlier, the following research objectives will be covered in this thesis:

Objective #1: To identify available approaches in feature extraction from natural language requirements for requirements reuse.

Objective #2: To explore the current state of practice for requirements reuse among software practitioners.

University of Malaya

Objective #3: To propose a feature extraction process approach as the solution to requirements reuse problem.

Objective 3a: To provide a method that allows selecting textual requirements that can produce similar output as the one produced manually.

Objective#4: To evaluate the outcome of the proposed approach.

1.4 Research Design

This research has been divided into three phases. During the first phase, a brief statement about the research problem has been firstly discussed with the research supervisor, which is then followed by searching for related literatures on the initial research problem. As the problem gradually identified and understood, a Systematic Literature Review following the (Kitchenham & Charters, 2007) method has been carried out, with the purpose to identify a more specific area of the problems. Concurrently, a preliminary survey has been conducted among software practitioners to get an overview of the current state of the problem. This simple survey has been administered through email link and distributed to software practitioners in Malaysia. The results of this survey are not deemed to reflect the global sampling of the requirements reuse practice, but it is merely to gauge an overview of the subject. Based on the findings from Phase 1, the solution to the requirements reuse problem has been designed to have a four-phase process. This process was demonstrated through lab experiment, and the results are validated. Table 1.1 lists out the Research Phases and activities involved.

University of Malaya

Table 1-1 Research Phases and Activities Involves

Research Phases	Activities
1) Problem Identification	1. Performed Systematic Literature Review 2. Conducted Preliminary Study (Survey)
2) Research Design	Proposed a model to perform the following tasks: 1. Accessing Requirements 2. Terms Extraction 3. Feature identification 4. Formation of Feature Model
3) Evaluation	Performed: 1. Accuracy (Precision, Recall & F-Measure) Measurements 2. Statistical Validation

1.5 Research Scope and Limitations

This research focuses on the feature extraction from natural language requirements for reuse in Software Product Lines. The reuse of model-based requirements is beyond the scope of this research. The features extraction approach presented is limited to extract the software features, i.e. the user visible characteristics of a software system, and might not include the non-functional requirements such as user interface or security requirements. Additionally, the implementation of the solution in this research is limited to laboratory settings, and not yet tested for industrial practice.

1.6 Contributions of the Research

This research contributes to the body of knowledge in the area of requirements reuse for software product lines. The first outcome of this research is the Systematic Literature Review entitled "Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review", published in the Journal of Systems and Software (Bakar, Kasirun, & Salleh, 2015).

University of Malaya

Secondly, the preliminary investigation conducted through survey contributes to the software engineering research pertaining to the state of the practice of requirements reuse in Malaysian context. This second outcome was published in (Bakar & Kasirun, 2014). The details implementation of the feature extraction approach is presented in Chapter 5 and the demonstration can be used to guide further research in this area. Additionally, the extracted data set is made available online in case of future replication is needed.

1.7 Significance of the Research

Prior to this research, there is no guideline or process model exists that consider the feature extraction from requirements that exists in natural language forms. This research demonstrates the process of feature extraction from natural language, and the step-by-step procedure is outlined for other researchers to further explore in the future.

1.8 Thesis Overview

This thesis is organised as follows: Chapter 2 presents the systematic literature reviews conducted for the features extraction problems in requirements reuse. Chapter 3 provides an overview of the research design. Chapter 4 gives an account of the preliminary investigation made towards the state of requirements reuse practice among software practitioners. Chapter 5 presents the implementation of Feature Extraction from Natural Language requirement (FENL). Chapter 6 explains the evaluation strategy and discusses the results of the conducted FENL experiment. Lastly, Chapter 7 summarises the contributions of this thesis.

University of Malaya

CHAPTER 2: SYSTEMATIC LITERATURE REVIEW OF FEATURE EXTRACTION APPROACHES FROM NATURAL LANGUAGE FOR REUSE

2.1 Introduction

Software Product Lines Engineering (SPLE) refers to software engineering methods, tools, and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production (Northrop & Clements, n.d.). These shared software assets or sometimes referred to as core assets may include all artefacts in the product lines: requirements, architecture, codes, test plans, and more (Pohl et al., 2005). Meanwhile, requirements reuse (RR) is the process of reusing previously defined requirements for an earlier product and applying them to a new, similar product. Generally, RR can produce more benefits than only the design code reuse since it is done earlier in the software development (Clements & Northrop, 2002). When RR was planned systematically in the SPLE context, several studies (Eriksson, Borstler, & Borg, 2006)(Monzon, 2008)(Moros, Toval, Rosique, & Sánchez, 2012)(Knethen et al., 2002) indicated positive improvement in software development: speed up time to market, increase team productivity, reduce development costs in the long run, and provide a better way of sustaining core assets' traceability and maintainability. Software requirements can be reused either in an ad hoc basis such as in clone and own applications, software maintenance, or when systematically planned in SPLE. However, many problems exist when dealing with ad hoc reuse of natural language (NL) requirements. The problems with manual requirements reuse includes arduous (Weston et al., 2009), costly (Niu & Easterbrook, 2008), error-prone (Ferrari et al., 2013), and labour-intensive (Boutkova & Houdek, 2011) process, especially when dealing with large requirements.

University of Malaya

In the following subsections, the terms that bring together features extraction and RR in the SPLE context is described: Requirements versus features, core assets development in SPLE, and the contributions of our work in SPLE.

2.1.1 Requirements versus features

Firstly, it is important to understand the key distinction between software requirements and features. Software requirements describe the functionality of a software system to be developed. The definition of software *requirements* in accordance with IEEE Standard Glossary of Software Engineering Terminology, page 62 in (*IEEE Computer Society (1990). "IEEE Standard Glossary of Software Engineering Terminology". IEEE Standard.*, n.d.) is given as:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in (1) or (2).

The majority of requirements are written in NL (Denger, Berry, & Kamsties, 2003). This is because text is commonly used to convey information to communicate stakeholders' needs (Niu & Easterbrook, 2008). Pohl et al. (Pohl et al., 2005) emphasised that in SPLE, software requirements are documented either by using NL or model-based. As an example, NL requirements do not only appear in the form of Software Requirements Specification (SRS) format. NL requirements can also be recorded in the forms of goals and features, product descriptions including product brochures, user manual, or scenarios. Model-based requirements can be recorded in the

University of Malaya

forms of functional data analysis such as data flow diagram, UML models such as class diagram, state dependent system behaviour and more, and they are usually supplemented by NL descriptions of features (Nicolás & Toval, 2009).

Meanwhile, software feature is defined as a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems (K. Kang, Cohen, Hess, Novak, & Peterson, 1990). In most cases, requirements tend to be lengthy in nature, while features represent services that a system must provide to fulfil customers' needs, most of the time in a shorter or precise manner. Software features tend to be more focused and granular as compared to software requirements.

2.1.2 Core assets development in SPLE

Fundamentally, in SPLE, core assets (including requirements) can be developed through three approaches: ***proactive, reactive,*** or ***extractive*** (Krueger, 1992). In the proactive approach, assets are developed prior to software development. In the reactive approach, common and variable artefacts are iteratively developed during the software development. Reuse in the context of extractive tends to be in between the proactive and reactive (Krueger, 2002). To ease the transition from single systems to software mass customisation, Krueger proposed the extractive adoption model as a means to reuse existing products for SPLE (Krueger, 2001). With the extractive approach, core assets are no longer created from scratch, but extracted from the existing repository and reused in developing similar system. The extractive approach is particularly very effective with organisations that have accumulated development experience and artefacts in a domain and intended to quickly shift from conventional software development to SPLE (Frakes & Kang, 2005). Nan Niu and Easterbrook highlighted the basic tenets of extractive approach of Software Product Lines (SPL) that include maximal reuse and reactive

University of Malaya

development, particularly for small and medium-sized enterprises(Niu & Easterbrook, 2008).

2.1.3 Contributions of this work in SPLE

Up to date, various research works have been produced in SPLE focusing on the product line architecture, domain analysis tools (Lisboa et al., 2010), variability management (L. Chen & Ali Babar, 2011)(Metzger & Pohl, 2014), detailed design, and code reuse (Faulk, 2001). However, there are few works that looked at the extractions of features from the requirements in SPLE (Niu & Easterbrook, 2008)(Alves et al., 2008)(Kumaki, Tsuchiya, Washizaki, & Fukazawa, 2012)(Davril et al., 2013). Therefore, more parties can benefit from the formulation of feature extractions from NL requirements when various forms of input (not only SRS) are taken into consideration. In particular, we are interested in how current approaches that are used to extract features from NL requirements can support the reuse of requirements in SPL. Additionally, we are also looking at the implications for further research in this area. None of the related reviews presented in Section 2.2 adequately covers these issues. Figure 2.1 illustrates the scope of this SLR contribution in regard to other related works in SPLE.

University of Malaya

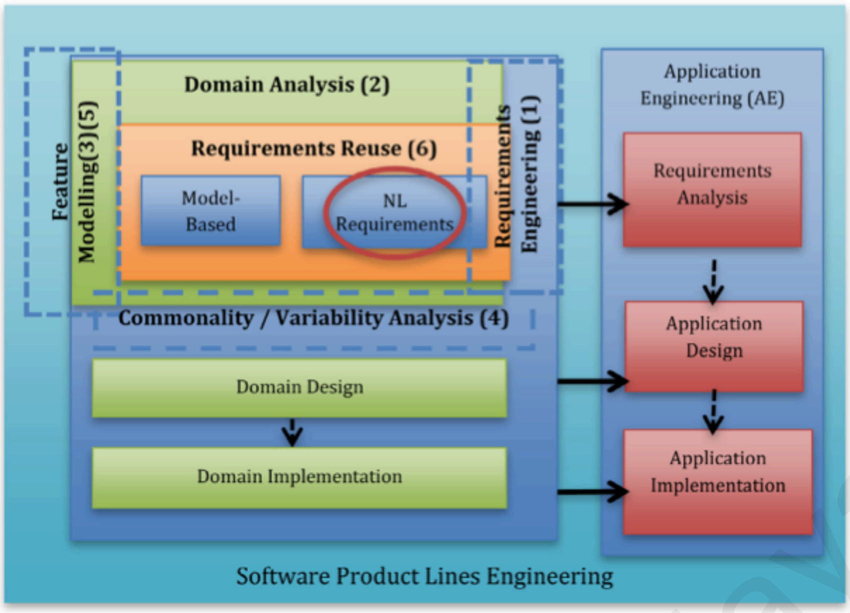


Figure 2.1 Contribution of this SLR to SPLE

SPLE is a paradigm to develop software applications (software intensive systems and software products) using platforms and mass customisation (Pohl et al., 2005). Meyer and Lehnerd (Meyer & Lehnerd, 1997) defined software platforms as a set of software subsystems and interfaces that form a common structure from which a set of derivative products can be efficiently developed and produced. The subsystems within a platform contain artefacts beyond source-codes which include requirements, architectures, test plans, and other items from the development process.

SPLE is distinct from the development of a single system, in which it involves two life cycles: Domain Engineering (DE) and Application Engineering (AE) (Pohl et al., 2005). In DE, the reusable assets (including requirements) are built. This is an entire process of reusing software assets for the production of a new similar system, with variation to meet customer demands. DE is responsible for defining and realising the commonality and the variability of software product line. On the other hand, AE is the process where the applications of the product lines are built by reusing the domain and exploiting the product line variability (Pohl et al., 2005). The most important part in

University of Malaya

Figure 2.1 is the Domain Analysis (DA), where a specific set of common and variable features from the existing requirement documents to be reused for developing similar product is identified. DA is the key method for realising systematic software reuse(Frakes & Kang, 2005). It can provide a generic description of the requirements (either in model-based or Natural Language form) for that class of systems and a set of approaches for their implementation(K. Kang et al., 1990).

The process of reusing requirements takes place within the DA process and it is a part of general Requirements Engineering. Reuse of software artefacts is the key aspect of SPLE. This is different to non-SPL based methodology in Software Engineering where requirements are gathered through elicitations of stakeholders' needs with or without using the existing documentation for similar systems. In normal RE, reuse of requirements is not planned systematically and always occurs in an ad hoc manner. Pohl describes Domain Design as a subprocess within DE that refines the variability into design variability, defining the reference architecture/platform (Pohl et al., 2005). Essentially, as a result, the outcome from all subprocesses within the DE phase should be the representation of most (if not all) possible application for a given domain. Related literature reviews around the DA area were numbered in Figure 2.1, and its summary is presented in Section 2.2.

Meanwhile, the second lifecycle, AE is concerned with the configuration of a product line into one concrete product based on the preferences and requirements of stakeholders produced in DE. Usually, the domain model produced within DE will now be used in AE. In AE, instance software products are often derived through the consultation with domain stakeholders that have specific requirements in mind (Bagheri & Ensan, 2013). Selection of desirable features that is now readily available should be

University of Malaya

gradually performed with ample interaction with the stakeholders, as described in (Czarnecki, Helsen, & Eisenecker, 2004) as staged configuration.

Various literature reviews have been published in the area of DE and AE (as numbered in Figure 1); however, none of the reviews reported the approaches used to select features from NL requirements for reuse in SPLE. This SLR was performed in order to obtain a better comprehension of the current state-of-the-art in feature extraction approaches from NL requirements for reuse in SPLE.

The key contributions of this SLR are as follows:

- it offers detailed comparisons of the published researches regarding the extraction of common and variable features from NL requirements for reuse in SPL through a Systematic Review; and
- it derives a number of key dimensions[2] of the feature extraction processes from the selected studies that will provide a structured overview of the attributes needed in RR for SPLE.

In particular, there are three specific objectives for this SLR:

a) To identify the approaches for extracting features from NL requirements for reuse in SPL.

b) To collectively summarise the quality of the approaches in the selected studies.

c) To identify research implications and highlight areas of improvement for RR research in the future.

[2] Some of these dimensions were discussed at the Information Retrieval Approaches in Software Evolution at 22nd IEEE Conference on Software Maintenance (ICSM '06): http://www.cs.wayne.edu/~amarcus/icsm2006, which were also used in B. Dit et al. "Feature location in source code: A taxonomy and survey" (Dit, Revelle, Gethers, & Poshyvanyk, 2013).

University of Malaya

This review may benefit a wide variety of audiences ranging from Information Sciences and Data Mining, Mathematical Computing, Data Management and more, particularly audiences with interests in Software Engineering. The implication of this review has opened up a lot of work that have direct or indirect effect on the scientific and practical community, namely research on making feature extractions fully automated, research on enhancing the available extraction and clustering methods by either being replicated, hybridised, or new ideas, research on enhancing the RR metrics, research on investigating the state of RR practice globally, research on exploring the opportunity for mathematical computing in aiding the RR process, and more.

In Section 2.2, the related literature reviews is summarised. Section 2.3 reports the organisation of the SLR process: the research questions, search process, inclusion and exclusion criteria, and study quality assessment. Section 2.4 presents the results of this review based on the synthesis of the evidence. Section 2.5 provides a discussion of open issues and research implications, and lastly Section 2.6 provides the concluding remarks.

2.2 Related Work

While conducting this review, there are other reviews found to be related to areas that are close to RR in SPL, namely Domain Analysis (DA), Requirements Engineering (RE) in SPL, and Automated Feature Modelling. This section provides a brief summary of the related reviews.

2.2.1 Requirements engineering for software product lines

Alves et al. reviewed the studies in the area of RE for SPL (Alves, Niu, Alves, & Valença, 2010). This work aims to assess the research quality, synthesise evidence to provide suggestions on important implications for practice, and provide a list of open

University of Malaya

problems and areas for improvements. This work differs from ours because it reviews selected work on general RE area for SPLE, while our work is more focused on the subarea of RE, the reuse of NL requirements in SPLE. A total of 49 studies between 1 January 1990 and 31 August 2009 have been selected for this review. Important findings from this review reveal that the overall quality of the reviewed studies needs improvement in terms of empirical validations. In addition to that, the authors report that most of the studies did not provide sufficient guidelines for practitioners to adopt the proposed approach. Furthermore, very limited commercial or open source tools are currently accessible, which hinders the practitioners' adoption of the proposed approach. As for the research trend, a growth in the number of approaches to handle NL requirements in a more automated way is anticipated in the future. In terms of the type of SPL adoption, proactive adoption was more common among the reviewed studies. However, this approach was very costly and the most risky. Thus, future work is expected to combine the use of the extractive and reactive SPL adoption. Lastly, Alves et al. conclude that future research should extend and improve the present research in an integrative manner (joint research and industry).

2.2.2 A systematic review of domain analysis solutions for product lines

Khurum and Gorschek conducted a review that covers a total of 89 primary studies in the DA solutions presented up until 2007 (Khurum & Gorschek, 2009). The findings reveal that although many DA approaches have been proposed, the absence of qualitative and quantitative results from empirical application makes it hard to evaluate the potential of the proposed approaches. In addition, many DA tools claim to base their approach on the need raised by the industry but fell short on the approach used to identify the need for a solution. Many studies claimed to apply or validate the proposed

University of Malaya

solution in industry. However, the claims made were not supported by any qualitative or quantitative evidence.

2.2.3 Literature Review on Automated Modeling

Benavides et al. provided a comprehensive literature review on the automated analysis of feature models for a period of 20 years, from 1990 to 2010 (Benavides, Segura, & Ruiz-Cortes, 2010). This review collates together various works in the area of automated feature modelling. The authors provide a conceptual framework to help understand different proposals in the area as well as categorise the future contributions. A total of 53 studies have been reviewed by the authors to answer three main research questions. As the main result, the authors present 30 analysis operations and classify the existing proposal providing automated support for them according to logical paradigm such as propositional logic, constraint programming, description logic, hybrid paradigm or multi-solver, studies that use their own tools, and proposals that present different operations with no support tools. In addition, the authors provide a summary of the tools used to perform the analysis, with the results and trends related to the performance evaluation of the published proposals. The identified challenges are mainly related to the formalisation and computational complexity of the operations, performance comparison of the approaches, and the support for the extended feature models.

2.2.4 A systematic review of evaluation of variability management

Variability management (VM) is an important area in SPL (Northrop & Clements, n.d.) and has been studied for almost 20 years since the early 1990s(K. Kang et al., 1990). The work in (L. Chen & Ali Babar, 2011) systematically investigates the evaluation of VM approaches. In addition, this work looks into the available evidence regarding the effectiveness of the VM evaluation performed in the selected studies.

University of Malaya

From the 97 selected studies, the authors identified 91 different types of VM approaches. Most of the approaches were based on feature modelling and/or UML-based techniques. In addition to that, only a small number of the approaches used other mechanisms to express variability such as NL, mathematical notations, and domain-specific language. The authors found that only a small number of the reviewed approaches had been evaluated rigorously by using scientific approaches. In addition, a large majority of them had never been evaluated in the industrial settings. Result of the reviewed studies indicates that the quality of the presented evidence is quite low. Hence, the authors conclude that the status of the evaluation of VM approaches in SPL is quite dissatisfactory.

2.2.5 Review on separation of concerns in feature diagram languages

Hubaux et al. conducted a systematic review of separation of concerns in feature diagram languages (Hubaux, Tun, & Heymans, 2013). In this work, the authors reviewed various concerns on feature diagrams and ways in which those concerns were separated. The four research questions they were trying to answer include: What are the main concerns of Feature Diagrams? How are concerns separated and composed? What is the degree of formality used to define Feature Diagrams? Is there any support tool available? A total of 127 papers were qualitatively analysed to answer the four research questions. Important findings include classifying the concerns in feature diagrams into feature groups and types of feature relationships. Concern feature groups can be further separated into functional and non-functional property, facets, and configuration processes. While concerns separating relationships among features are various, to name a few, the authors collected concerns relating to aggregation relationship, composed-of, concurrent activation dependency, conflict, excluded configuration, and more. A very

University of Malaya

detailed review and explanation of the techniques for composing concerns was also provided in this review.

2.2.6 Evaluation of a systematic approach in requirements reuse

Baretto et al. highlighted the reuse of requirement specifications by presenting a comparison of seven studies related to RR (Barreto, Benitti, & Cezario, 2013). Criteria used in the comparisons include the scope of reuse, characteristics of the approach, the support of some types of computational tools, and the evaluation done for the selected studies. They observed that six out of seven studies came from application in the SPL. When not applied in SPL, the reuse occurs in a very specific scope, namely in the real-time systems.

Although related studies presented in this section provide good information to the software engineering community regarding various issues in SPLE, none of the studies provide a thorough review of the approach that exists to extract features from NL requirements, from the SPL context. Knowing the available approach can be useful for researchers to identify what is available and what needs to be done in future research, and can be beneficial to practitioners for industry adoptions. Therefore, our SLR aims to contribute not only to the body of knowledge for RR, but also to the RE and SPLE practice in general.

2.3 Review Method

This section describes the process involved in conducting this SLR. Kitchenham and Charters described systematic literature review (SLR) as a process of identifying, assessing, and interpreting all available research evidence with the aim to answer specific research questions (Kitchenham & Charters, 2007). SLR provides a more systematic way to synthesise the research evidence by specifically using inclusion and

University of Malaya

exclusion criteria to set up the boundaries of evidence to be included in the review. In general, we are referring to Kitchenham and Charters' guidelines on performing SLR (Kitchenham & Charters, 2007); however, the guidelines on performing complementary snowballing search in locating articles to be included in the review suggested by (Wohlin & Prikladnicki, 2013) is as well considered. Additionally, this SLR also take into account the recommendations on the importance to include manual target search on popular venues as appeared in (Jørgensen & Shepperd, 2007).

2.3.1 Formulating Research Questions

Petticrew and Robert in (Petticrew & Roberts, 2006) suggested that the formulation of research questions should focus on five elements known as PICOC. Table 2.1 shows the *Population, Intervention, Comparison, Outcomes*, and *Context* for the current SLR's research questions.

University of Malaya

Table 2-1 Summary of PICOC

Population	Software requirements/specifications/software product reviews
Intervention	Feature extraction approaches
Comparison	None
Outcomes	The usability of the feature extraction approaches (empirical validation)
Context	Reviews of feature extraction approaches from all forms of requirements (textual-based) for reuse in the context of Software Product Lines

The primary focus of this SLR is to understand the available feature extraction approaches from the NL requirements to be reused in SPLE. In this SLR, all empirical studies presenting feature extraction approaches for NL requirements, specifically in the SPLE context are included. Comparison for feature extraction approaches in the PICOC is not included, as it is not applicable to our research objectives. Our SLR aims to answer the following Research Questions (RQ) that are formulated based on the PICOC:

Table 2-2 Research Questions for this SLR

RQ#	Research Question Details	
RQ1	What approaches are available to extract features from Natural Language Requirements in the context of Software Product Lines?	
	1.1	How are commonality and variability being addressed? Which technique is used?
	1.2	Is there any support tool available? If support tool is provided, is it Automated or Semi-automated?
RQ2	How was the evaluation performed against the proposed approaches?	
	2.1	What were the context, procedure, and measure used in the evaluation?
	2.2	What application domains were the studies tested or applied to?
	2.3	What procedures were used to evaluate the approach? Are proposed solutions in selected studies usable and useful? (Empirically validated?)

2.3.2 Identification of Relevant Literature

Based on Kitchenham and Charters's guidelines (Kitchenham & Charters, 2007), identification of relevant literature can be done by generating a search strategy. Initial search can be undertaken by using online database. However, there are some challenges to normal online database searches: mainly the nature of different interface for different database makes it difficult to use a standardised search string. Thus, making a complementary manual citation-based (snowballing) search is necessary (Wohlin & Prikladnicki, 2013) to minimise the possibility of missing important evidence. Additionally, Kitchenham and Charters (Kitchenham & Charters, 2007) also suggested that manual search from leading venues can bring out a number of high-quality articles that were not retrieved by the online and snowballing searches.

The article search process in this review is separated into three phases; Phase 1: Online Database Search, Phase 2: Complementary Citation-Based Search, and Phase 3: Manual Target Search.

University of Malaya

2.3.2.1 Phase 1: Online Database Search

Kitchenham and Charters (Kitchenham & Charters, 2007) used structured questions to construct search strings for use with the electronic database. To formulate the search string, we use the keywords derived from the PICOC (with synonyms and alternatives words). The Boolean search OR is used to incorporate synonyms and alternative words. The Boolean AND is used to link the major terms from population, intervention, and context. Therefore, the complete search string derived is:

```
(("feature extraction" OR "feature mining" OR "feature clustering" OR "feature similarity") AND ("natural language" OR "requirement" OR "textual requirement" OR "product description" OR "product specification" OR "product review") AND ("Software Product Lines" OR "product family" OR "software family"))
```

The following five databases that consist of Computer Sciences and Software Engineering articles are searched: ACM, IEEE Xplore, ScienceDirect, Springer, and Scopus. In the initial selection, the Inclusion and Exclusion criteria were applied and irrelevant studies were removed based on screening of titles and abstract. When the titles and abstracts were not sufficient to identify the relevance of the paper, the full text was then referred to.

2.3.2.2 Phase 2: Complementary Citation-Based Search

In Phase 2, the author used the citation-based search to find who cited the selected papers from Phase 1. The references from each selected paper (backward snowballing) are examined the titles that are relevant to the current SLR are listed down. In addition, Google Scholar is also referred to find out who have cited their papers (forward snowballing) and the titles that look relevant to this SLR are listed out. Selected papers from both citation-based searches (backward and forward snowballing) were compiled in a list and any duplicate studies were removed. Inclusion and exclusion criteria were

University of Malaya

applied when skimming the title and abstracts. Papers with poorly written abstract were downloaded and read to get more information. Only relevant articles are selected.

2.3.2.3 Phase 3: Manual Target Search

Despite the practical limitations related to the use of manual search such as the required search effort, manual target search has proven to bring high-quality search result when combined with the use of searches from digital library (Jørgensen & Shepperd, 2007). Manual target search from the most relevant venues in Software Engineering and Requirements Engineering fields in the article search process were included. Twelve leading journals were manually searched: Information and Software Technology, Journal of Systems and Software, IEEE Transactions on Software Engineering, IEEE Software, IEEE System Journal, ACM Computing Surveys, ACM Transactions on Software Engineering and Methodology, Software Practice and Experience, Empirical Software Engineering Journal, Requirements Engineering Journal, IET Software, and Automated Software Engineering Journal. The journals were selected because they were known to have been used as sources for other SLRs related to the author's research topic (Alves et al., 2010), (Benavides et al., 2010), and (Barreto et al., 2013). Additionally, the following conferences and workshop are searched manually too: International Conference on Software Engineering (ICSE), International Software Product Lines Conference (SPLC), Requirements Engineering Conferences (RE), International Conference on Software Reuse (ICSR), International Conference on Aspect-Oriented Software Development (AOSD), International Symposium on Foundations of Software Engineering (FSE), and International Workshop on Variability Modelling of Software Intensive Systems (VaMOS). These sources were selected because they presented a collection of flagship venues on SPL and RE. The author has

University of Malaya

searched for all papers published in the selected venues starting from January 2000 up until December 2014.

2.3.3 Selection of Studies

2.3.3.1 Inclusion and Exclusion Criteria

When conducting this review, it is necessary to set some criteria on which studies to be included and also those that need to be excluded. The candidate article is selected as one of the Primary Studies if it satisfied at least one of the inclusion criteria. Similarly, if a study fulfilled any of the exclusion criteria, then it will be excluded. The main inclusion criteria aim to only include all articles describing extraction approaches for NL requirements for reuse within the context of SPLE.

The main exclusion criteria comprised of articles that did not focus on feature extraction approaches for SPLE. Articles describing the ad hoc reuse or opportunistic approach, which clearly were not appropriately applied in the SPL context, were excluded. Additionally, articles that fulfilled any of the criteria listed below were excluded:

- Articles describing reusing model-based requirements (OOP model, feature model, or diagram), non-requirement artefacts in SPL (codes, test plans, architecture, etc.), or extraction of items not related to requirements (image extractions): Many articles describe research in the area of feature modelling: articles describing extension or improvement to elements in feature model, integrating specification into feature models, automated derivation from feature models, and more researchers related to Feature Modelling were excluded from our SLR. The author also found many articles mentioning

University of Malaya

feature extractions; however, these are related to image processing and pattern recognition.

- Short papers, proposals, lecture notes, summary of conference keynotes, work in progress reports, doctoral symposium papers, and posters: Articles describing the concepts of RR which appear in short papers, work in progress papers, or business model proposal for RR that are usually not empirically validated were excluded.
- Review papers (tertiary studies) related to the topic: The search string from online database has produced many tertiary studies (related literature review or survey papers). These are secondary studies and therefore were not included as primary studies in this SLR.
- Papers not written in English

2.3.4 Data Extraction Plan

Data Extraction plan is designed to accurately record the information obtained by the researchers from the primary studies (Kitchenham & Charters, 2007). The form for data extraction plan records the standard information as follows:

- Study ID
- Date of extraction
- Name of the study
- Title, Author, Publication Type (Journal/Conference), and details (if available)
- Website (if available)
- Answers obtained from each research question

2.3.5 Study Quality Assessment

When designing the study quality assessment, some of the questions from Quality Assessment section of the published literature are reused. Table 3 outlines six relevant criteria used to evaluate the quality of the selected studies, inspired by the quality

University of Malaya

assessment criteria for performing SLR used in (Dybå & Dingsøyr, 2008)(Leedy & Ormrod, 2013)(Petticrew & Roberts, 2006)(Salleh, Mendes, & Grundy, 2011) and the guidelines provided in (Lam, McDermid, & Vickers, 1997) pertaining to 10 steps towards systematic RR. The following ratio scales are used: Yes = 1 point, No = 0 point, and Partially = 0.5 point.   Table 2. 3 outlines these criteria:

Table 2-3 Quality Assessment Checklists

Item	Answer
QA1: Was the article refereed? (Leedy & Ormrod, 2013)	Yes/No
QA2: Was there a clear statement of the aims of the research? (Dybå & Dingsøyr, 2008)	Yes/No/Partially
QA3: Is there an adequate description of the context in which the research was carried out? (Dybå & Dingsøyr, 2008) For example, the problems that lead to the research are clearly stated, descriptions of research methodology used, study participants, etc.	Yes/No/Partially
QA4: Was the data collection done very well? For example, did the evaluation of proposed approach answer the research questions and did the paper provide a thorough discussion of the collected results? (Dybå & Dingsøyr, 2008)	Yes/No/Partially
QA5: Were the testing results rigorously analysed? (Petticrew & Roberts, 2006) For example, are there any software metrics provided in evaluating the test results, is there any threat to validity being presented in the study, etc.	Yes/No/Partially
QA6: Are any practitioner-based guidelines on requirements reuse being produced? Lam et al. suggested that practitioners' guidelines including producing explicit documentation is important to prevent reuse misuse. (Lam et al., 1997)	Yes/No/Partially

The author was responsible for reading and completing the checklist for all the selected studies. As a way to validate the data extraction, the first supervisor randomly selected 20% of the selected studies (in this case three papers were randomly picked by the supervisor). She then completed the QA checklist. Discrepancies found from the

University of Malaya

results were compared and discussed until a consensus is met. The template used for the data extraction and quality assessment is available in APPENDIX C.

2.4 Results

In this section, the synthesis of evidence from this SLR is presented. This begins with the analysis of the results from article searches, followed by the quality assessment results. Next, the following are the answers to the main research questions from Table 2.3.2.

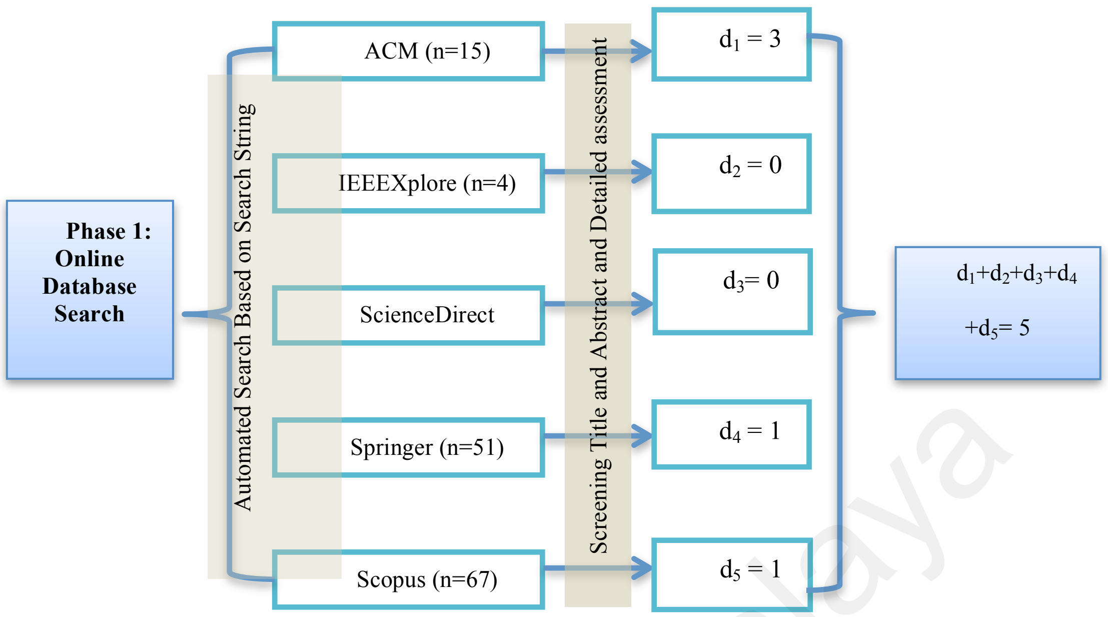
2.4.1 Results of Article Searches

As mentioned in Section 2.2, the article searches were divided into three phases: Online Database Search, Complementary Citation-Based Search, and Manual Target Search in journals and conferences. In this section, we will present the results of the search process.

2.4.2 Online Database Search

The results of the online database searches returned 168 hits. After screening the titles and abstract, and applying the Inclusion and Exclusion criteria, only five articles met the inclusion criteria. Figure 2.2 illustrates the result on the number of articles retrieved from the online database searches.

University of Malaya



Phase 1: Online Database Search
Automated Search Based on Search String
ACM (n=15)
IEEEXplore (n=4)
ScienceDirect
Springer (n=51)
Scopus (n=67)
Screening Title and Abstract and Detailed assessment
$d_1 = 3$
$d_2 = 0$
$d_3 = 0$
$d_4 = 1$
$d_5 = 1$
$d_1+d_2+d_3+d_4+d_5 = 5$

Figure 2.2 Results of online database search

2.4.3 Complementary Citation-Based Search (Snowball Search)

Based on selected studies in Phase 1, the backward and forward snowball searches were applied. Firstly, with backward snowball search, the author looked at the reference lists from the selected five articles from Phase 1. Six relevant papers were found from the first round of snowball search. Secondly, with forward snowball search, the author has performed searches on Google Scholar on who had cited each of the five papers. These backward and forward snowball searches were repeated until no new related article was found. After screening the titles or abstracts and applying the inclusion criteria, 13 additional papers are selected.

Figure 2.3 illustrates the result of the number of articles retrieved from the complimentary citation-based searches.

University of Malaya

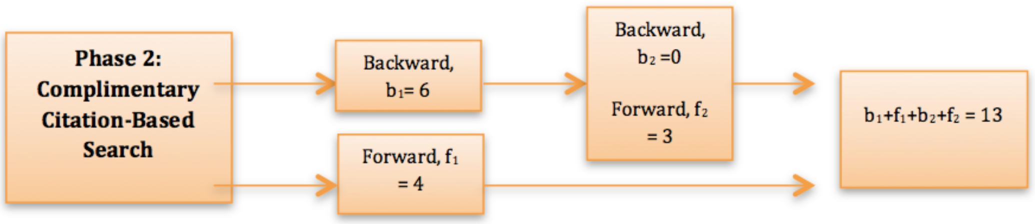


Figure 2.3 Results of complimentary citation-based search

2.4.4 Manual Target Search

As mentioned previously, the Manual Target Search was also performed to compliment automated search and snowball searches. Popular venues are used to manually locate papers that were possibly not reached by the Phase 1 and Phase 2 searches. Figure 2.4 illustrates the result of the Manual Target Searches.

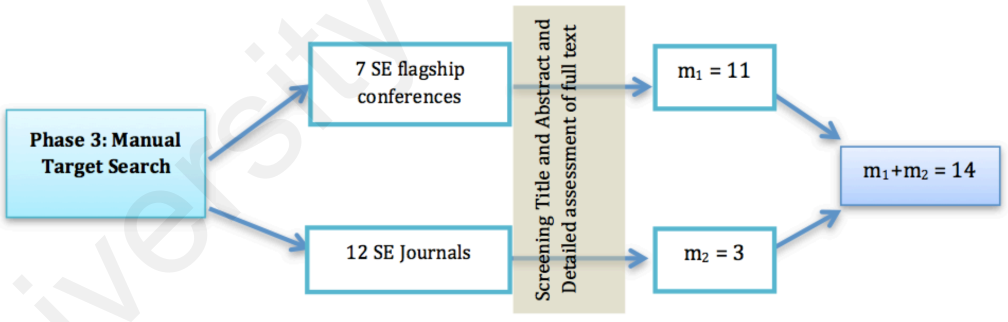

Phase 3: Manual Target Search
7 SE flagship conferences
12 SE Journals
Screening Title and Abstract and Detailed assessment of full text
m_1 = 11
m_2 = 3
m_1+m_2 = 14

Figure 2.4 Results of manual target search

Manual target searches are very important to ensure that no relevant study is missed. However, in current case, although the titles and abstracts from more than 6000 titles in journals and about 2000 articles in selected conferences were screened, no new study is retrieved. This indicates that the search string in Phase 1 is reliable and the snowball search in Phase 2 is sufficient.

University of Malaya

In total, this process has collected 32 articles from the three phases of article searches. However, after removing duplicates, only 13 studies are left. Duplicate entries are either articles that are already retrieved by the earlier searches or work from the same group of authors being published at different venues. For the second duplicate condition, only the most recent publication or the most comprehensive version of the articles is included (see Appendix A for the complete list of selected primary studies).

2.4.5 Publication Venues

Selected studies came from various publication venues with Software Product Line Conference as the most popular venue, followed by Requirements Engineering Conference as indicated by Table 2.4:

Table 2-4 Publication Venues for the Selected Studies

Venues	Selected Studies
International Software Product Lines Conference (SPLC)	S1, S3, S4, S5
Requirements Engineering Conferences (RE)	S6, S12, S13
International Workshop on Variability Modelling of Software Intensive Systems (VaMOS)	S7
IEEE Systems Journal	S2
IEEE Transaction Software Engineering	S8
Internetware	S10
International Conference on Information and Multimedia Technology (ICIMT)	S9
Automated Software Engineering Journal (ASE)	S11

There are duplicate publications found for three selected studies: S2 (three publications), S8 (three publications), and S12 (two publications). For example in S8, the three duplicate studies are (Dumitru et al., 2011), (Davril et al., 2013), and (Hariri, Castro-Herera, Mirarkholi, Cleland-Huang, & Mobasher, 2013). Two of the works were published in two conferences: ICSE 2011(Dumitru et al., 2011) and another one is in ESEC/FSE 2013(Davril et al., 2013). The other study, a more comprehensive one (Hariri et al., 2013), was published in a journal, the IEEE Transaction of Software Engineering. In general, duplicate studies would inevitably bias the result of the

University of Malaya

synthesis, hence only the most comprehensive version of the articles included, in the case of S8, only (Hariri et al., 2013) is selected as primary study.

2.4.6 Publication Chronology

The work on RR emerged as early as 1988, when Finkelstein published a paper in the Software Engineering Journal, entitled "Re-use of formatted requirements specifications" (Finkelstein, 1988). This is followed by other publications pertaining to reusing specifications through analogy, for example work by Maiden and Sutcliffe in 1992 (Maiden & Sutcliffe, 1992), a framework proposal on reuse of requirements and specification by Paredes and Fiadero in 1995 (Paredes & Fiadeiro, 1995), and work by Massonet and Lamswerdee in 1997 (Massonet & Lamsweerde, 1997). However, these works have either been restricted to small-scale academic example, use model-based requirements, or not describing the NL requirements for reuse. Additionally, these works were not specifically dedicated for the SPL domain, which clearly did not meet our main inclusion criteria. The paper by Lam, McDermit, and Vickers in (Lam et al., 1997) came out in 1997 describing the systematic RR relating to system families, which embark on the start of work on RR in the context of software family. Although this work did not specify the approach on how to reuse the NL requirements, it explains the experience of reusing requirements patterns at Rolls Royce and Smyth Industries in the domain of engine controller. Since our SLR is very focused on the extractions of features from requirements that appear in NL or textual based for reuse within the context of SPLE, this work by Lam, McDermit, and Vickers as well did not fit into our inclusion criteria. Then, the first formal conference for SPLC, the premium venue for SPLE was held in July 2000 (prior to this date, SPLC was done in the forms of

University of Malaya

symposium or workshop[3]) is identified. With this, it is credential that SPLE research topic has already achieved certain maturity for research publications, which potentially have published some works related to our interest. Thus, the year 2000 is used as the starting point for the automated searches of articles in databases. Unfortunately, only one study found to be relevant to the SLR's RQs, which was published five years later (in 2005) and appeared in Requirements Engineering conference – Chen et al. (K. Chen, Zhang, Zhao, & Mei, 2005). Other relevant studies appear from 2008 onwards. Based on this, 2005 is used as the year to start the complimentary manual searches. Thus, it becomes clear to that 2005 marks the emergence of the interest in feature extractions from NL requirements for SPLE. Figure 2.5 illustrates the distribution of the selected studies from 2005 to 2014, with 2013 as the major contributor. There was an increasing trend in the number of related publications across these years.

University of Malaya

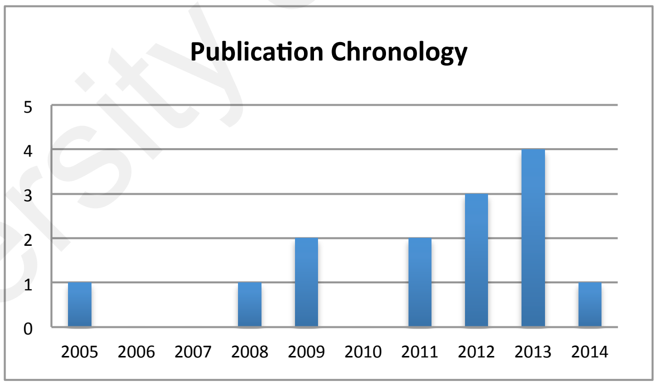


Figure 2.5 Distribution of papers from 2005 to 2014

2.4.7 Quality Assessment Results

A score scale of 0 to 6: Very Poor (Score < 2), Poor (Score of 2 to <3), Fair (Score of 3 to <4), Good (Score of 4 to <5), and Very Good (Score of 5 to 6) was used in the Quality Assessment. Most studies (11 studies) achieved the score of more than 4, which

[3] http://splc.net/history.html

are deemed to be of good quality. Two studies (15.39%) scored 3.5 and deemed to be of fair quality; one of the studies provided a very brief introduction to the problem they were investigating and the other study provided comprehensive numerical figures with less discussion on their testing results. However, it is identified that none of the studies claimed to have produced practitioners' guidelines for their feature extraction approach, but only explained the processes in the published academic paper.

2.4.8 Answering the Research Questions

The overall goal of this study is to review the current state of research in the area of feature extraction from NL requirements for reuse in the SPL. The transformation from the requirements in the NL documents to features can be done manually when dealing with small to moderate amount of requirements. However, this process can be arduous (Weston et al., 2009) when dealing with a large corpus of textual documents. For a large size of requirements, it is impossible for humans to manually analyse all feasible requirements for reuse (Falessi, Cantone, & Canfora, 2010). Thus, there is a need for automated or semi-automated approach to cater to this extraction process. In this section, the available approaches that extract the features from textual requirements are examined, based on the studies selected for this review. To provide more structured results, the research questions are answered through the key dimensions of the selected extraction approaches as outlined in Table 2. 5.

University of Malaya

Table 2-5 Research Questions and Dimensions in Reporting the Review

Research Question	Dimension	Example
RQ1: What approaches were available to extract features from Natural Language requirements?	Types of Input: Types of Output:	SRS Documents [S1,S2,S4,S5,S6] Product Descriptions/Product Brochures [S3,S7,S8,S9] User Comments [S13] Features [S3] [S13], Feature Tree/Feature Model [S4,S5,S6,S7,S8], Verb-phrase [S2], Clustered Requirements [S1,S8,S11]
RQ1.1: How were the commonality and variability addressed? Which technique was used?	Processes used	Text Preprocessing: Natural Language Processing (NLP) and Information Retrieval (IR) approaches [S2,S3,S7,S8,S9,S10,S12,S13] Similar Requirements Identification Latent Semantic Analysis/Vector Space Model (S1,S4,S5) Clustering of Features (See Table 2.7)
RQ1.2: Were there any support tools available? If support tools were provided, were they Automated or Semi-automated?	Availability of support tools:	Support tools: Automated support tool [S4,S5,S8] Semi-Automated support tool [S1,S2,S3,S6,S7, S13]
RQ2: How was the evaluation performed on the proposed approaches? RQ 2.1: Evaluation context, procedure, and measure used in the evaluation	Evaluation:	**Evaluation context:** Academia [S1,S2,S6], Industry [S3,S4,S5,S7] **Evaluation procedure:** Experiment [S1,S2,S3,S4,S6,S8, S13] Case Study [S2,S5] **Measure Used:** Recall [S8,S9, S10, S11], Precision [S8, S9, S10,], F-Measure [S9, S11, S13]
RQ 2.2: Domain Application	Domain Application:	Automarker Assignment [S2,S9] SmartHome [S4,S5] Antivirus [S8] Wiki [S7] MobileApps [S13]

2.4.8.1 RQ1: What approaches were available to extract features from Natural Language requirements?

Textual requirements were recorded in various forms. In seven studies (S1, S2, S4, S5, S6, S9, and S11), SRS has been used as the input to the extraction process. Four studies (S3, S7, S8, and S10) have used product descriptions and brochures, while the most recent work, S13, uses user comments as the input to feature extraction process.

University of Malaya

As for the output, feature trees or models were produced from the extraction process, as appeared in most of the studies (S4, S5, S6, S7, and S8). S3 was reported to produce features in the form of keywords. The output from the approach presented in S1 was in the form of classification of sentences (or clustered requirements), which were also reported in S10 and S11. Meanwhile, S2 and S9 were reported to have produced verb phrase or direct objects as the output of their feature extraction process. See APPENDIX B: Input (Types of Requirements) and Output (Features).

2.4.8.2 RQ1.1: How were the commonality and variability addressed? Which technique was used?

Feature extraction process involves selecting common or variant features from the requirements so that they can be seen in a more structured way. Commonality is defined as a set of mandatory characteristics that appear in SPL while variant features are characteristics that can be optional in SPL. To understand feature extraction process from NL requirements, it is worthwhile to investigate the approaches used, in which NLP was used by most selected studies in this review.

(a) ***Extracting Common Features: NLP approaches***

To classify the approaches used in extracting common features from NL requirements, the characterisation proposed in (Falessi et al., 2010) and (Falessi, Cantone, & Canfora, 2013) is used. Table 2.6 details out the types of NLP approaches across the selected studies in this review.

University of Malaya

Table 2-6 Various Feature Extraction Approaches from NLP

NLP Classification	Techniques	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
Algebraic Models	i) Vector Space Model	/			/									
	ii) Latent Semantic Analysis				/	/								
Text Preprocessing	i) Tokenisation		/					/	/				/	/
	ii) Part of Speech Tagging		/	/				/	/	/	/		/	/
Terms Weighting	i) Raw		/											
	ii) Hybrid (TF-IDF)						*		/	/	/			
Similarity Metrics	Vector Similarity Metrics (Cosine, Jaccard, Euclidean)	/								/	/			
NLP Tools	i) Stanford NLP								/	/	/	/	/	
	ii) Open NLP		/											
	iii) NLTK Toolkit													/
Thesaurus-based	WordNet											/		/

Note: Most selected studies used more than one NLP approach. The checkmarks here indicate approaches directly mentioned by the selected papers.
**S6 did not specify any NLP techniques, but used clustering algorithm.*

The following subsections briefly describe NLP techniques employed by the selected studies to aid feature extractions from the requirement documents for reuse in the SPL. Detailed descriptions on each of the NLP techniques mentioned in section (i), (ii), (iii), and (iv) can be found in [4].

i) Algebraic Model

Two techniques were found under the category of Algebraic Model: Vector Space Model (VSM) and Latent Semantic Analysis (LSA) (Falessi et al., 2010). VSM was used in two studies (S1 and S4), and LSA was mentioned by S4 and S5. In S1, Requirements and Structural models were used as objects to be analysed. Commonality and Variability for requirements and classes were analysed using cosine similarity calculation. In S4, an exploratory study was conducted to investigate the suitability of

[4] A detailed description of the NLP techniques is documented here: https://www.dropbox.com/s/yqnknjyp8mf6f3h/Descriptions%20of%20NLP%20Approaches.docx?dl=0

University of Malaya

Information Retrieval technique for identifying common and variable features by comparing the VSM and LSA (Alves et al., 2008). The framework was produced in an industrial context focusing on textual requirements. Comparisons were done towards a combination of Hierarchical Agglomerative Clustering (HAC) and LSA, as well as a combination of HAC and VSM, to observe which one would perform better. The findings of the study indicated that the textual requirement documents have latent structures that complemented both VSM and LSA. With small-sized requirements, VSM performed better than LSA.

In S5, the author described ArboCraft as a tool suite that can automatically process NL requirements into a feature model that later can be refined by the requirement engineers. This approach employed the LSA in terms of grouping similar requirements. In-text variability was identified through a tool that detected uncommon words. Requirements were considered similar if they concerned similar matters. Thus, in ArboCraft, the subject matters of requirements were compared, resulting in similar subject matters to be clustered together. The GUI representation of ArboCraft was presented to illustrate the feature tree construction resulting from the feature extraction.

ii) Text Preprocessing

Text preprocessing involves tokenisation, removing of stop words, and Parts Of Speech tagging (POS Tagging). In some of the reviewed work, tokenisation processes are also referred to as Lexical Analysis (LA) (S3, S7, S8, S9, S10). S2 and S3 indirectly reported applying the text preprocessing. LA was presented in S2 and verb-direct object extractions were mentioned there. Author in S2 proposed a semi-automated approach to identify functional requirements assets by analysing NL documents. The functional requirements in each document were identified on the basis of lexical affinities and

University of Malaya

"verb-direct object" relations (Niu & Easterbrook, 2008)(Niu, Savolainen, Niu, Jin, & Cheng, 2013). Fillmore's case theory was used to characterise each Functional Requirements Profile's (FRP) semantics. A verb followed by an object in a requirement sentence would be extracted as a FRP. The authors defined the FRP of a document to be the domain-aware LA that has a high information value and bears a verb-direct object relation. Fillmore's case theory was applied to each FRP, by filling up the details for six semantic cases. Then, Orthogonal Variability Modelling was used to rigorously express the variability. Mu et al. (Mu, Wang, & Guo, 2009) improved Nan Niu's FRP by proposing ten semantic cases instead of just six, naming it as Extended Functional Requirements Framework (EFRF). The extractions were done based on the structure of EFRF. The extraction process came in two phases: NLP and rule-based converting process. OVM and SRS were also used in this work.

Text preprocessing technique was also highlighted in S3 to identify common features in product brochures from various vendors (Ferrari et al., 2013) and also used when mining specifications for typical antivirus products in S8 (Hariri et al., 2013). In S3, conceptually independent expressions (i.e. terms) were identified through POS Tagging, Linguistic Filters (filtering terms with adjectives and nouns), and lastly identifying C-NC Value that computed term-hood metric. Then, Contrastive Analysis was applied to select the terms that were domain-specific. $C_1 \ldots C_n$ are sets of domain-specific terms for $D_1 \ldots D_n$ documents. Contrastive analysis is an approach in NL processing for extracting the domain-specific terms from textual documents. The aim of this technique is to refine the obtained result (from word extraction) either by filtering noise due to common words or by discriminating between semantically different types of terms within varied terminology (Bonin, Orletta, Venturi, & Montemagni, 2010). Ranking values were provided by calculating the average rank of each term. If a term is domain-specific and appears in all of the documents, it is more likely to be a common feature. If a domain-specific term appears in some of the documents of the different vendors, but not in all documents, it is more likely that it is a variant feature.

S8 proposed an approach to mine software features from publicly available product descriptions and construct feature model based on extracted features for typical antivirus products from the Internet (Hariri et al., 2013). The approach was divided into two primary phases: Mining Features from product descriptions and Building the Feature Model. Screen-scraper facility was used to scrape raw product descriptions from 165 antivirus products from the Internet. These product specifications were preprocessed by stemming each word to its morphological root and stop words were removed as well. The remaining descriptors were then modelled as a vector of terms.

iii) Terms Weighing

Terms Weighting or sometimes referred to as Weighting Schema is the mechanism used to assign different weights to terms based on its occurrences in the document (Falessi et al., 2010), in which $tfidf$ term-frequency-inverse-document frequency being mentioned in S8, S9, and S10. For example, in S9, $tfidf$ was used to assign the frequency of terms to occur in a processed document that would later be fed into the clustering algorithm.

iv) Similarity Metrics

Similarity Metrics refer to a specific formula used to compute the fraction of common words between two text fragments. A wide variety of measures can be used to group similar texts. Falessi et al. (Falessi et al., 2010), (Falessi et al., 2013) categorised the Similarity Metrics into two categories: Vector Similarity Metrics (Dice, Jaccard, &

University of Malaya

Cosine) and WordNet Similarity Metrics. None of the selected studies mentioned the use of Jaccard. Cosine Similarity Metrics were used in S1, S10, and S11. S1 reported using Cosine Similarity Metrics to detect similar requirement text and classes. Choosing different similarity measures may affect the quality of clustering common features. For further reference, the effects of choosing different similarity measures in clustering problems can be found in (Huang, 2008) and (Cui, Potok, & Palathingal, 2005).

v) NLP tools

Few selected studies mentioned using the open source NLP tools provided by Stanford NLP[5] (S8 and S11) while S2 used OpenNLP[6]. Extracted nouns were considered as candidate features, which can be further refined by the requirements engineer. Bagheri et al. in S11 used Stanford Name Entity Recogniser to train the NER model that was provided by The Stanford NLP Group for it to label features and integrity constraints (Bagheri, Ensan, & Gasevic, 2012). Additionally, the NLP Toolkit provided to aid with Python programming is mentioned by S13 during the text preprocessing stage. NLTK Toolkit is an open source platform used to build Python programmes that deal with human language data. The tool provides easy-to-use integration to suite text processing for classification, tokenisation, stemming, tagging, parsing, semantic reasoning, and more.[7]

vi) Thesaurus-based

[5] http://nlp.stanford.edu/software/index.shtml
[6] https://opennlp.apache.org
[7] http://www.nltk.org

University of Malaya

WordNet is an example of a thesaurus-based variant of algebraic model capable of handling a large collection of synonyms to compare terms. The purpose of WordNet is to function like thesaurus and dictionary, and it may be used as a knowledge base of individual words semantically (Falessi et al., 2013). WordNet was used in S11 and S13. S11 proposed a decision support platform during domain engineering phase to perform NLP tasks over domain documents and help domain analysts to identify domain information. This approach employed Name Entity Recogniser (NER) to identify features and integrity constraints from domain documents: features and integrity constraints were labelled accordingly to form the annotated document. Features identified were cross-referenced with term definitions provided by WordNet (Bagheri et al., 2012). This way, annotated features inside the documents would be interrelated with the concepts from the widely used and well-understood source. This approach employed the semantic annotations of the identified features to create feature graphs. Features that were similar to each other were placed close together, while those not in common were placed as far as possible. The distribution of these features on the graph would aid the analysts to identify the most related features. This visualisation of feature is able to form clusters of features and help analysts during the design of a domain model. The final step in this approach is to integrate the annotated domain documents and the visualisation graph into the MediaWiki format for easy collaboration among analysts. In S13, Guzman and Maalej used WordNet lemmatiser from NLTK to group different inflected forms of words with similar part of speech tags (semantically equal but syntactically different) (Guzman & Maalej, 2014). This step reduces the number of feature descriptors that needed to be inspected at later stages.

University of Malaya

(b) ***Extracting Common Features: Clustering approaches and more***

This SLR has also identified proposals from the selected studies that used other than NLP techniques to extract features from textual requirements. The approaches included various clustering algorithms, for example Hierarchical Agglomerative Clustering, K-Means, K-Medoids, and Fuzzy K-Means (see Table 2. 1). Other approaches that are beyond clustering such as Latent Dirichlet Allocation, Propositional Logic, and more are also listed in Table 2.7.

Table 2-7 List of Feature Extraction Approaches

Clustering Approaches	Paper(s)
Hierarchical Agglomerative Clustering	S6, S4
Incremental Diffusive Clustering	S8,S10
K-Means, K-Medoids	S9, S10
Fuzzy K-Means	S8
Miscellaneous Approaches	Paper(s)
Latent Dirichlet Allocation	S10, S13
Propositional Logic	S7
Contrastive Analysis	S3
Rule-Based Mining	S9
Association Mining	S8

Hariri et al. (2013) in S8 used data mining approach to find common features across products and also relationships among those features. An incremental diffusive clustering, IDC algorithm, was used to extract features from online product listings. Association mining was applied together with k-nearest neighbour machine learning method to analyse the relationships among features and make recommendations during the domain analysis process. The end results were a set of recommended features, which could be supplied to the requirements engineering process to help project stakeholders to define features for specific product lines.

University of Malaya

Chen et al. (2005) in S6 manually constructed requirements relationship graph from various requirements specification documents. Hierarchical clustering was also used in their work to merge requirements into feature trees. Unfortunately, the paper did not provide a detailed description on how this is obtained. Furthermore, this approach required heavy manual human involvement.

Latent Dirichlet Allocation (LDA) is a probabilistic distribution algorithm which uses Gibbs sampling to assign topics to documents. LDA was used in S10 (Yu, Wang, Yin, & Liu, 2013), together with an improved HAC algorithm to identify similar social feature elements from open source-based software repositories such as Sourgeforge.net, Softpedia.com, Onloh.com, and Freecode.com. The hidden relationships among the extracted features were mined and a recommender system was proposed to recommend relevant features to stakeholders. Students were asked to evaluate the questions. The findings from HESA reported achieving a reasonable precision (reasonable elements in a cluster) and relatively low deviations (performance across different domains, in this case they used Antivirus, Audio-Player, Browser, File Manager, Email, and Video Player during the testing). Additionally, LDA also appeared in Guzman and Maleej in S13 (Guzman & Maalej, 2014) to group features that tend to co-occur in the same user reviews of various mobile apps.

(c) ***Extracting Variant Features***

Not many works mentioned explicitly how variant features were extracted from NL requirements. This makes it hard for us to classify the approaches used in extracting variant features. Indirectly, features that were not classified or clustered were somehow regarded as variant features. For example, Kumaki et al. (2012) in S1 used VSM to determine the common or similar features, and manually determined the variant

University of Malaya

(leftover) features. In S3, Ferrari et al. (2013) identified conceptually independent expressions (i.e. terms) through POS Tagging, Linguistic Filters (filtering terms with adjectives and nouns), and lastly identifying C-NC Value that computed term hood metric. Then, Contrastive Analysis was applied to select the terms that were domain-specific. If a term is domain-specific and appears in all of the documents, it is more likely to be categorised as a common feature. If a domain-specific term appears in some of the documents of the different vendors, but not in all documents, it is more likely be considered as a variant feature. Variant candidates are identified as $V=\{C_1 \cup C_2 \ldots \cup C_n\} \setminus C$. In order to do that, human operator is needed to assess the relevancies for each of the variant candidates. Meanwhile, in S5, the authors described EA-Miner tool to detect and flag words that may denote the presence of variability. The enumerators like "such as, like, as follows, etc." and words about multitude like "different, various, etc.", may denotes the presence of alternatives in requirements text. EA Miner provides clues on how each extracted features can be reviewed against the textual clues on variability (Weston et al., 2009).

Archer et al. (2012) in S7 proposed an automated process, language, and support tool to extract variability for a family of product from product descriptions on public data (Acher et al., 2012). VariCell, the developed language, was proposed to extract features from the product line descriptions represented in a tabular form into a hierarchical form of feature model. An experiment was conducted that looked at eight different Wiki engines that form a family of product. Their aim was to build a model for this product line that represents the commonalities and variabilities of those eight Wiki Engines. VariCell allowed the parsing, scoping, organising, and transforming product descriptions into a set of feature model. Product descriptions were extracted into tabular form, employing Comma Separated Value (CSV) format, with some user involvement.

University of Malaya

Five variability patterns were found: mandatory, optional, dead feature, multivalue, and real value.

Ebrahim et al. (2012) in S11 trained the NER model that was provided by the Stanford NLP group for it to label features and integrity constraints, but did not offer an approach that would extract the structural relations between the features (i.e. type of variant features: alternatives or optional). It still remains as a challenge within a NLP approach to automatically classify variant features by only performing NLP programming.

S2 and S9 transformed the extracted semantic cases into Orthogonal Variability Model to show the variant features. The results indicated that EFRF extraction in S9 can extract EFRFs to help generate the functional variability models and save manual efforts. However, this demand further explanation on how it can be done as the paper did not further elaborate on how to handle variants feature extraction.

2.4.8.3 RQ1.2: Are there any support tools available?

It is not easy to precisely categorise the approach to whether they have provided any support tools or not. Most studies were implicitly reported to provide semi-automated tools, in which at least the text preprocessing and clustering of features used automated approaches. This could be due to most approaches were validated in experimental or research settings, in which most likely tools provided are not fully automated. We only identified six studies to have named their support tools: ARBOCRAFT (S5), VariCell (S7), CoSS (S8), HESA (S10), AUFM (S11), and MIA (S12). The rest were mentioned as approaches only, with no specific tool names given.

University of Malaya

2.4.8.4 RQ2: How was the evaluation being performed against the proposed approach?

The second objective of this review is to assess the quality of the mechanism used in evaluating the approach proposed in the selected studies. For this, we will report on the context, subjects, evaluation procedures, and measures used in the evaluation. In addition, this section also reports the domain application involved in the studies.

(a) Evaluation context

Out of the 13 studies selected, seven studies reported having evaluation done in the industrial settings: S3, S4, S5, S7, S8, S9, and S11. The remaining six were done in academia: S1, S2, S6, S10, S12, and S13. Figure 6 indicates these distributions.

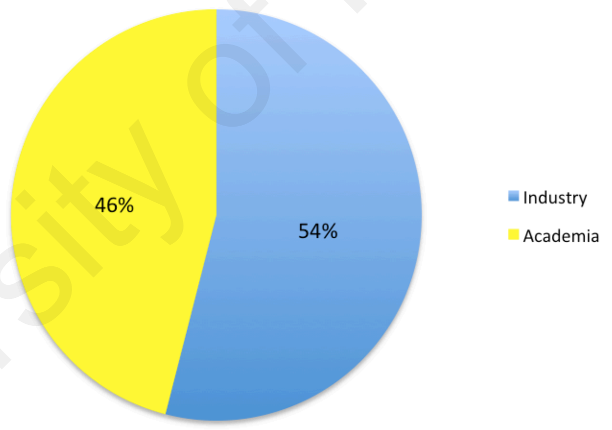
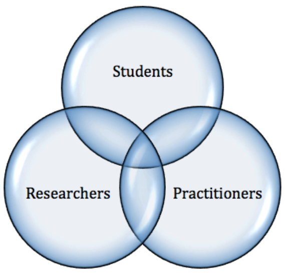


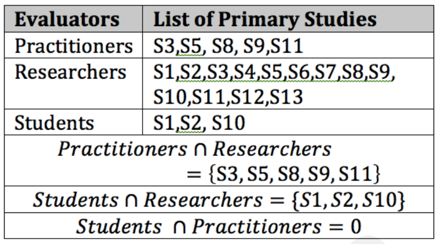
Figure 2.6 Evaluation context

Most of the studies were done for research purposes from the industrial or academic settings, or as a joint research-industry work. From the 13 selected studies, five studies were reported having the actual practitioners' involvement during the evaluation. Four studies used students as the evaluators and the remaining used researchers as their evaluators. As for collaborative work, five studies reported having researchers–

University of Malaya

practitioners' collaboration and four studies reported the collaboration between students and researchers. Figure 2.1 illustrates the summary of this result.





Evaluators	List of Primary Studies
Practitioners	S3,S5, S8, S9,S11
Researchers	S1,S2,S3,S4,S5,S6,S7,S8,S9, S10,S11,S12,S13
Students	S1,S2, S10
$Practitioners \cap Researchers = \{S3, S5, S8, S9, S11\}$	
$Students \cap Researchers = \{S1, S2, S10\}$	
$Students \cap Practitioners = 0$	

Figure 2.7 Evaluators

(b) ***Evaluation Method***

The majority of the selected studies employed quantitative method while evaluating their proposed approach with experiments to be the most popular method (S1, S2, S3, S4, S6, S7, S8, S9, S11, and S13), followed by case studies as reported in S2 and S5. Additionally, expert opinion was used in S2 through semi-structured interview as an additional effort in measuring the validity of their experiment. Feature extraction approach in S12 was employed and tested at an automotive industry, the Daimler Chrysler.

(c) ***Measure Used***

In Software Engineering research, metrics are useful to improve software productivity and quality. Apart from having experiment or case studies, the use of software metrics in evaluating the performance of proposed approach is essential too. However, not all the selected studies in this review reported using software metrics in evaluating their approach. Table 2.8 details out the metrics used by eight studies.

University of Malaya

Table 2-8 Measure Used in Selected Studies

Metrics	List of studies
Purity[8]	S8
Entropy[9]	S4, S10
Recall and Precision[10]	S2, S8, S9, S10,S11,S12, S13
F-measure	S9, S10, S11, S12, S13

Purity is calculated by comparing the clusters generated by the algorithm to the answer set clusters. Each cluster generated is then matched with the set clusters with which it shares the most descriptors (Hariri et al., 2013). Purity is measured as:

$$Purity(w,c) = \frac{1}{N}\sum_{k} max_j \left|w_k \cap \, c_j\right|$$

Let $w = \{w_1,w_2,\ldots,w_n\}$ be the set of clusters found by a clustering algorithm.

$c = \{c_1,\ldots c_j\}$ be the set of classes

Purity may take values between 0 to 1 with a perfect clustering solution having a purity value close to 1, and a poor clustering solution holding a value close to 0. In the context of this SLR, S8 reported employing purity together with Recall and Precision in their work.

Entropy is a measure of the average information content one is missing when one does not know the value of the random variable. Entropy is measured as:

$$H = -\sum_{i=1}^{n} Pi \log_2 Pi$$

University of Malaya

[8] C.D. Manning, P. Raghavan, and H. Schutze, "Introduction to Information Retrieval". Cambridge Univ. Press, 2008.

[9] Shannon, Claude E. (July-October 1948). "A Mathematical Theory of Communication". Bell System Technical Journal 27(3): 379-423. doi:10.1002/j.1538-7305.1948.tb01338.x.

[10] *Source: http://en.wikipedia.org/wiki/Precision_and_recall*

The index *H* equals 0 in the case of perfect clustering and log k in the case of maximum heterogeneity (dissimilar). From the 13 selected studies, S4 and S10 reported using this measure in their research.

Precision is the probability that a (randomly selected) retrieved document is relevant. Recall is the probability that a (randomly selected) document is retrieved in a search. F-measure is a metric that combines recall and precision. Observing results in Table 2.11, Recall and Precision, and its variation the F-measure were reported to be the most popular metrics used by the selected studies.

Other works reported the use of cost estimation and man hours (S1, S3), comparing the result produced by the algorithm and manually produced by experts (S5), time complexity, and cluster quality with independency metric (S6) in their evaluation.

(d) ***Application domain***

Selected studies were tested in various application domains including auto marker assignment (S2, S9), Robot Design Contest (S1), Antivirus Products (S10), Transportation (S2, S3), Media Wiki (S7), Smart Home (S4 and S5), MobileApps (S13), and Library Management Systems (S6).

2.5 Discussion

This section firstly presents a discussion on the implications of this study (Section 2.5.1). Later in Section 2.5.2, the author presents a discussion on the reduction in the number of selected studies in this article when comparing with other related work followed by a discussion on the threats to validity in Section 2.5.3.

2.5.1 SLR Study Implications

This section discusses the implications made from the literature study conducted.

University of Malaya

2.5.1.1 SRS documents as the main input to the extraction process

SRS or Requirements documents were found to be the most frequently used input to the feature extraction process. Product line requirements define the product lines together with the features and constraints of those products. Most features are high-level functional requirements. Thus, more than one feature can be found by extracting key terms from the functional requirement documentation. As compared to using product descriptions from publicly available brochures, SRS documentation is more structured and may consist of technical details because it is meant to be read by the development team, while product description from publicly available brochures is more general in nature as it is intended for potential customers. The nature of SRS documentation allows easier feature extraction process. Commonly, SRS for earlier products were already tested and underwent several refinement phases and thus the risk of including NL ambiguity may be reduced. The nature of SRS documentation sentences which specifies the verb (functionality) and object makes it easier for feature extraction process, as features are defined as end-user's visible characteristics of a product (K. Kang et al., 1990). The popular usage of SRS documentation in feature extraction area might imply the adoption of systematic process of reuse such as extractive adoption model (Krueger, 2001) as a means to extract the core assets from existing software assets (in this case, the SRS documentation being the software assets).

Although detailed description from SRS documentation is good for feature extraction, SRS documentation is not easily accessible by everyone due to the company privacy or copyright issues. This SRS unavailability and inaccessibility can be the contributing factors to the lower number of publication in the RR research area. In fact, this is reported in (Bakar & Kasirun, 2014). In case where SRS documents are not available, product descriptions from brochures and user comments were used as

University of Malaya

alternatives to extract common and variable features for a product line as appeared in selected studies in recent years (Davril et al., 2013), (Yu et al., 2013), (Acher et al., 2012), and (Guzman & Maalej, 2014).

2.5.1.2 Feature extraction approaches were done in phases and supported with semi-automated tools

As an overall picture from reviewing the selected studies, the feature extraction process for reuse of NL requirements can be separated into four phases: Phase 1: Assessing the Requirements, Phase 2: Terms Extraction (Tokenisation, POS Tagging, and Stemming), Phase 3: Features Identifications and Phase 4: Formation of Feature Model. The overall feature extraction process for RR is depicted in Figure 2.8 and its details pertaining to input, process, and output is presented in Table 2.12. This process can be interpreted as taxonomy since it provides detailed granularity on the available processes and approaches, which ~~is~~ are very useful for practitioners and researchers interested in this area to guide future research, development, and implementation.

University of Malaya

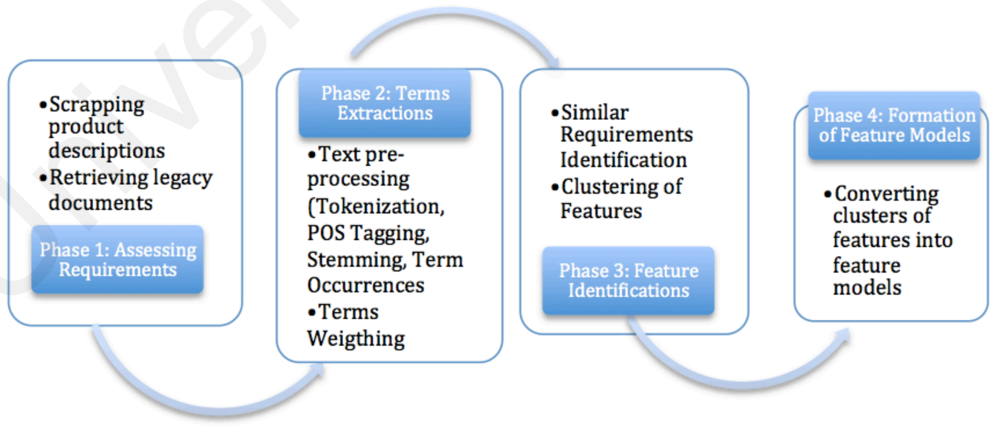

•Scrapping product descriptions
•Retrieving legacy documents
Phase 1: Assessing Requirements
Phase 2: Terms Extractions
•Text pre-processing (Tokenization, POS Tagging, Stemming, Term Occurrences
•Terms Weigthing
•Similar Requirements Identification
•Clustering of Features
Phase 3: Feature Identifications
Phase 4: Formation of Feature Models
•Converting clusters of features into feature models

Figure 2.8 Feature extraction process for requirements reuse

Most primary studies proposed at least semi-automated tools for Phase 3 and Phase 4, with Phase 1 to be done manually. Requirements in the form of product descriptions, brochures, or online customer comments may be automatically scraped using open source scraping tools available on the Internet. On the other hand, legacy documents from similar systems can be retrieved manually by requirements analyst prior to term extraction process. As for Phase 2, term extractions were mostly done using automated process available from NLP.

Many selected studies used POS tagging (e.g., S2, S3, S8, S9, and S11) for term extraction with the openly available tools such as StanfordNLP, OpenNLP, or NLP Toolkit. Phase 3 is subdivided into two smaller phases: Similar Requirements Identification and Clustering of Common and Variant Features. Similar requirements can be determined by using the LSA (S1 and S4) or VSM (S4 and S5).

Table 2-9 Input, Process, and Output for Feature Extraction Process

Phase	**Input**	**Process & Tools (if available) (A=Auto, S=Semi-auto, M=Manual)**	**Output**
1:Assessing Requirements	Product Descriptions, Brochures, Legacy Requirement Documents, Use Case Descriptions, User Comments	Scraping (A), Search and Retrieving (S), Copying and Pasting (M) Tools: Open source scrapping tools available on the Internet	Domain-Specific Documents or Collection of Natural Language Requirements
2: Term Extractions	Collection of Domain Specific Requirement Documents	Text preprocessing (A), Terms-Weighting (A) Tools: Stanford NLP (A), OpenNLP (A), NLTK Toolkit (A) Thesaurus-Based: WordNet (A)	Terms-Documents-Matrix, Keywords, Nouns, Verbs, and Objects
3:Feature Identifications	Terms-Documents-Matrix, Keywords, Nouns, Verbs, and Objects	Similar Requirements Identification (S), Approach: LSA (S), VSM (A) Clustering of Features (S) Approach: Clustering Techniques	Document Similarity Distance Clusters of Features
4:Formation of Feature Model	Clusters of Features	C & V Analysis (S)	Feature Trees/Models

The problem found with using VSM was that this approach ignored the semantic meaning of the identified features and thus some other significant features might be ignored. This problem was addressed by LSA by providing the semantic matching (that

University of Malaya

includes polysemy and synonymy) with LSA to provide the matching with items from similar domain. However, LSA might ignore some significant features due to the noise reduction while applying the Singular Value Decomposition algorithm and thus human intervention is definitely still needed during the feature extraction process. Limitations of LSA that were mentioned in (Hoenkamp, 2011) included scalability issues and LSA did not recover optimal semantic factors as it supposed to, resulting in more proposal to enhance the LSA algorithm. To improve similar requirements identification, some studies proposed clustering algorithm used in Information Retrieval area such as Incremental Diffusive Clustering or IDC (as appeared in (Hariri et al., 2013)), Centroid-based clustering (Casamayor, Godoy, & Campo, 2012; Davril et al., 2013; Yu et al., 2013)), K-means clustering (as appeared again in S9 and S10), integrating Hierarchical Agglomerative Clustering (HAC) in S6 and S4, Latent Dirichlet Allocation (LDA)in S10 and S13, and more.

As for Phase 4, among the selected studies, research effort on formation of feature tree or models are explicitly discussed in S4, S5, S6, S7, and S8.

2.5.1.3 The evaluation metrics and evaluators

The use of software metrics primarily serves to provide quantification on the entire reuse process, such as gauging the efficiency or productivity of a process, finding defects, or even estimating costs. From the selected studies, eight mentioned about quantitatively measuring their proposed approach by using either Purity, Entropy, Recall and Precision, or F-measure. Recall and Precision was reported to be popular among all selected studies. Although it is easy to implement, measuring recall requires certain conditions. User needs to determine the actual relevant records that exist, however most of the time, recall is estimated by identifying a pool of relevant records

University of Malaya

and then determine what proportion of that record exists, which may require manual human judgements. In order to adopt suitable metrics for the problem context, one should consider looking at the purpose, scope, attributes, scale of the measure, and result expected from the measures used (see guidelines provided by the IEEE for details on software metric criteria). Additionally, the use of Goal Question Metrics, GQM by Basili (Basili & Rombach, 1994) is essential for the same purpose. Since no dedicated RR metrics were proposed, future research on suitable metrics in the context of RR for SPLE would be an interesting opportunity to explore. Similarly, issues on measuring variabilities and metrics on performance of variability techniques have been recently highlighted in (Metzger & Pohl, 2014) as open research challenges. More discussions on handling and measuring variability will be presented in Section 2.5.1.5 of this chapter.

Pertaining to subjects used or the evaluators in the evaluation, all studies involve researchers (researchers or research students) as the evaluators in the evaluation (see Fig. 2.7). This may introduce bias towards the evaluation results. Since the researchers already informed about the proposed approach, time taken for them to use the approach might be lesser as compared to real practitioners. Ideally, evaluation subjects should be addressed to the actual practitioners. However, it is understood that research-industry collaboration requires additional effort, while most researchers which are postgraduate students tend to have very limited time frame in their research.

2.5.1.4 Practitioners' guidelines and support tools

A question in the Quality Assessment have been included pertaining to the availability of practitioners' guidelines in the selected studies. The importance of having practitioners' guidelines for RR had been highlighted in (Lam et al., 1997) back in

University of Malaya

1997. Although most studies described the methodology used in their work, none has reported to implicitly produce practitioners' guidelines. This could be due to the research carried out has not reached the appropriate maturity level at the time the work was published. Additionally, this finding confirms an earlier observation of lack of practitioners' guidelines for SPLE of which RR is an important aspect, as reported for the period of 1990 to 2009 in (Alves et al., 2010).

Pertaining to support tools, most studies did not clearly mention whether they have produced support tools that were made available publicly. Some feature extraction approaches produced resulted from the fundamental research experiments and no actual tools were produced, instead a theoretical experimentation was set up for research purpose only. Some other tools were made for research purposes and no longer maintained at the moment this paper is written, thus making it less convenient for researchers to explore or for practitioners' adoption.

2.5.1.5 Automated variant features extraction remains as a challenge

Selected studies indicate how to extract common features from the requirements by using approaches from NLP, IR, or even hybrid approaches. Relationships between identified features may provide some information on variability. Additionally, textual requirements express variability by certain keywords or phrases, but this may introduce ambiguity (Pohl et al., 2005). The process of variant identification either requires manual intervention or some approaches use too complex calculations and algorithms. Moreover, very limited demonstration or support tools were made available publicly, making automated variant features extraction from NL requirement remains as a challenge.

University of Malaya

To minimise this, requirements variability needs to be expressed either through explicit variability modelling or developers need to use the model-based requirements, which reflects why most of the selected studies use feature models or Orthogonal Variability Models (OVM) when handling requirements variability. This is why feature models were reported to be the most frequently reported notation in the industry when it comes to handling variability. Moreover, the popular usage of feature models (instead of textual-based) when it comes to handling variability partially indicates why we have less number of selected studies.

Prior to transforming features into feature models, for example, experimental settings in S1 and S3 used subsets formula to help identify variability candidates. S2 and S10 mentioned transforming extracted functional requirements profile into OVM such as the use of XML tags. S6 derived variability information from product descriptions based on patterns, whereas patterns were used to guide the selection of variant features in S7 and S8.

A SLR conducted by Chen and Babar in 2011 (L. Chen & Ali Babar, 2011) reported that a large majority of the variability approaches are based on feature modelling or UML-based techniques, and only small number reported on the mechanism of expressing variability through mathematical notations, natural languages, or domain specific language. Additionally, many methods for handling variability (in common) suffer from lack of testing. This was mentioned earlier in this chapter and also reported in a recent SLR publication in 2014, pertaining to variability in software systems (Galster, Weyns, Tofan, Michalik, & Avgeriou, 2014).

Allowing other researchers in the area to understand the variability handling approaches properly by providing the details of the research design in publications or

University of Malaya

manuals will definitely make the studies more attractive to practitioners and open up rooms for future improvements and research explorations. Lastly, more empirical experiments should be conducted not only to increase the validity of the proposed variability handling approach, but to address the actual practitioners' needs in this area.

2.5.2 The reduction in number of selected studies

Thirteen studies ranging from 2005 to 2014 were selected in this review, a considerably lower number of selected studies comparing to the related reviews in SPLE area mentioned in Section 2.2. Review by (Alves et al., 2010) included 49 papers focusing on requirements engineering approaches for SPL and 89 studies were selected as primary studies when reviewing about domain analysis approaches in (Khurum & Gorschek, 2009). Benavides in (Benavides, Segura, & Ruiz-cort, 2009) included 53 studies when reviewing about automated feature modelling while the authors in (L. Chen & Ali Babar, 2011) selected 97 studies when reviewing about variability management approaches. Hence, the number of selected studies in our review is small when comparing to other related reviews (Alves et al., 2010), (Khurum & Gorschek, 2009), (Benavides et al., 2009), and (L. Chen & Ali Babar, 2011). This is because our focus is only towards feature extraction approach that deals with requirements from NL documents: a subset of RR topic in SPL. We disregard the studies on feature extractions from model-based mentioning RR, for example work in (Monzon, 2008), (Knethen et al., 2002), and (Robinson & Woo, 2004). We also excluded the studies regarding extractions of source code such as in (Marcus & Maletic, 2003) or selection of components for reuse in SPL (Abraham & Aguilar, 2007), or RR through pattern (Renault, Mendez-Bonilla, Franch, & Quer, 2009). A literature review section in another published paper related to RR provided only seven studies (Barreto et al., 2013). RR is a part of Requirement Engineering activity in SPL and RR also is truly one

University of Malaya

of the many activities within Domain Analysis (Neighbors, 1984). This justifies the reduction in the number of selected studies in our review, which is only 13, when comparing to other related reviews that focus on a bigger scope of research in SPL such as domain analysis, requirements engineering in SPL practices, or feature diagramming and modelling in SPL.

2.5.3 Threats to validity

The results of this SLR might have been affected by certain limitations such as inaccuracy in data extractions, bias in the selection of primary studies, and inaccuracy in assigning scoring to each study for the quality assessment criteria. To minimise the bias in data extraction and QA assessment, the second author selected about 20% of the selected studies and filled in the appropriate data collection forms. The accuracy of assigning scores to the selected studies on quality assessment criteria was very subjective. For example, some of the studies did not explicitly mention the strategy employed and required a very subjective judgement from the researchers. Any discrepancies found were discussed among the authors until a consensus is met. This SLR might have also missed out other feature extractions for reuse approaches that have been patented and commercialised but have not been published in literature, possibly due to privacy or copyright reasons. The issue of bias in study selection is addressed through multiphase search approaches (online database, snowballing, and manual search on targeted journals and conferences) that help to minimise the possibility of missing evidence.

Summary

RR if done systematically will increase the efficiency and productivity in SPLE. Although various approaches have been reported in this area, there was no attempt to

University of Malaya

systematically review and synthesise the evidence of how to extract features from NL requirements for reuse in SPLE. To fill this gap, a systematic literature review for feature extraction approaches from NL requirements for reuse in SPLE was conducted. Thirteen primary studies are selected resulting from searching the literature through three main phases: automated database search, complimentary citation-based search, and manual target search. The inclusion and exclusion criteria for selecting the primary studies were outlined, which were meant to answer the main SLR's research questions.

The main research questions were answered, and importantly the result is presented in this chapter. The main findings from this SLR include the following: i) SRS documents followed by product descriptions were found to be the most frequently used input for the feature extraction process, while most of the approaches derive the feature trees or models as output; ii) Most feature extraction processes are done in four primary phases: assessing requirements, terms extractions, feature identifications, and formation of feature model; iii) Although many approaches were well-documented in research publications and received high scores in the quality assessment conducted, none of the selected studies has explicitly produced any practitioners' guidelines and thus confirming the earlier observation of lack in practitioner's guidelines for SPLE (Alves et al., 2010) in which RR is an important aspect to be considered; iv) This SLR revealed that limited software metric approaches were used in conjunction with experiments and case studies as part of the evaluation procedures; and v) Not many studies produced automated support tools that are made available publicly.

The findings of this SLR are believed to be an important contribution to the practitioners and researchers as it provides them with useful information about the different aspects of RR approaches. For practitioners, this SLR has categorised the

University of Malaya

process for features extraction from NL requirements into phases with detailed information on what approaches are available for adoption in each phase, including some information on tools that are available from open sources. For researchers, the lower number of selected studies in this SLR indirectly indicates that a lot of research work need to be done in this area. The popular publication venues gathered from our searches can be useful information for those who want to further perform literature review on RR. Our observation in this study as well highlights the areas, which needed immediate attention for future collaboration between researchers and practitioners, mainly on who can use the proposals from academia. Moreover, the summary of domain information reported in this study may provide significant information to researchers and practitioners regarding the needs to extend the applicability of feature extraction approach to various other domains in SPLE.

University of Malaya

CHAPTER 3: RESEARCH DESIGN

Introduction

The first problem statement as specified in Chapter 1 stated that reusing requirements in natural language (textual) form can be arduous, complicated, time consuming, labor intensive, decrease productivity and exposed to the risk of errors if carried out manually. Secondly, the state of practice for this area is not known. Thirdly, there are missing guidelines, taxonomy or even process model for the overall process for feature extraction in requirements reuse to guide researchers or practitioners in this area. Although a number of proposals have been reported in literature, there are mostly publications resulting from research proposals that are not empirically validated and reported. Therefore, this research aims to investigate the requirements reuse problem and propose a suitable solution to it. This chapter provides the explanation mainly on how the research is conducted and what activities are involved in order to answer the research questions specified earlier in Chapter 1.

3.1 Formulating the Research Design

In scientific research, research design is defined as "a blueprint for conducting a study with maximum control over factors that may interfere with the validity of the findings" (Burns and Grove, 2003). A research design should typically consists of how data to be collected, what and how selected instruments to be used and how the data analysis will be conducted. Together with defining the design or methodology for this research, the strategies that constitute good research in software engineering as specified in (Shaw, 2002) is also considered. Shaw emphasised that a good research project should have benefit from a better understanding of the research strategies that have been most successful in terms of type of *Research Questions* being investigated, the *type of*

University of Malaya

Results produced and the *Criteria* used in validating the result. According to Shaw, research questions can be asking about method of developing, analysing, designing, evaluating or implementing specific software systems. The type of results produced by a research may be a specific procedure or technique in software development, or sometimes results can be the outcome of a specific problem or results from a particular analysis. Lastly, criteria for validating the results can be the actual use (experience) and systematic analysis. In the following subsections, these three important elements of good research mentioned by Shaw will be described in terms of the current research settings.

3.1.1 Type of Research Questions and Research Methods

Table 3.1 revisited the main research questions posted in Chapter 1. Additionally, this table specifies what are the type of research questions being used and its research methods based on what being characterised by (Shaw, 2002).

Table 3-1 Type of Research Questions

RQ#	Research Question	Type of research question	Research Method
RQ1	What approaches are available to extract features from natural language requirements in the context of requirements reuse?	Design, evaluation, or analysis of particular instance	Literature Review
RQ2	How does the existing approach being validated?	Design, evaluation, or analysis of particular instance	Literature Review
RQ3	What is the current state requirements reuse practice?	Design, evaluation, or analysis of particular instance	Explorative Survey
RQ4	Why requirements reuse is not that common among software practitioners	Design, evaluation, or analysis of particular instance	Explorative Survey
RQ5	What is the proposed feature extraction process and how to demonstrate the solution?	Method for development	Experiment

University of Malaya

RQ6	How is the proposed method being evaluated?	Analysis method	Validation

The first four research questions in Table 3.1 relate best to the third type of RQ as characterised by (Shaw, 2002), as "Design, evaluation, or analysis of particular instance". This type of question more specifically seek to answer questions such as "What is property X of method Y?" and "What is the current state of X / practice of Y?". In case of this research, RQ1 until RQ4 can be answered through conducting a thorough literature review or a survey, or even by using both methods. The RQ5 in this research seeks to answer the question of "What is the proposed feature extraction process and how to demonstrate the solution?" This RQ type relates to the "Method of development", which further explore on "How can we do/automate doing X?", in which the researcher have proposed a FENL process model and conducted an experiment in order to answer this RQ. The RQ6 asks "How to evaluate the proposed approach?" This RQ type relates best to the "Method for analysis" as specified by (Shaw, 2002), in which further explore on "How can one evaluate the quality/correctness of X?" To answer this question, the researcher conducted a series of evaluation towards the results obtained from the experiments (Chapter 6).

3.1.2 Type of Results Produced

Research yields new knowledge, and the types of knowledge produced by research are expressed in the form of a particular results (Shaw, 2002). From most common kind of paper published in ICSE, Shaw reported that most paper reports a new or better procedure or techniques for software development or analysis. In this thesis, the results or knowledge produced by conducting the systematic literature review and explorative survey includes a detailed synthesise of evidence that forms the structure or taxonomy for the requirements reuse problem area (a process model for feature extraction is

University of Malaya

presented in Chapter 2, as an implication of SLR conducted). Proposal for an improved feature extraction approach yields a defined procedure for solving the feature extraction problem. The process model and step-by-step procedure on implementing the feature extraction is specified in Chapter 5. This improved procedure exposes important design decision for the feature extraction in requirements reuse problem.

3.1.3 Criteria Used in the Evaluation

The most common kinds of evaluation reported in ICSE papers compiled in (Shaw, 2002) are experience in actual use and systematic analysis. In this thesis, the result from the experiment will be validated using the widely used metrics to evaluate the accuracy of : information retrieval that measures Recall, Precision and F-Measure (Cleverdon, 1970) and its time efficiency. Additionally, results from the accuracy evaluation will be validated for their significance using statistical method.

3.2 Research Phases and Research Activities

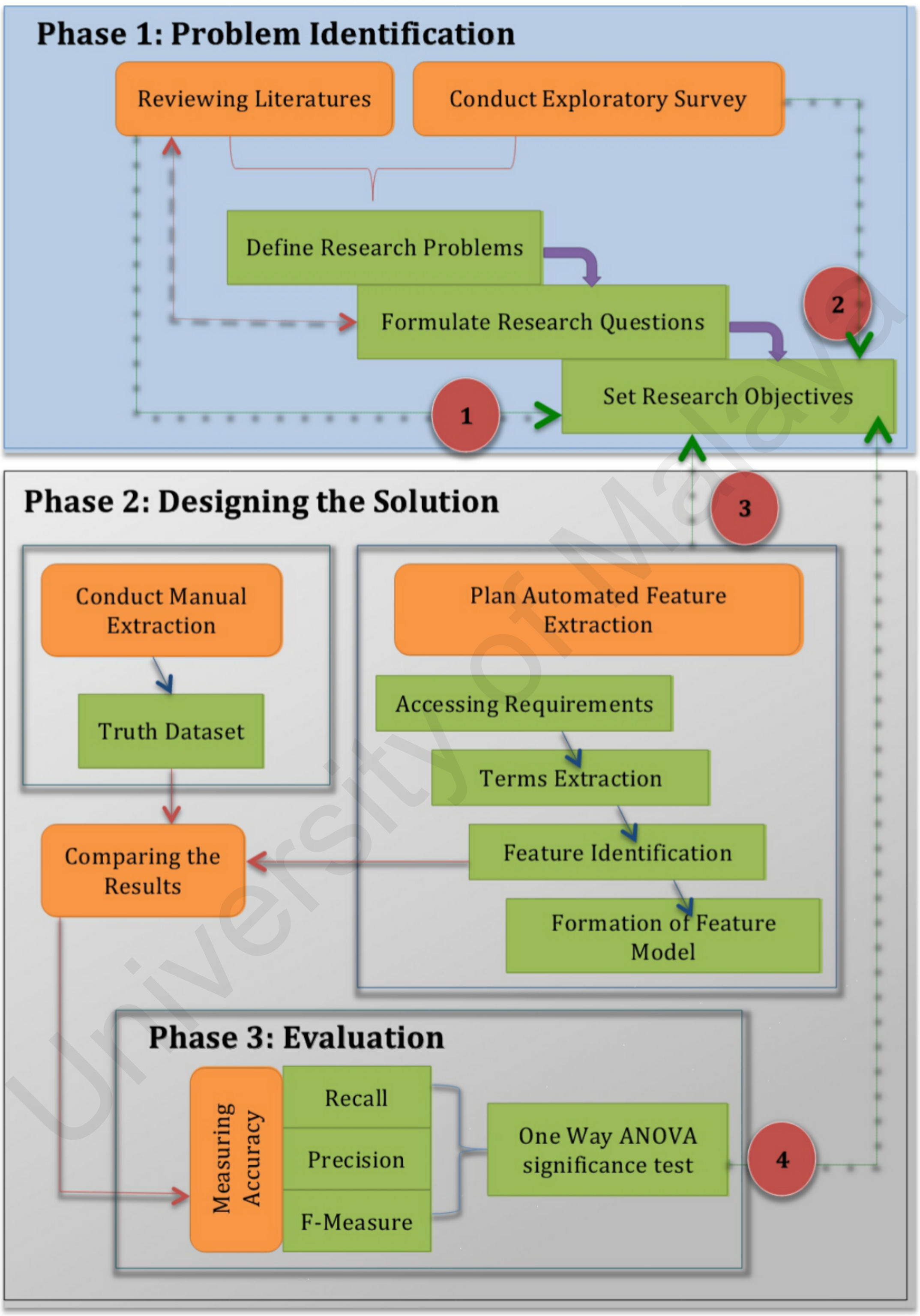
This research is divided into three phases: Problem Identification, Design of the Solution and the Evaluation. Figure 3.1 illustrates the Research Phases.

In Phase 1, there are two main activities involved: reviewing literature and conducting explorative survey. In Phase 2, the solution to the feature extraction for requirements reuse problem is formulated and experiment is conducted to demonstrate the propose solution. This also includes conducting the manual extraction process. Lastly, in Phase 3 the proposed solution is evaluated. Results from manual approach are compared with the automated process and the performance averages are tested for significance. The numbers circle in red in Figure 3.1 indicates the research objectives that are mapped to the activities in the research phases. Additionally, Table 3.2 outlined the research activities involved in order to satisfy the research questions, covering

University of Malaya

specific research objectives mentioned earlier in Chapter 1. Each of the research phases will be further explained in the following sections of this chapter.

University of Malaya



Phase 1: Problem Identification
Reviewing Literatures
Conduct Exploratory Survey
Define Research Problems
Formulate Research Questions
Set Research Objectives
1
2
3
4
Phase 2: Designing the Solution
Conduct Manual Extraction
Truth Dataset
Comparing the Results
Plan Automated Feature Extraction
Accessing Requirements
Terms Extraction
Feature Identification
Formation of Feature Model
Phase 3: Evaluation
Measuring Accuracy
Recall
Precision
F-Measure
One Way ANOVA significance test

University of Malaya

Figure 3.1 Research Phases

Table 3-2 Research Phases and Detail Research Activities

Phase	Method	Research Activities	Objectives Covered
Phase 1: Problems Identification	a. Literature Review	• Perform Systematic Literature Review in accordance to Kitchenham's guidelines for performing SLR in Software Engineering research (Kitchenham & Charters, 2007).	***Objective#1:*** To identify available approaches in feature extraction from natural language requirements for requirements reuse.*(Answers to RQ1-RQ2) – Chapter 2*
	b. Exploratory Survey	• Research on available literatures from the library online database pertaining to survey being developed in the area of software reuse/requirements reuse. • Select suitable survey elements for constructing Online Survey, to be distributed to software practitioners in the context of requirements reuse in Malaysia. • Distribute Online Survey to Software Practitioners in Malaysia. • Collect and analyse the survey responses.	***Objective#2:*** To explore the current state of practice for requirements reuse among software practitioners.*(Answers to RQ3 – RQ4) – Chapter 4*
Phase 2: Designing the Proposed Solution	a. Formulation of process model based on SLR and survey results	• Analyse the findings from Phase 1. • Propose a process model for feature extraction and clustering in the context of RR for SPLE.	***Objective#3:*** To propose a feature extraction process as the solution to requirements reuse problem *(Answers to RQ5) – Chapter 5*
	b. Experiments	• Select applicable approach found in Phase 2 for experiment in this phase. • Extract raw requirements from online software reviews available on the Internet • Perform text preprocessing and data cleaning using NLP techniques. • Use Latent Semantic Analysis, and clustering algorithm to identify similar requirement documents. • Use clustering algorithm to group common software features • Use LSA to trace extracted features	
Phase 3: Evaluation of the Proposed Approach	a. Evaluation of experiment results	• List out all evaluation methods used in related works. • Select suitable evaluation methods and evaluate results from Phase 2.	***Objective#4:*** To evaluate the proposed approach *(Answers to RQ6) – Chapter 6*

University of Malaya

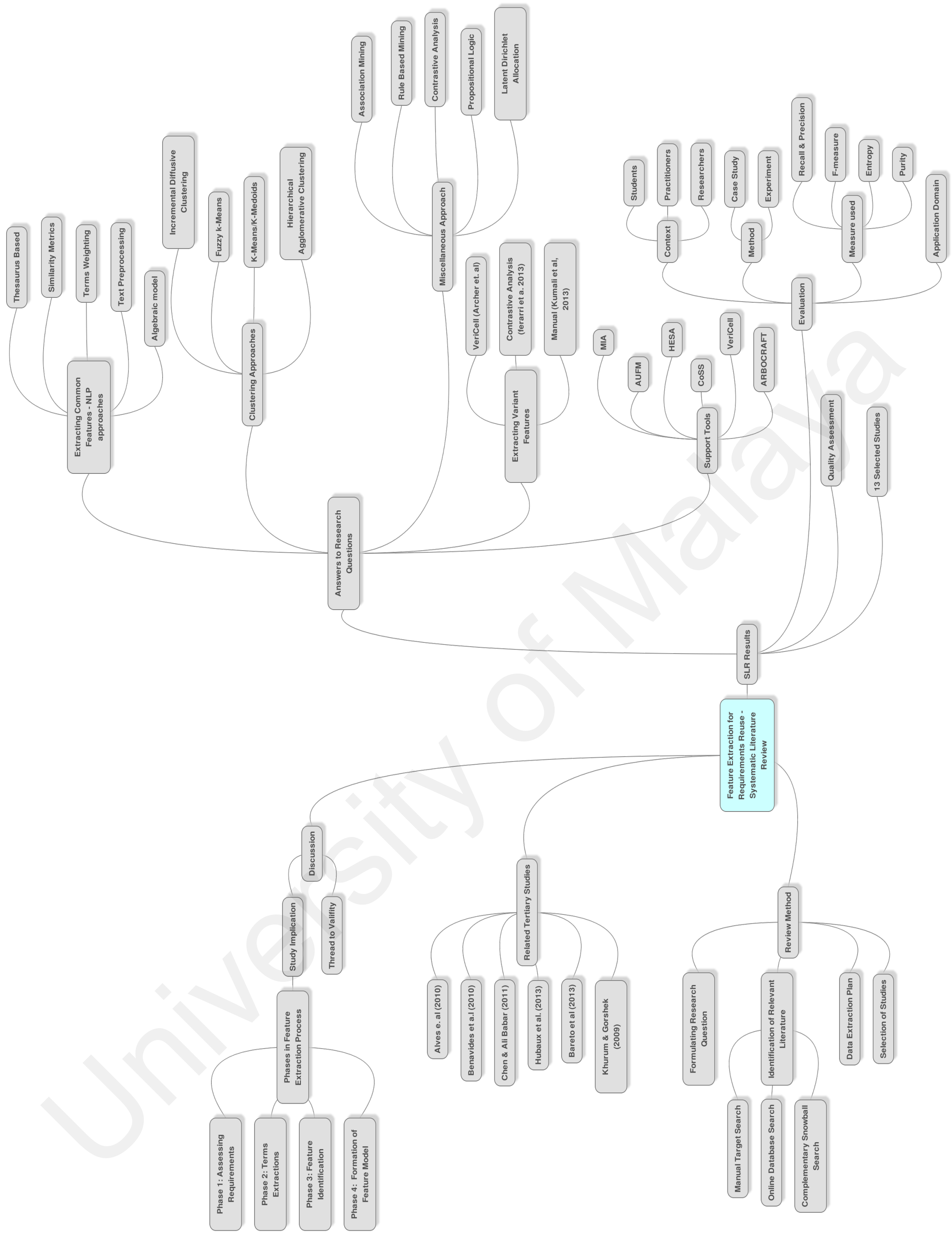
3.2.1 Phase 1: Gather Research Problems

Requirement Engineering is one of the most important phases for software development. The success of a software development relies most on the results produced by the Requirement Engineering phase. Simultaneously, SPLE methodology has proven to efficiently save the development time, in terms of reducing time to market, save the development costs and increase team productivity by leveraging the software reuse concepts. SPLE deals with reusing software artefacts namely the requirements, architectures, designs, test plans and more. Many works in SPLE has been devoted in terms of architectures and code reuse, but not many have focused on the reuse of textual requirements within the SPLE paradigm. To gather the research problems, the author started the literature search with two phrases from the above observations: "Software Product Lines" and "Requirements Reuse". The following subsection describes the process involves in obtaining the research problems: reviewing literatures and conducting a preliminary survey to confirm the research problems.

3.2.1.1 Conducting Systematic Reviewing Literature

The main aim of the conducted Systematic Literature Review is to find gap in existing literatures pertaining to the extraction of natural language requirements for reuse in the context of Software Product Lines. There are various approaches found from the existing literatures for the requirements reuse topic, however some were not selected because either the works did not fit within the context of Software Product Lines or the works did not deal with natural language requirements. A detail about inclusion and exclusion criteria for the Systematic Literature Review is presented in Chapter 2. To provide an overview for content of Literature Review conducted, a mind map in Figure 3.2 is prepared.

University of Malaya



University of Malaya

Figure 3.2 Mind Map for Systematic Literature Review

3.2.1.2 Conducting Exploratory Survey

An exploratory survey is conducted to investigate the state of problem for requirements reuse. Studying related surveys in the area helps in formulating the survey questions. Only one article in IEEE (Chernak, 2012) was found to have reported a survey pertaining to requirements reuse in general. Additionally, article on surveys related to software reuse in general were also referred, for example (Agresti, 2011), (Slyngstad et al., 2006), (Mellarkod, Appan, Jones, & Sherif, 2007) and (Frakes & Fox, 1995). The selected articles are thoroughly reviewed, analysed and questions that are related to the current research objectives are adopted in the exploratory survey conducted for this research. This exploratory survey is distributed to software practitioners through emails with Google docs survey link. The detail about the survey conducted is presented in Chapter 4.

3.2.2 Phase 2: Designing the Proposed Solution

Chapter 5 describes in detail the activities involves in Phase 2. Techniques from Natural language Processing and machine learning are used in the feature extraction process. Additionally, features extracted are grouped by using clustering algorithm. The process model is realised through set of laboratory experiments on raw data extracted from the selected software reviews that are available on the Internet. The implementation of feature extraction is made possible with Python 2.7 compiler, Matlab 2011R and Miscrosoft Excel.

3.2.3 Phase 3: Evaluation of the Proposed Approach

One of the main research problem specified earlier in this chapter is the lack of empirical validations among the related works published in this area. As a way to contribute to the software engineering community, validation towards the results produced by the proposed

University of Malaya

procedure is done by comparing the features obtained by the proposed approach as compared to the result extracted manually by human in terms of recall, precision and F-Measure (accuracy) and comparison of time taken by human versus the time taken by the auto-generated approach. Eight teachers (as the subject matter expert), and eleven software practitioners (as a professionals with technological background), are engaged to perform manual feature extractions. This manual extraction procedure was adopted from suggestion made in (Carreno & Windbladh, 2013). Adding to accuracy measure, statistical measure is used to determine the result significance, where One Way ANOVA test in SPSS is used. The details of the evaluation process together with its discussions are detailed out in Chapter 6.

Summary

This chapter describes the Research Design used in this study, and Figure 3.1 provided the earlier summarised phases of this research. Next, in Chapter 4 the exploratory survey is reported.

University of Malaya

CHAPTER 4: EXPLORATORY SURVEY ON THE STATE OF REQUIREMENTS REUSE

Introduction

Reuse of software artefacts such as requirements, architectures, designs, codes, and test plans can produce many benefits including reducing development costs, increasing developers' productivity, and expediting time to market. This is true especially when reuse is considered early during software development. Despite the known benefits of requirements reuse, the current state of practice is not known or well studied by the software engineering community. In this chapter, a survey conducted to explore the state of requirements reuse practice is reported.

This chapter is structured as follows: Section 4.1 discusses related surveys generally in the area of Software Reuse and Section 4.2 describes the design for exploratory survey conducted. Later, Section 4.3 discusses the findings from the survey and Section 4.4 presents the threats to validity for the conducted survey, and lastly a summary to the chapter will be provided.

4.1 Related Surveys

In 1995, an empirical study was conducted to investigate the software practitioners' attitudes, beliefs, and practices in reusing codes and other software development artefacts (Frakes & Fox, 1995). In this study, Frakes et al. (1995) conducted a survey to answer sixteen common questions about software reuse in organisations within the US and Europe. The participants include software engineers, managers, educators, and other software development and research communities. Important findings from their survey revealed factors promoting systematic reuse include education about reuse, developers' understanding of the economic

University of Malaya

benefits of reuse, instituting common development process, and making high-quality assets available to developers.

Slyngstad et al. investigated the developers' view on software reuse through a survey conducted at Norway's Oil and Gas company in 2006 (Slyngstad et al., 2006). The study collected responses from 16 software developers at Statoil ASA. The results showed that reuse benefits from the developers' view include lower costs, shorter development time, higher quality of reusable artefacts, and a standardised architecture. Component understanding was found to be sufficient; however, an improvement to documentation is needed. In addition, they have found that there is no relation between reuse and increased rework.

Mellakord et al. conducted a study on multilevel analysis of factors affecting software developers' intention to reuse software assets in general (Mellarkod et al., 2007). The survey was administered to 50 companies in India back in 2007. Technology Acceptance Model (TAM) from (Davis, Bagozzi, & Warshaw, 1989) was used in developing their conceptual research model. Results from (Mellarkod et al., 2007) revealed that technological-level (infrastructure) and individual-level (reuse-related experience and self-efficacy) were major determinants. In addition, the findings suggested that more investigation is needed on nontechnical factors (i.e. prevailing attitudes and perceptions) that are barriers to software reuse.

Work by (Agresti, 2011) investigated the developers' experiences and perceptions on software reuse in 2010. In this work, Agresti introduced the "4A" model which emphasised that for each organisation to obtain any benefits from code reuse, four conditions must be met: Availability, Awareness, Accessibility, and Acceptability. Agresti, in his study, was more

University of Malaya

specific where the investigation done focused on code reuse. The findings from (Agresti, 2011) revealed the greatest obstacle to reuse was shown to be awareness of reusable code and the developers' perceptions of its acceptability for use on their new projects. Interesting to note also, the developers felt that the complexity of old codes was the main reason why the codes were not reused.

In 2012, Chernak, Y., from Valley Forge Consulting reported a survey conducted pertaining to the state of requirements reuse practice (Chernak, 2012). The respondents came from the author's professional network across the globe. 82 responses (in which 60% of them resided in North America) were gathered during a six-month survey in 2010. Even though the respondents were aware of the reuse benefits, he found that poorly structured and badly maintained existing requirements were the main obstacles for adopting requirements reuse. He concluded that to improve reuse adoption, organisations should include refactoring existing requirements into a better structured model, maintaining a complete requirements model through releases, separating the stakeholder and product types, and imposing change impact analysis in their reuse practice.

The first three related works (Frakes & Fox, 1995), (Slyngstad et al., 2006) and (Mellarkod et al., 2007)focused on software reuse in general and the fourth one (Agresti, 2011) focused on code reuse. To summarise, Figure 4.1 illustrates the important findings from the related works used in this section pertaining to the factors that can influence reuse adoption among practitioners.

University of Malaya

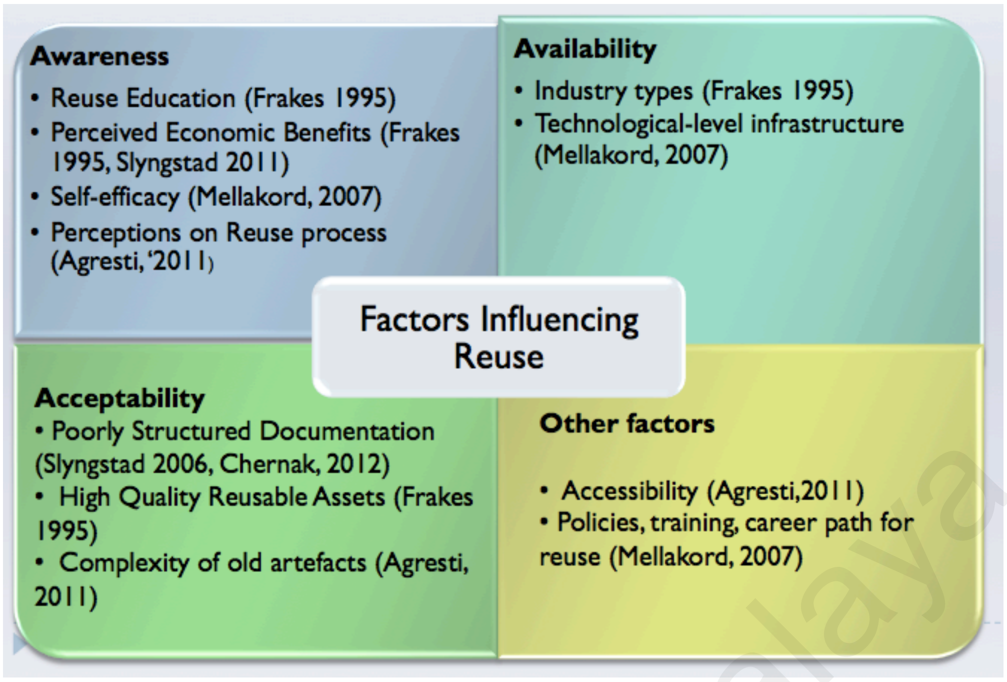

Awareness
• Reuse Education (Frakes 1995)
• Perceived Economic Benefits (Frakes 1995, Slyngstad 2011)
• Self-efficacy (Mellakord, 2007)
• Perceptions on Reuse process (Agresti, '2011)
Availability
• Industry types (Frakes 1995)
• Technological-level infrastructure (Mellakord, 2007)
Factors Influencing Reuse
Acceptability
• Poorly Structured Documentation (Slyngstad 2006, Chernak, 2012)
• High Quality Reusable Assets (Frakes 1995)
• Complexity of old artefacts (Agresti, 2011)
Other factors
• Accessibility (Agresti,2011)
• Policies, training, career path for reuse (Mellakord, 2007)

Figure 4.1 Important findings from related surveys on factors influencing software reuse

University of Malaya

The survey presented in this chapter is very similar to the one conducted earlier in (Chernak, 2012). However, the author tried to adopt some of the important items imposed in (Mellarkod et al., 2007) and (Agresti, 2011) in this survey and place the context for understanding the practitioners' perceptions and experience in requirements reuse.

4.2 The Design of Explorative Survey

One of the research objectives for this thesis is to investigate the current state of practice of requirements reuse. This objective can be achieved by examining the current state of practice for requirements reuse through survey, as what will be presented in this chapter. The aim of this survey comes in twofold: 1) to provide a brief overview on the requirements reuse practice among software practitioners; and 2) to explore the common obstacles for adopting requirements reuse from software developer's perspective. Hence, the following are two research questions to be answered in this survey:

- RQ1: "To what extent have the RR been in practice?"
- RQ2: "What are the factors that might hinder the RR practice?"

4.2.1 The survey construction

The Chernak's survey was administered back in 2010, with the main findings stating on the challenges on requirements reuse adoption. Inspired by the findings, this survey will use the reason found in Chernak's survey and mapped them to the 4A's factors used in the survey from (Agresti, 2011) . Table 4.1 indicates the 4A factors from (Agresti, 2011) and how the factors are adapted to the survey in this research as a guide on finding the answers to the research questions.

Table 4-1 4A Factors Adaptation

4A Factors	Used in Agresti (Agresti, 2011)	Adaptation
Availability	Reusable artefacts	1. Support tools
Awareness	The existence of reusable artefacts	1. Self-efficacy 2. Reuse benefits
Accessibility	Ability to get the reusable artefacts	Ability to get the reusable requirements
Acceptability	Agreements on accepting the reusable artefacts in new projects	1. Conditions of the existing requirements. 2. Who decide to accept

There were a total of 40 questions in the questionnaire. Part A comprises 12 questions related to demographic background of the respondents, Part B consists of 17 questions related to requirements reuse perceptions and experiences and Part C consists of 11 questions related

University of Malaya

to general issues in requirements reuse. Survey questions were adopted from the three related surveys (Mellarkod et al., 2007), (Chernak, 2012) and arranged according to the 4As factors specified in (Agresti, 2011). The survey questions firstly were sent out for pilot testing to check for reliability.

Demographic questions in Part A investigate the background of the survey respondents including:

- Position in current job
- Number of years in Requirements Engineering
- Industry group that describes their organisations
- Size of development team
- Requirements format used

Table 4.2 indicates the survey items used in Part B, which were related to the 4A factors presented earlier in Table 4.1.

Table 4-2 Survey Questions in Part B

4A Factor	Survey Item (Rating 1 to 7)
Behavioural Intention	I intend to increase my use of reusable requirements in the future development of application.
Availability	My organisation has appropriate support tools for: • developing reusable assets • managing reusable assets
Awareness	I feel reusing requirements requires a lot of mental effort (self-efficacy).
Awareness	It is easier for me to understand existing requirement documents as compared to developing new requirements (reuse versus develop new).
Awareness	Reusing existing requirements: • improve my job performance • improve my team productivity • decrease software maintenance costs

University of Malaya

Accessibility	1. Assuming I have access to existing requirements, I intend to use them when developing future applications. 2. Given that I have access to existing requirements, I predict that I would make use of them in developing future applications.
Acceptability	1. It is impossible to reuse the existing requirements because the existing requirements developed in previous releases are incomplete or do not exist. 2. It is difficult to identify which requirements can be reused because the existing requirements are poorly structured. 3. It is difficult to use the existing requirements because the existing requirements are not kept updated.

Additional question imposed in the survey but are not related to 4As factors includes a question asking whether there is anyone who reuse requirements in the latest project. If they reuse, what are the reasons for them to reuse. Is it because reuse:

- is systematically planned (SPL); or
- just happens because the new project is very similar to the one completed before (ad hoc reuse); or
- occurs because of maintaining previous release (software maintenance).

For each item in Part B, the Likert-Scale 1 to 7 response options was used. Table 4.3 indicates the score rating.

University of Malaya

Table 4-3 Likert Scales Rating

1	Strongly Disagree	Negative attitude (Disagree)
2	Disagree	
3	Slightly Disagree	
4	Neutral	Undecided
5	Slightly Agree	Positive attitude (Agree)
6	Agree	
7	Strongly Agree	

In the analysis, scores 1 to 3 indicate negative attitude (disagreement) while scores 5 to 7 indicate the respondents' positive attitude (agreement) to items in the questionnaires. The higher the score indicates, the more positive attitude (agreement) towards the item imposed.

4.2.2 Pilot Testing

For pilot testing purposes, the survey was distributed to the Software Engineering experts at 6 public universities in Malaysia that offer Software Engineering programmes – fourty one sets of questionnaire were sent out and thirty six were returned. These experts were chosen because the pilot survey is aiming to get fast feedback confirming the readability of the questions. Each of the survey questions in Part B was tested in SPSS for internal consistency check by using Cronbach's α before being sent to the actual survey participants. Only two items found to use negative words. Hence, the wordings were changed, (for example, the word "decrease" was changed to "increase" and "is not important" was changed to "important") and the scores were reversed (i.e. from scale 1 to 7, the original score 6 was reversed to 2 and vice versa). The new scores were plugged in and retested in SPSS. As a result, the Cronbach's alpha is improved to 0.711, a more reliable value (Shull, Singer, J, & Sjoberg, 2008).

University of Malaya

4.2.3 Actual Survey

This survey is targeted to get responses from personnel who have experience dealing with requirement documents in software development. The respondents consist of Software Engineers, Project Managers, Requirements Managers, Educators and Authors in the Software Engineering areas, which were grouped in this survey as software practitioners. Although random selection is desirable, this cannot be obtained. This is because no statistics were made available by MDec regarding the number of software practitioners in Malaysia that dealt with requirement documents during software development. Thus, a snowball sampling technique was used as the process to gather survey responses.

4.3 Results And Discussions

Survey invitations through email were sent out to 48 contacts. These 48 contacts were identified from the researcher's personal contacts who worked in the software development environment. In addition, a link to the web-based survey questionnaire was posted on IT Professionals in Malaysia group on LinkedIn page, Malaysian Software Engineering Interest Group (MySeIG) page and Malaysian Research and Education Network (MyREN) page. From the period of April 2013 until September 2013, 41 responses were collected from the actual survey. Basic quantitative data analysis was done in Microsoft Excel and Frequencies Analysis in SPSS.

4.3.1 Demographic Information

In this section, the summary for the demographic information of survey respondents is reported. The majority, 22 respondents (53.7%) are Software Engineers. Other respondents held various posts including Researchers and Educators in Software Engineering (12.19%),

University of Malaya

Project Managers (7.3%), System Analysts (4.9%), Technical Specialists (4.9%), Requirements Manager (2.4%), and Software Tester (2.4%).

When asking about experience in requirements engineering, more than half of the respondents have more than 1 year experience in Requirements Engineering as shown in Figure 4.2.

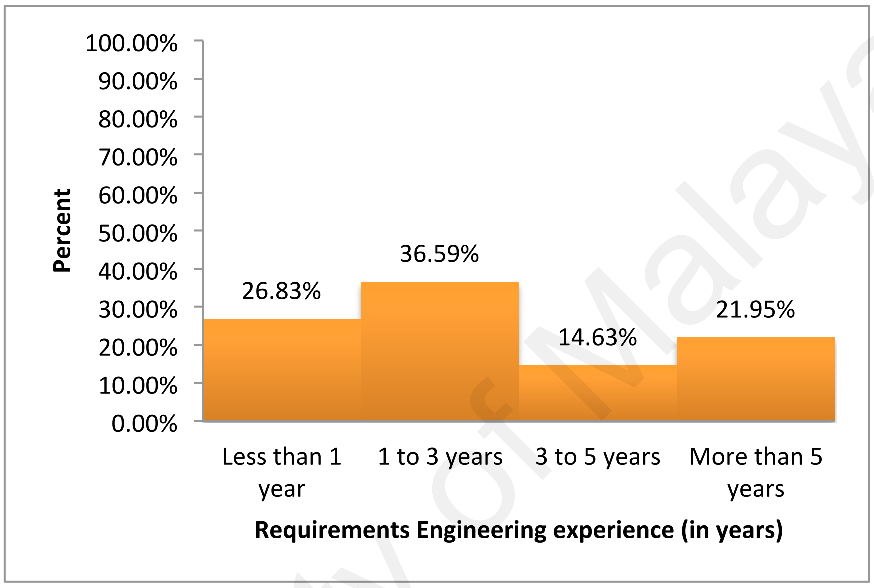


Figure 4.2 Requirements Engineering Experience

University of Malaya

Respondents came from various industries (Fig. 4.3), in which 36% came from Software Development House, 26% from IT Consultancy and 15% from Education, Research & Development category.

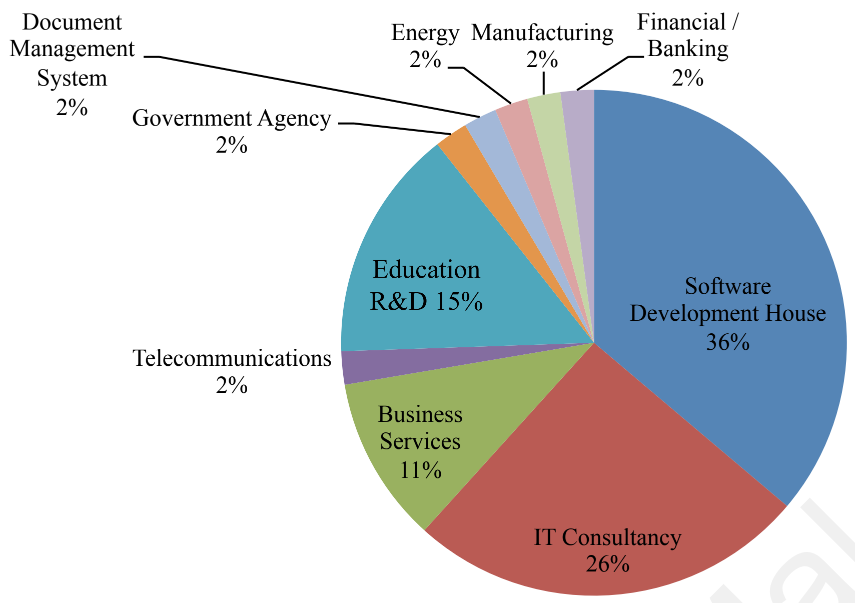


Figure 4.3 Profile of Survey Respondents

Table 4.4 indicates that 20 of the respondents (48.78%) worked in small development teams (between 1 to 5 people) and the remaining worked in various development team sizes.

Table 4-4 Size of development teams

Team size	Frequency	Percent
1 to 5 people	20	48.78%
6 to 10 people	7	17.07%
11 to 20 people	8	19.51%
21 to 50 people	6	14.63%
Total	41	100.00%

Respondents were asked to categorise the requirements format used in the software development they were involved with. Requirements in the form of features (63.3%) and

University of Malaya

textual (63.89%) were among the famously used form of requirements (See Fig. 4.4). Based on sample data gathered, natural language requirement is popular among the software practitioners participated in the survey. This trend is similar to the findings by Neill and LaPlante on the state of requirements engineering practice in (Neill & Laplante, 2003), where developers mostly used requirements presented in natural language during software development. This is because software requirement requires human interpretations, thus making natural language requirements more popular or commonly used (Oliveira, Alencar, & Cowan, 2011). However, not that big different is reported in between the adoption of use cases and textual form of requirements as gathered from this survey. Additionally, user stories and other requirements formats are reported the least to be used (5.6% - 8.3%).

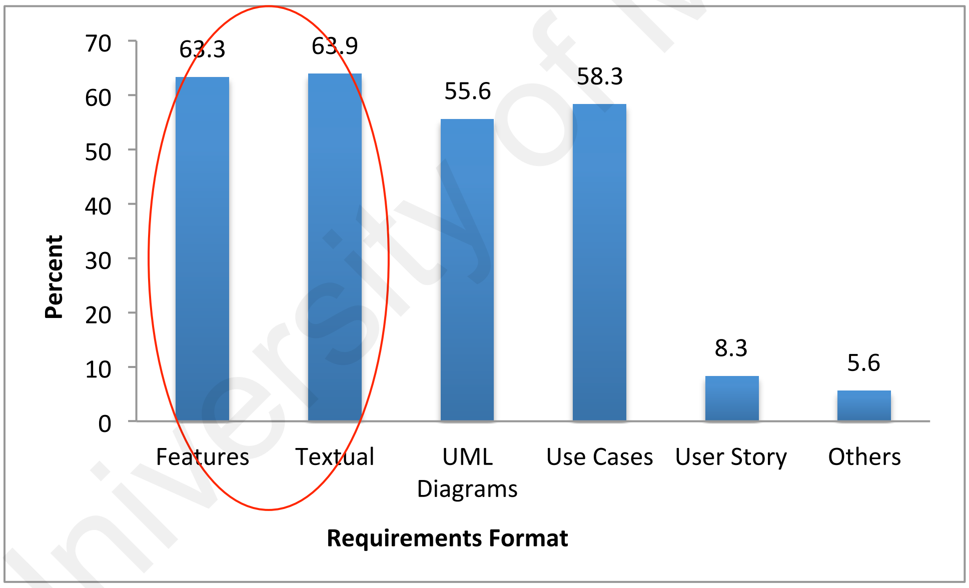


Figure 4.4 Requirements Format Used by Respondents

**Note that respondents may choose more than one category, thus results to reach more than 100%.*

University of Malaya

4.3.2 Perceptions and Experience in Requirements Reuse

This section provides results from the responses gathered on the practitioner's intention to reuse the requirements with each of the "4A" factors used in Table 4.2. Data collected were re-coded in terms of agreement (Likert scores 5 to 7) and disagreement (Likert 1 to 3) as mentioned earlier in this chapter. Throughout this section, undecided responses were discarded from the analysis (number of items discarded will be presented at each section as and when applicable).

4.3.2.1 Practitioner's Intention

Firstly, the survey seeks for information regarding the software practitioner's intention towards requirements reuse. Results gathered indicate that 25 out of 36 respondents have the intention to reuse requirements in the future development (see in Table 4.5). Out of 41 responses, with 5 undecided responses are discarded.

Table 4-5 Intention Towards Requirements Reuse

		Frequency	Percent
Valid	Agree	25	69.4
	Disagree	11	30.6
	Total	36	100.0

4.3.2.2 Availability

The first 4A factor in determining software practitioners' intentions to reuse is the availability of requirements reuse tools. For example, tools associated with managing and reusing requirements available in the current market are produced by BigLever, PureSystems, JAMA software, The Reuse Company, and some software extensions to DOORS by IBM.

University of Malaya

Most of the software mentioned are available for purchase, but not available for free download. The availability of support tools (fully automated or semi-automated) may reduce the burden put on the requirements analysts while identifying the core and variant features for reuse in new product family development (Weston et al., 2009)(Kumaki et al., 2012). The availability of automatic tool support can also offer an order-of-magnitude savings over manual feature extraction for reuse (Niu et al., 2013), and obviously increase productivity when reuse is done systematically (Barreto et al., 2013).

There were two survey questions pertaining to the availability of support tools: the first question seeks respondents who agreed that their organisation provided support tools for developing reusable requirements, while the second seeks respondents who agreed that their organisation provided support tools for managing reusable requirements. For the first question, 54.2% of the respondents reported that their organisation did not provide support tools for developing reusable requirements. For the second question, 53.8% of the respondents reported that their organisation do not provide support tools for managing reusable requirements. This result indicates that more than 50% of the respondents reported that no support tools are provided by their organisation to aid RR activities (developing and managing).

Although support tools are important, this survey data indicated that most organisations did not provide the tools in the RR activities. In conjunction to this, data collected also showed that most RR practitioners did not use any support tools during their last RR project: 27 out of 33 respondents (81.8%) did not use any support tool while requirements were reused in their latest project. Organisations did not provide support tools for RR, thus most of the practitioners did not use any tools in RR activities. This observation is suspected to have a relationship with the

University of Malaya

ad hoc RR practice. Therefore, a further analysis is done - the crosstab analysis was conducted to determine the pattern between respondents who did not use support tools in their latest RR project and compared it with the reason why requirements are reused in their latest project (SPL, ad hoc or software maintenance). As suspected, 16/27 of the respondents who did not use support tools were actually practicing RR on ad hoc basis, 9/27 did not use support tools and reuse requirements for maintenance purposes, and 2/27 who did not use support tools were involved in systematic RR (SPL). In addition, only 4/6 practitioners who practiced SPL use support tools in their latest RR project. This indicates that support tools were only used by many of the respondents who practised systematic RR (SPL), whereas those who were not involved in SPL did not use any support tools to aid their RR activities. SPL development have systematic way of reusing software artefacts, as for managing requirements various tools exists to support the SPL development, i.e commercial tools to support SPL development provided by BigLever, PureSystems or JAMA software. Table 4.6 details out the crosstab analysis.

Table 4-6 Crosstabulation - Reason for Reuse versus Using Support Tools in RR

Requirements are reused in latest project because:			Total
SPL	Just-happen (ad hoc)	Maintaining prior release	
2	16	9	27 (not using support tools)
4	2	0	6 (using support tools)
6	18	9	33
**8 undecided responses was removed from the original 41 responses, totaling up to only 33 responses counted for this item.*			

University of Malaya

In relation to this, three comments received in the open-ended section suggested that some of the practitioners still need to see a tool or framework for RR. According to the comments received, the closest RR tools they have seen is the UML diagrams, but not specific tools that are capable to search and select existing textual-based requirements for reuse in new software development. Based on the sample data collected, these findings provide information stating that support tools are needed to induce require reuse practice.

4.3.2.3 Awareness

Requirements reuse practice can be influenced by the practitioners' awareness. Factors such as awareness on self-efficacy, awareness on easiness to reuse versus develop new requirement, and awareness on the impact of RR towards improving job performance, team productivity and decreasing software maintenance costs. Fig. 4.5 below indicates the results pertaining to the awareness factors based on the data collected in this survey.

University of Malaya

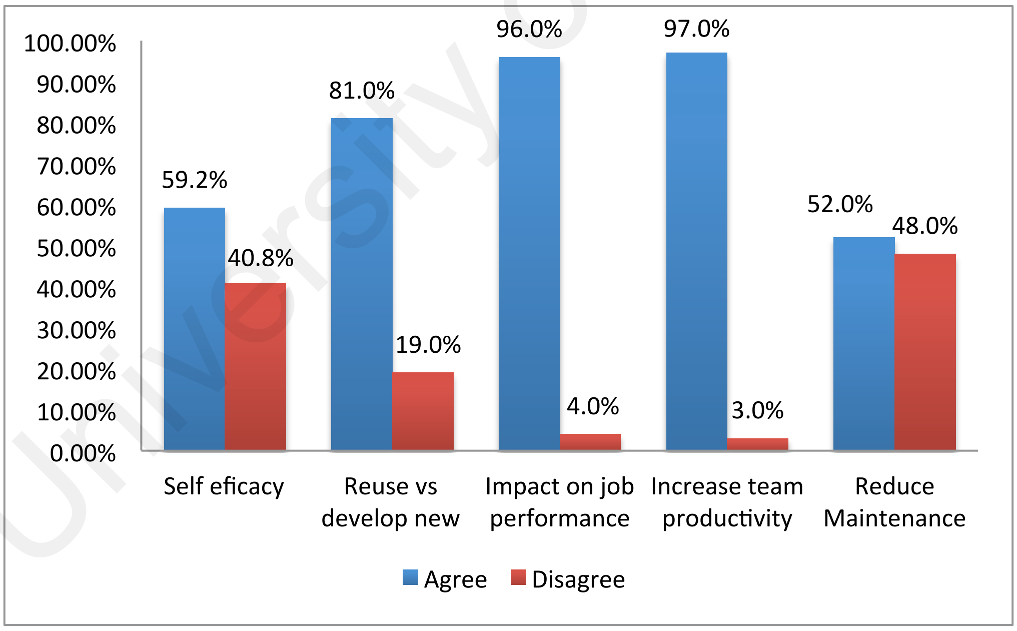


Figure 4.5 Awareness factors

When answering the awareness regarding self-efficacy, 16 out of 27 or 59.2% of the respondents agreed that RR requires a lot of mental effort. Although the respondents were aware of the difficulties to reuse, 81% agreed that it is easier to understand the reusable requirements as compared to developing new requirement documents. The findings indicated that software practitioners who participated in this survey were aware that to reuse is easier than to develop; however, reusing existing requirements will still need careful and rigorous thinking.

Regarding the awareness of the impact of RR, the majority of survey respondents agreed that RR provides good impact on their job performance and organisation. 26 out of 27 (96%) of the respondents perceived that reuse can give positive impact on their job performance, 29 out of 30 (97%) agreed that RR increases their team productivity, and 15 out of 29 (52%) agreed that RR may reduce the maintenance costs at the later stage of software development.

4.3.2.4 Accessibility

The next survey item seeks to answer whether RR practice is related to easy accessibility to reusable requirements. This was asked in two survey questions (refer to Table 4.2 for the two questions under Accessibility). The results are presented in Table 4.7.

University of Malaya

Table 4-7 Accessibility factors in RR

		Acc1*	Acc2**
N	Valid	36	30
	Missing	0	0
Median		1.00	1.00
Mode		1(29)	1(29)

**Acc1*: Assuming I had access to reusable requirements, I intend to use them when developing future applications

***Acc2*: Given that I have access to reusable requirements, I predict that I would make use of them when developing future applications

From Table 4.7, mode 1 indicated that the respondents agreed to the statements in *Acc1* and *Acc2*. Twenty nine out of 31 intend to reuse and 29 out of 30 predict to reuse requirements if they have access to reusable requirements. This is reflected in the open-ended section as well, where the practitioners tend to refer back to existing documentation (functionality and templates) when developing requirements for new releases.

4.3.2.5 Acceptability

The next survey question deals with acceptability factor. In a previous research conducted by (Solemon, Shahibudin, & Abdul Ghani, 2008) on requirements engineering problems in 63 software companies in Malaysia, the authors found out more than 70% of their respondents experienced problems related to requirements-process. These include inconsistent or changing requirements and incomplete requirements. With that in mind, RR is not being widely practiced could be due to the conditions of existing requirements produced from RE activities, namely reusable requirements are incomplete, poorly structured (inconsistent) or do not exist. In this survey, respondents were asked to respond to the three reasons why RR was not practiced in their organisations. Fig. 4.6 summarises the collected responses.

University of Malaya

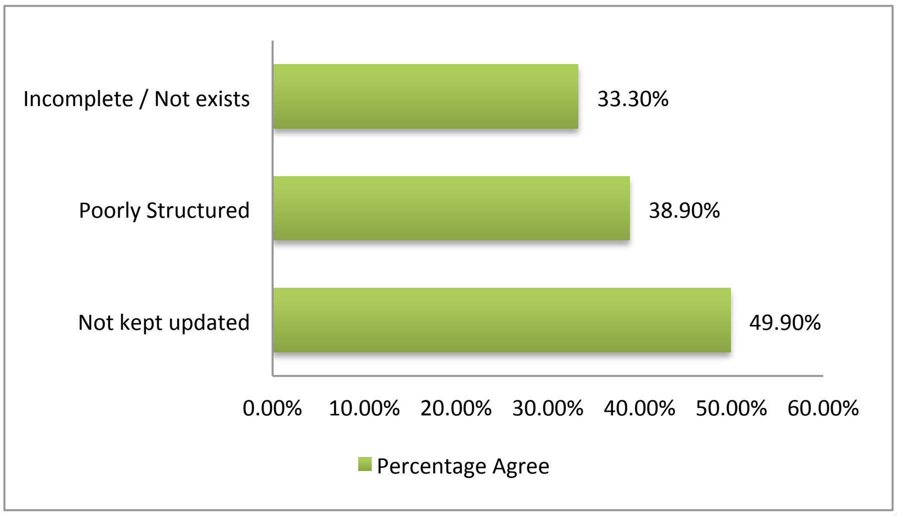


Figure 4.6 Conditions of reusable requirements

About 33.3% of the respondents agreed that the requirements developed in previous releases were incomplete (or did not exist), so it is impossible to reuse them. Moreover, the results revealed that 38.9% of the survey respondents agreed that existing requirements were poorly structured, and lastly 49.9% of the respondents believed that the existing requirements were not kept updated.

Although most respondents have the intentions to practice RR in the future, the three reasons in the acceptability factors hinder RR to happen. When mapping the behavioural intention to acceptability factors, 8 out of 25 who intended to reuse requirements in the future development reported that old requirements did not exist in their organisations. In addition, 9 out of 25 reported to have poorly structured requirements and 13 out of 25 thought that old requirements were not kept updated.

Therefore, although the intention to reuse exists, the conditions of reusable requirements (do not exist, not updated, and poorly structured) contribute to the reasons why RR is not widely practiced from the data collected in this survey.

University of Malaya

4.3.2.6 Additional Item

Additional items imposed in Section B include queries regarding the reasons why requirements were reused in the respondents' latest project as captured in Table 4.8. Only 19.4% of the respondents were involved in Systematic Reuse (SPL), while the majority of respondents (52.8%), reuse requirements in an ad hoc manner.

Table 4-8 Reasons requirements were reuse in latest project

Reasons:	Frequency	Percent
We are involved in SPL	7	19.4
Just happen the new project has similar requirements with the previously developed (ad hoc)	19	52.8
We are maintaining prior releases	10	27.8
Total	36	100

Software requirements gathered in ad hoc manner for reuse, for example by copying and pasting from the old requirements documents. This ad hoc process is unplanned, and most likely is done manually without tools. These activities in fact are time consuming and very prone to human error.

4.3.3 General Comments on RR (Open-Ended Question)

A section for the respondents to write their general comments on RR practice is provided at the end of the survey. The comments gathered are classified into the "4A" category as depicted in Table 4.9. Note that the general comments collected are only related to 3 out of 4 factors from the "4A" category, whereby there is no general comment gathered that is related to Accessibility factor.

University of Malaya

Table 4-9 Mapping General Comments to 4A Factors

4A Factors	**General Comments (Open-Ended)**
Availability	*"Software engineering community has yet to see any RR tools or framework"*
Availability	*"Currently, my organisation reuses 50% of older requirements. We make them as template for new development."*
Availability	*"Need for an industry standard for exchanging and sharing requirements in repository."*
Awareness	*"It is very important to educate developers on RR."*
Awareness	*"Older requirements need to be revalidated prior to reuse and thus RR will not necessarily increase productivity."*
Awareness	*"It is a good idea to use older requirements as it can help junior personnel involved in RE activities to learn."*
Acceptability	*"RR in my organisation is a case-by-case basis. Only those experienced will influence the decision-making of whether to reuse or not."*

There were two comments that are related to the first 4A factor, Availability: to enable reuse, there is a need to have the RR tools, framework or the industry standard for exchanging and sharing requirements in repository. Tools will help to expedite reuse, while industry standard requirements repository will enhance reuse practice as practitioners can have a variety of requirements from a broad range of domain to choose from. Thus, time spent on RE activities can possibly be reduced. The second important factor in 4A is Awareness. From the general comments section, comments that are captured include remarks that are related to the awareness of the RR: the importance to educate developers with RR and practitioners with RR knowledge can educate others in the development team. The last comments mentioned that

University of Malaya

only experienced personnel can make the decision whether to accept older requirements to be reused or not, in which reuse decision is as well related to level of authority in organisation.

4.4 Threats to Validity

One major threat to validity for this survey is pertaining to the method to reach the respondents. Using snowball sampling can introduce some threats to validity. This is because the survey link can be passed on to almost everyone and this is beyond my control whether the respondents are the actual software practitioners who deal with requirement documents in their job.

At the time this thesis is written, the result from online survey captured only 41 responses. The reliability of the survey results can be improved if more responses are collected, thus a more rigorous statistical evaluation can be performed. In this chapter, only frequency analysis (median, mode, and manual cross-tabulation comparison) is performed against the data collected. Additionally, it is highly noted that this amount of respondents can not be used to generalize to represent the whole community practicing software reuse in Malaysia. However, the information gathered from this survey can provide an overview on the state of practice in RR among the software practitioners.

Summary

In this chapter, a survey have been conducted to explore regarding to what extent the RR have been in practice, as for current data, respondents are from Malaysia. The survey explores seven factors that can influenced the RR practice: behavioral intention, availability of support tools, awareness factors (self-efficacy, easiness to reuse versus developing new requirements and impact of RR), accessibility to reusable requirements, and the acceptability conditions of

University of Malaya

reusable requirements. Additionally this survey indicates that the RR practice is not widely practiced in Malaysia mainly due to three impediments: unavailability of RR tools, unacceptable conditions of requirements to be reused, and the lack of RR education or guidelines provided.

This chapter explores the current practice of requirements reuse among practitioners. Next, in Chapter 5, a proposal for a semi-automated process to aid requirements reuse will be presented.

University of Malaya

CHAPTER 5: IMPLEMENTATION OF FENL

Introduction

Based on the findings from Systematic Literature Review, it was found that (1) most related studies use Software Requirements Specifications (SRS) as inputs, but product descriptions, brochures, and user comments are also used due to practical reasons; (2) the outputs from feature extraction process are commonly illustrated feature diagrams, clustered requirements, keywords or direct objects; and (3) the extraction process can be divided into four phases: requirements assessment, terms extraction, features identification, and feature model formation. Additionally, among important findings gathered from the Preliminary Survey in Chapter 4 reveals that the main impediments to RR practice include the unavailability of support tools or guidelines for requirements reuse process. Motivated by the above-mentioned findings, this chapter presents the implementation of semi-automated process for feature extraction from requirements in natural language for reuse in SPLE. This implementation is demonstrated through experiments.

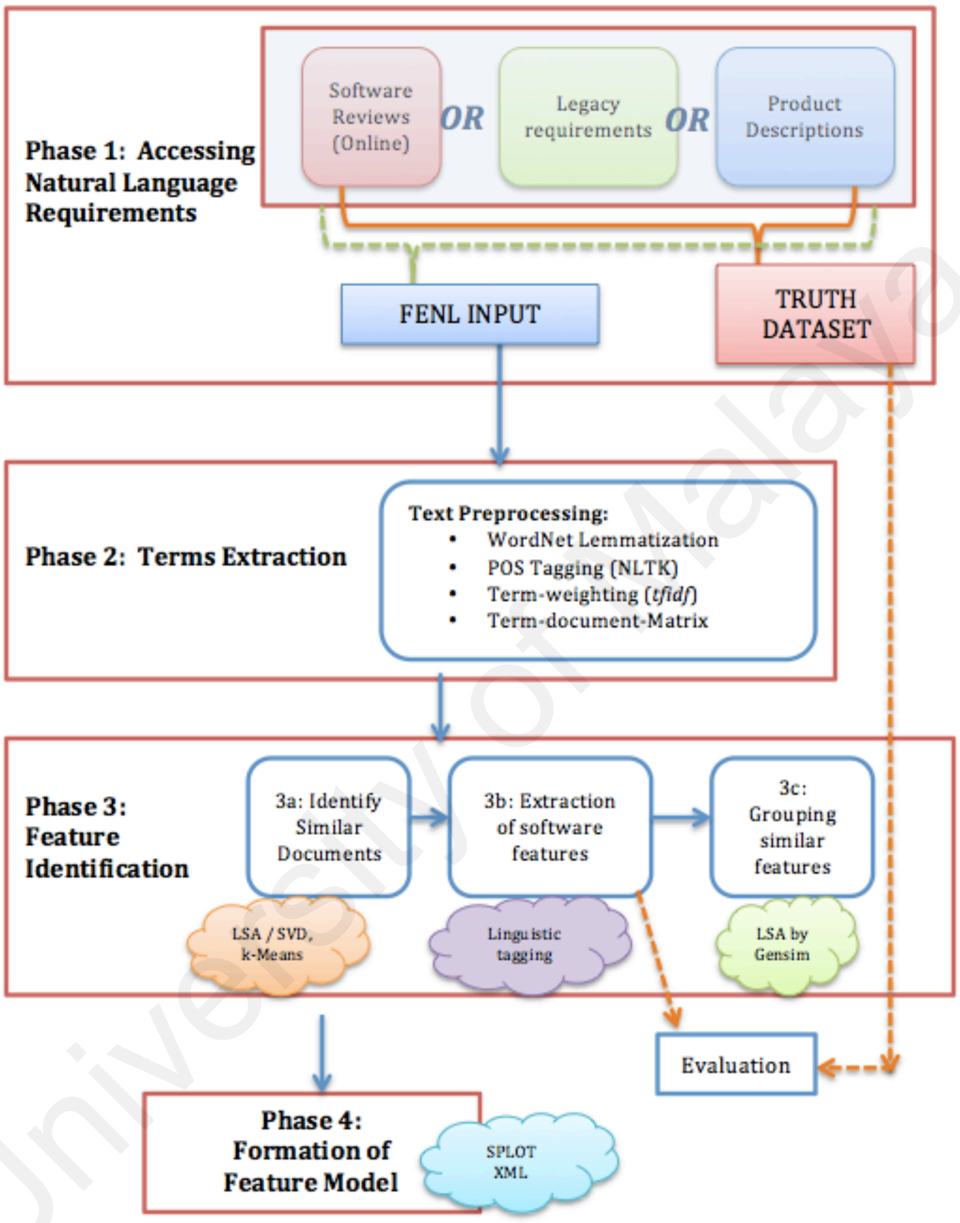
Section 5.1 firstly describes the process model for FENL, with some descriptions about the data sets used in the experiment. Section 5.2 until Section 5.5 detail out the four phases involves in the FENL, and finally Section 5.6 concludes this chapter.

5.1 Process Model

FENL is separated into four main phases, with the first three phases to be automated. The FENL offers to extract software features from various forms of requirements, such as online software reviews, legacy requirements or product descriptions by using NL processing, and IR techniques. However, for the experiment conducted in this research, only the freely available

University of Malaya

software reviews from the Internet are used. The overall process for FENL is illustrated in Figure 5.1.



Phase 1: Accessing Natural Language Requirements
Software Reviews (Online)
OR
Legacy requirements
OR
Product Descriptions
FENL INPUT
TRUTH DATASET
Phase 2: Terms Extraction
Text Preprocessing:
• WordNet Lemmatization
• POS Tagging (NLTK)
• Term-weighting (tfidf)
• Term-document-Matrix
Phase 3: Feature Identification
3a: Identify Similar Documents
3b: Extraction of software features
3c: Grouping similar features
LSA / SVD, k-Means
Linguistic tagging
LSA by Gensim
Evaluation
Phase 4: Formation of Feature Model
SPLOT XML

Figure 5.1 Overall process of FENL

University of Malaya

Before going through the FENL process in detail, a description of the data sets used for this research will be provided.

5.1.1 Data Set for the experiment

Due to confidentiality of most Software Requirements Specifications, SRS, software reviews that are available on the Internet are selected for use as the input to demonstrate the FENL approach. Related works in this area used various forms of requirements that include product descriptions, brochures, use cases, and the most recent ones used the user comments available for the mobile applications. In the case of this research, the software reviews compiled by experts that are available on the Internet are opted due to difficulty of accessing SRS documentation.

When looking at recent works such as (Carreno & Windbladh, 2013), (Iacob & Harrison, 2013), and (Guzman & Maalej, 2014), it is noted that the authors have used the first-hand user comments i.e. the comments left by mobile app users. The data used in their works are raw, unprocessed and usually contains sentiments and user complaints. In order to obtain reliable data, the compiled expert reviews are employed as the input to FENL process. There are some characteristic comparisons in between user feedbacks or comments as opposed to the compilation of expert reviews, as indicated in Table 5.1:

University of Malaya

Table 5-1 Characteristic Comparisons for User Comments as opposed to Expert Reviews for Software Products that can be Extracted from Publicly Available Sources

Characteristics	User Comments/Feedback	Expert Reviews
Features to be added	Yes	Yes
Current Features	Yes	Yes
Step by step how to use	Sometimes	Sometimes
Bugs report	Yes	Sometimes
User complaints	Yes	No
Moods or sentiments	Yes	No
Size	Very Large	Manageable
Probability to find unpleasant words	Yes	No
Bias	Very Likely	Unlikely
Authors	Anyone can write	Experts appointed by organisations

Among important aspects of data from the compiled reviews include the data used will less likely to contain bias, user complaints, moods or sentiments, and the use of unpleasant words. The reason why the firsthand reviews from users were not used in this research is because; the author is aware on the needs to filter out users' complaints and sentiments from the inputs. It is the aim of this research to focus on information related to the software features available for reuse, and possibly to minimise the noises from the input. These aspects are deemed important to ensure features extracted are free from noise that might reduce the accuracy of feature extraction results. Therefore, using compiled reviews from experts will be a better choice.

To suit this purpose, reviews are selected from *toptenreviews.com* websites that provide a compilation of software reviews by experts. Reviews in the *toptenreviews.com* are for users and developers who wanted to get an overview of the product they wanted to buy. These reviews are also beneficial for developers (and domain analysts) who did not have access to SRS and can use these expert reviews as a source for identifying features for the product they

University of Malaya

want to build without having to initiate the RE process from scratch (reuse of requirements). In *toptenreviews.com*, software are reviewed by experts and compiled periodically as compared to the first-hand review data sets used in the related works (Guzman & Maalej, 2014),(Iacob & Harrison, 2013)and (Carreno & Windbladh, 2013). Data used in these compiled reviews are more formal and are believed to contain more information about the product functionalities, and have fewer user complaints such as design flaws or bugs, which make it more usable for this research, i.e. to extract information related to software features.

5.2 Phase 1: Assessing Requirements (software reviews)

Phase 1 seek for software reviews available on the Internet as an alternative to using SRS documents. To demonstrate this, 52 software reviews pertaining to various software products posted in *toptenreviews.com* are extracted. The 52 software reviews came from nine categories as follows:

a) PL1: Preschool Learning (10 compilations)

b) PL2: Algebra Learning (10 compilations)

c) PL3: Language and Reading Software (3 compilations)

d) PL4: Creative Writing (9 compilations)

e) PL5: Vacation Management Software (10 compilations)

f) PL6: Social Networking Site (5 compilations)

h) PL7: Online Storage Service (5 compilations)

i) PL8: Backup Service (10 compilations)

j) PL9: Apps Maker (10 compilations)

University of Malaya

In this experiment, the software reviews obtained are used for two purposes:

1) As the input to the extraction method (as it is), and

2) For creating truth data set for evaluation (see Figure 5.1 for FENL process model).

Section 5.2.1 describes the process for creating the ground truth data set, from the extracted software reviews.

5.2.1 The creation of the ground truth data set

Since there is no ground truth data set for comparing the feature extraction result in the current research context, this data set need to be manually constructed. Table 5.2 list the steps involved in the manual process for creating truth data set.

Table 5-2 Steps for Creating Truth Data set

Steps	Input	Action	Role/Responsibilities	Output
Step 1:	Raw Reviews	Read reviews.	Teachers and Software Practitioners	List of software features
		Highlight potential features and record on the spreadsheets.	Teachers and Software Practitioners	
Step 2:	List of software features	Compile list of software features (from Step 1)	The Author	Truth Data set

In Step 1, potential participants are identified, an invitation email is sent out with the instructions and sample expected output[10] from the manual feature extraction process. The

[10] Sample invitation letter to teachers and the instruction is available at: https://www.dropbox.com/s/393u4p56ljgdeoh/LetterTeachers.docx?dl=0

University of Malaya

description for the task are provided to the participants including the definition of features, following the feature definition provided by (K. Kang et al., 1990). The participants were given two weeks to provide the response. When no response is received within one week, an email reminder is sent out. All of these were done through emails and when questions arise, the questions are attended through emails and phone calls when necessary. Invitation letters were sent to 15 teachers and 15 Software Practitioners, however only eight teachers (see Table 5.3) and eleven software practitioners replied (see Table 5.4) and agreed to participate. The teachers came from various educational institutions (ranging from kindergartens to matriculation centre). The purpose of having Software Practitioners to participate is because there is a need to have the Software Practitioners' perspectives on what can be considered as software features (from technical view), and consolidate it with features extracted by teachers as domain experts. Additionally, in organisations, Software Practitioners usually have experience dealing with various system requirements and are sometimes involved in various system purchasing. As the output from the manual extraction, the participants submitted list of features extracted from the reviews, recorded into spreadsheets.

In Step 2, the list of features extracted by teachers and Software Practitioners are now combined. Some features highlighted by the Software Practitioners but not highlighted by the teachers were added to the truth data set. The teachers were given the reviews that are related to their area of teaching expertise, namely teachers who taught English subjects were asked to do manual extraction for the Creative Writing Software and Language & Reading Software, teachers teaching Mathematics were asked to perform manual extraction for Math and Algebra Software, and kindergarten teachers were asked to do manual extraction for the Preschool Learning software. Teachers who taught Information Technology (IT) or Computer Science

University of Malaya

(CS) subjects at matriculation centre were asked to perform manual extraction for the remaining of the reviews (Online Storage Service, Social Networking and Vacation Management software).

Table 5-3 Demographics Information for teachers involved

No.	Current Institution Address	Total teaching experience	Job Title	SME
1	SMK Taman Daya 3, Jalan Nibong 48, Taman Daya, 81100 Johor Bahru, Johor	15 years	Teacher	Maths
2	SMK Taman Tun Aminah, Jln Bentara 21, Taman Tun Aminah, 81300 Skudai, Johor	13 years	Teacher	English
3	Itqan Intergrated Islamic School, Kampung Sungai Penchala, 60000 Kuala Lumpur	16 years	Teacher	English
4	Centre for Foundation Studies, UIAM Petaling Jaya	9 years	Matriculation Teacher	CS/IT
5	Centre for Foundation Studies, UIAM Petaling Jaya	6 years	Matriculation Teacher	CS/IT
6	Centre for Foundation Studies, UIAM Petaling Jaya	10 years	Matriculation Teacher	Math/CS
7	Qdees Kindergarten, Bandar Seri Putra, 43000 Kajang Selangor	8 years	Kindergarten teacher	Preschool Education
8	Qdees Kindergarten, Bandar Seri Putra, 43000 Kajang Selangor	9 years	Kindergarten teacher	Preschool Education

Similarly, reviews assigned to software practitioners are based on the domain they are currently working on or have been working on previously. The demographics information about the software practitioners that are involved in this process are provided in Table 5.4:

University of Malaya

Table 5-4 Demographic information on Software Practitioners involved

No.	Posts	Organisation Address	No. of years in current job
1.	System Developer	Universiti Kuala Lumpur, ACE - Level 26, Jalan Sultan Ismail, 50250 Kuala Lumpur	5
2	Technical Writer	Level 16 Sutra MARA, Universiti Kuala Lumpur, 1016 Jalan Sultan Ismail, 50250 Kuala Lumpur	7
3.	System Developer	Tingkat 16, Universiti Kuala Lumpur, 1016 Jalan Sultan Ismail, 50250 Kuala Lumpur.	3
4.	System Developer	UniKL, 1016, Jalan Sultan Ismail, 50250 Kuala Lumpur	8
5.	IT Consultant/ Analyst	Freelance	12
6.	System Analysts	Kuwait Finance House (Malaysia) Berhad	12
7.	Software Engineer	Motorola Solutions, Penang	16
8.	Manager - Development team	Scope International, Standard Chartered Bank	14
9.	Technical Writer	Unikl Resources Sdn Bhd, 1016, Jln Sultan Ismail, 50250 Kuala Lumpur, Malaysia	8
10.	System Analysts	Kuwait Finance House (Malaysia) Berhad, Level 8, Menara Prestige, 1 Jalan Pinang, 50450 Kuala Lumpur, Malaysia	15
11.	Senior IT Consultant	MYtech Consulting Services	6 months

From Table 5.4, five out of eleven software professionals are from educational institutions and they have between 3 to 8 years of experience in the software development activities in educational institutions. Their working experiences provide some added values to the creation of the truth data sets (PL1 – PL4 are for the educational domain). Other practitioners who came from financial institutions, consulting services and mobile solutions mentioned to have

University of Malaya

worked with various software development projects and familiar with the software domain for PL5 – PL7 (vacation management software, social networking and cloud storage services).

However, it is important to note that the truth data set created for this experiment does not 100% represent the absolute truth of the data, as manual judgement may varies from human to human. This is understood as the internal threat to validity. The software reviews were extracted in June 2015 and have been made available online[11], for future replication if needed.

5.3 Phase 2: Terms Extraction

The documents being scraped in Phase 1 is now used as the input to the automated terms extraction process. Figure 5.2 lists out the process used for the terms extraction. Steps 1 until 4 in Figure 5.2 are repeated for all selected reviews.

Step 1: Each document went through text preprocessing to remove the stop-words, punctuations, numbers, and special characters.
Step 2: Apply WordNet Lemmatization[12]
Step 3: Apply the Part of Speech Tagging from NLTK[2] to the document and select the required terms (verbs and nouns).
Step 4: The terms with its occurrences were tabulated in a term-document-matrix. In excel, terms that occurs only once and twice are removed.

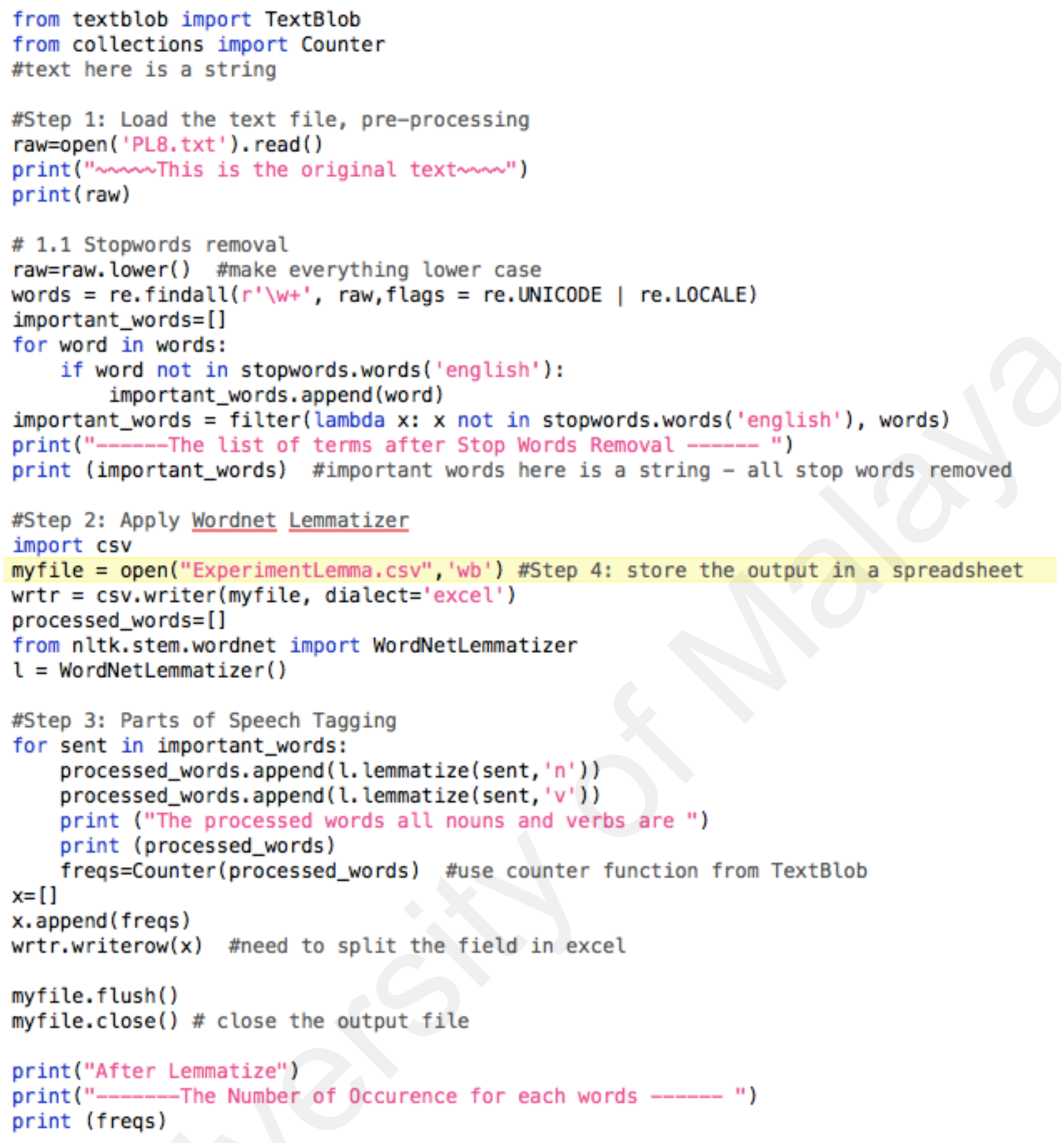
Figure 5.2 Terms extraction process

Figure 5.3 shows the code snippet from python program designed to perform the terms extraction process:

[11]https://www.dropbox.com/sh/kreg4cltqunni9o/AABeNj9kcnDwYFcGFYlG-eb6a?dl=0

http://textanalysisonline.com/nltk-wordnet-lemmatizer

University of Malaya



```
from textblob import TextBlob
from collections import Counter
#text here is a string

#Step 1: Load the text file, pre-processing
raw=open('PL8.txt').read()
print("~~~~~This is the original text~~~~")
print(raw)

# 1.1 Stopwords removal
raw=raw.lower()  #make everything lower case
words = re.findall(r'\w+', raw,flags = re.UNICODE | re.LOCALE)
important_words=[]
for word in words:
    if word not in stopwords.words('english'):
        important_words.append(word)
important_words = filter(lambda x: x not in stopwords.words('english'), words)
print("------The list of terms after Stop Words Removal ------ ")
print (important_words)  #important words here is a string - all stop words removed

#Step 2: Apply Wordnet Lemmatizer
import csv
myfile = open("ExperimentLemma.csv",'wb') #Step 4: store the output in a spreadsheet
wrtr = csv.writer(myfile, dialect='excel')
processed_words=[]
from nltk.stem.wordnet import WordNetLemmatizer
l = WordNetLemmatizer()

#Step 3: Parts of Speech Tagging
for sent in important_words:
    processed_words.append(l.lemmatize(sent,'n'))
    processed_words.append(l.lemmatize(sent,'v'))
    print ("The processed words all nouns and verbs are ")
    print (processed_words)
    freqs=Counter(processed_words)  #use counter function from TextBlob
x=[]
x.append(freqs)
wrtr.writerow(x)  #need to split the field in excel

myfile.flush()
myfile.close() # close the output file

print("After Lemmatize")
print("-------The Number of Occurence for each words ------ ")
print (freqs)
```

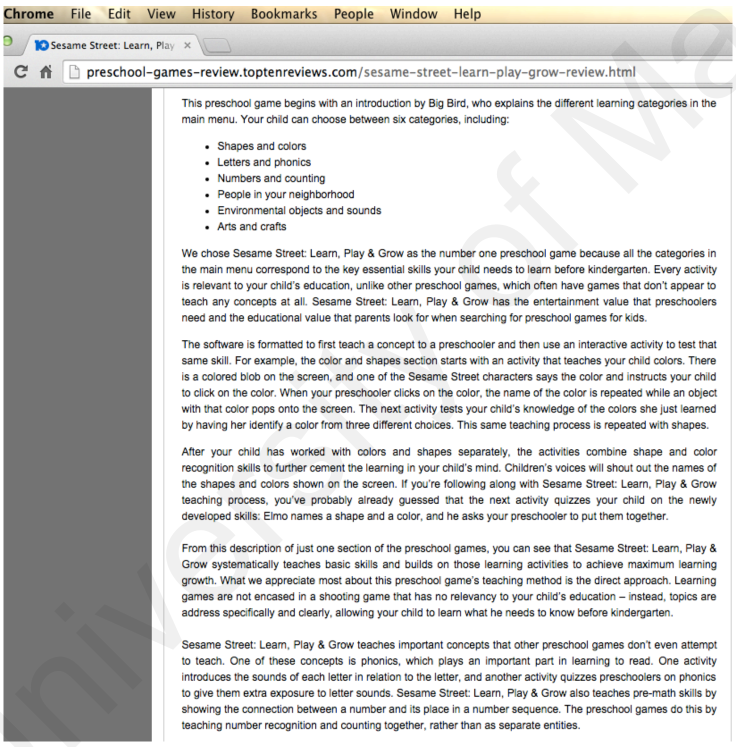
Figure 5.3 Python code for implementing Term Extraction

First, each document went through text preprocessing to remove stop-words (words such as “a, are, the, to, an, at, is,” and more, which do not provide added meaning if considered). Additionally, punctuations, numbers, and special characters are filtered out from the review documents. In Step 2, WordNet lemmatisation was applied. Lemmatisation is a

University of Malaya

process of grouping together the different inflected forms of word so they can be analysed as a single item. For example, words such as "coloring, colored, and colors" are now being referred to as the basic word "color". This helps in reducing the number of words extracted. In Step 3, words are tagged with Parts of Speech Tagging from NLTK. Only nouns, verbs, and adjectives are selected for further processing. In Step 4, the outputs are now exported to spreadsheets. Finally, the term-document-matrix is constructed with terms that occurs once or twice are removed.

Figures 5. 4 – 5.5 shows a sample input and output from the term extraction process:


Chrome File Edit View History Bookmarks People Window Help
Sesame Street: Learn, Play
preschool-games-review.toptenreviews.com/sesame-street-learn-play-grow-review.html

This preschool game begins with an introduction by Big Bird, who explains the different learning categories in the main menu. Your child can choose between six categories, including:

- Shapes and colors
- Letters and phonics
- Numbers and counting
- People in your neighborhood
- Environmental objects and sounds
- Arts and crafts

We chose Sesame Street: Learn, Play & Grow as the number one preschool game because all the categories in the main menu correspond to the key essential skills your child needs to learn before kindergarten. Every activity is relevant to your child's education, unlike other preschool games, which often have games that don't appear to teach any concepts at all. Sesame Street: Learn, Play & Grow has the entertainment value that preschoolers need and the educational value that parents look for when searching for preschool games for kids.

The software is formatted to first teach a concept to a preschooler and then use an interactive activity to test that same skill. For example, the color and shapes section starts with an activity that teaches your child colors. There is a colored blob on the screen, and one of the Sesame Street characters says the color and instructs your child to click on the color. When your preschooler clicks on the color, the name of the color is repeated while an object with that color pops onto the screen. The next activity tests your child's knowledge of the colors she just learned by having her identify a color from three different choices. This same teaching process is repeated with shapes.

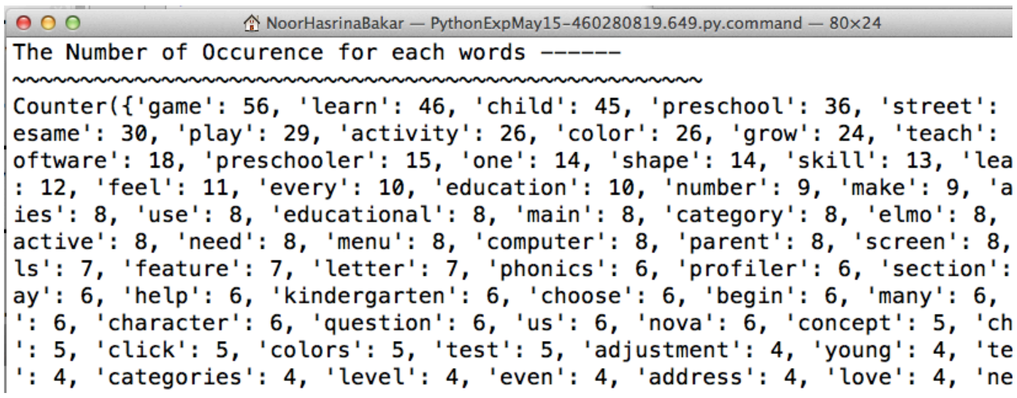
After your child has worked with colors and shapes separately, the activities combine shape and color recognition skills to further cement the learning in your child's mind. Children's voices will shout out the names of the shapes and colors shown on the screen. If you're following along with Sesame Street: Learn, Play & Grow teaching process, you've probably already guessed that the next activity quizzes your child on the newly developed skills: Elmo names a shape and a color, and he asks your preschooler to put them together.

From this description of just one section of the preschool games, you can see that Sesame Street: Learn, Play & Grow systematically teaches basic skills and builds on those learning activities to achieve maximum learning growth. What we appreciate most about this preschool game's teaching method is the direct approach. Learning games are not encased in a shooting game that has no relevancy to your child's education – instead, topics are address specifically and clearly, allowing your child to learn what he needs to know before kindergarten.

Sesame Street: Learn, Play & Grow teaches important concepts that other preschool games don't even attempt to teach. One of these concepts is phonics, which plays an important part in learning to read. One activity introduces the sounds of each letter in relation to the letter, and another activity quizzes preschoolers on phonics to give them extra exposure to letter sounds. Sesame Street: Learn, Play & Grow also teaches pre-math skills by showing the connection between a number and its place in a number sequence. The preschool games do this by teaching number recognition and counting together, rather than as separate entities.

Figure 5.4 Sample raw text scraped from the online reviews

University of Malaya



NoorHasrinaBakar — PythonExpMay15-460280819.649.py.command — 80×24

```
The Number of Occurence for each words ------
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Counter({'game': 56, 'learn': 46, 'child': 45, 'preschool': 36, 'street':
esame': 30, 'play': 29, 'activity': 26, 'color': 26, 'grow': 24, 'teach':
oftware': 18, 'preschooler': 15, 'one': 14, 'shape': 14, 'skill': 13, 'lea
: 12, 'feel': 11, 'every': 10, 'education': 10, 'number': 9, 'make': 9, 'a
ies': 8, 'use': 8, 'educational': 8, 'main': 8, 'category': 8, 'elmo': 8,
active': 8, 'need': 8, 'menu': 8, 'computer': 8, 'parent': 8, 'screen': 8,
ls': 7, 'feature': 7, 'letter': 7, 'phonics': 6, 'profiler': 6, 'section':
ay': 6, 'help': 6, 'kindergarten': 6, 'choose': 6, 'begin': 6, 'many': 6,
': 6, 'character': 6, 'question': 6, 'us': 6, 'nova': 6, 'concept': 5, 'ch
': 5, 'click': 5, 'colors': 5, 'test': 5, 'adjustment': 4, 'young': 4, 'te
': 4, 'categories': 4, 'level': 4, 'even': 4, 'address': 4, 'love': 4, 'ne
```

Figure 5.5 Sample extracted terms from a review (output)

A final spreadsheet contains *n-terms by m-documents* (terms-document matrix, where n represents number of unique terms and m represents the number of documents). Based on the terms collected, the term weights is calculated by using the term frequency inverse document frequency, $tfidf$. The $tfidf$ weight is the weight used in IR and text mining to evaluate how important a word is to a document in a collection. For this case, the spreadsheets comprising collection of terms from various documents are merged and the terms occurring within each document can be clearly seen. This produces the term by document matrix, in which can also be seen as vector.

For Vector Space Model (VSM), the idea originated in (G Salton, Wong, & Yang, 1975) speaks about algebraic model representing textual information as a vector. The components of this vector could represent the importance of a term $tfidf$or even the absence or presence (bag of words) of it in a document. In VSM, terms that occur in documents are represented as vector of numbers. The raw term occurrence records how many times (frequency) each term occurs in every document. The $tfidf$ is calculated for each of the terms that appear in every document by giving a weight according to how relevant a term is in regards to the whole

University of Malaya

corpus. This weight is a statistical measure used to evaluate how important a term is to a document in a collection or corpus. The importance increases proportionally to the number of times a term appears in the document but is offset by the frequency of the term in the whole corpus (Gerard Salton & Buckley, 1988). The $tfidf$ is calculated for each of the terms that appear in each of the documents. Raw weight is computed as the number of times the terms occur in a document. Binary weight assigns 1 if a term occurs and 0 vice versa. Term frequency, tf is calculated by a weight proportional to the frequency of the term occurrence in a given text (raw count of terms *x* in document *y* divided by the length of text fragment):

$$tf(x,y) = \frac{n(x,y)}{\sum_k n(k,y)} \tag{1}$$

$n(x,y)$ is the number of occurrence for term x in a document y.

$\sum_k n(k,y)$ is the sum of occurrences of all terms in document y.

Inverse document frequency, idf is computed by assigning the weight depending on a number of a given text that include the term.

$$idf(x) = \log \frac{|D|}{|d: t(x) \in d|} \tag{2}$$

where:

$|D|$ is the total number of documents

$|d: t(x) \in d|$ is the number of documents where the terms *t(x)* appears.

University of Malaya

The underlying idea for *idf* is that related text within a given domain shares a lot of words; such frequent occurrences do not provide a lot of semantic value. For example, the word “car” in automobile domain occurs frequently, thus they should be given a low weight because it will not add much information to the documents.

The hybrid $tfidf$ computes the multiplication of tf and *idf*. This $tfidf$ assigns to a term weight that is high when the terms occurs many times in a document within a small number of document and assigns a low term weight when the terms occurs few times in a document, or occurs in many documents. $tfidf$ will assign the lowest weight when the terms occur virtually in all documents. $tfidf$ value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others. The hybrid $tfidf$ is calculated as:

$$tfidf(t,d) = tf(t,d) * \ idf(t) \tag{3}$$

The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the collection (Gerard Salton & Buckley, 1988). These $tfidf$ value will be used to determine the document similarities with LSA – will be described in section 5.4. The main outputs for Phase 2 are important terms (verbs and nouns) in each document and their occurrences. These terms will later be used in Phase 3a to identify similar documents (reviews) by using LSA.

University of Malaya

5.4 Phase 3: Feature Identification

Phase 3 is divided into three smaller processes: Phase 3a: Identification of Similar documents, Phase 3b: Extraction of Software Features and Phase 3c: Grouping Similar Features. The following subsections describe each of the phases in detail.

5.4.1 Phase 3a: Identification of Similar Documents

The $tfidf$ values obtained from Phase 2, is now loaded into MATLAB software to create a Vector Space Model's extension, the LSA. LSA was used previously by other researchers in Software Engineering such as (Marcus & Maletic, 2003), (Maletic & Valluri, 1999), (Jiang, Nguyen, & Chen, 2008) . In this thesis, LSA will be used twice: 1) To determine document relatedness and 2) To trace the origin of the extracted features. In LSA, Singular Value Decomposition (SVD calculation) was applied to reduce the dimension of matrix representation to three different matrices: `S,U` and `V`. SVD computation in LSA is what distinguishes the LSA from the more traditional Vector Space Model, VSM. SVD computation reduces the dimension of the document, so that only relevant vectors are considered, while the traditional VSM uses the original dimension of the document which makes it less effective than LSA. Furthermore, VSM uses the keyword matching techniques as compared to concept-based techniques in LSA. The detailed implementation of LSA with the SVD calculation is beyond the discussion of this chapter, and available in (Deerwester, Dumais, Furnas, & Landauer, 1998) for further reference.

For implementing the LSA, MATLAB R2011a application is used. The following command is used to decompose the term-by-document matrix into matrices `S,U` and `V`:

```
[S U V] = svd(A)
```

University of Malaya

Matrix S is a m x m diagonal matrix of eigenvalues for each principal component direction, where m is equivalent to the number of terms. Matrix U is a k X m matrix where k represents the number of documents while m represents the number of terms. The columns in Matrix V when transposed, (V^T) provide a new orthogonal basis for the data, often referred to as principal component directions (Hand, Mannila, & Smyth, 2001). By keeping the dimension to lower dimensions, the SVD computation should bring together the terms with similar co-occurrences. Thus, in the experiment, rank 2 approximation was implemented, so that the first two columns of matrix U and V are kept:

U_k=U(:,1:2)

V_k=V(:,1:2)

The rank 2 approximation was used in the notion that most of the variance in the data is captured by the first two principal components. By retaining these two principal components, when tested for the amount of information been lost (in a mean-square sense) as described in Hand et al.,only 7.5% loss of the information was reported. Following this evidence, the rank 2 approximation is used in this experiment to obtain the rows in V_k which represents the coordinates of individual document vectors. These coordinates when projected to an x-y plane will indicate the position of all documents in the problem space. As a result, unrelated documents were discarded: the documents that are clearly far from other documents in the document space (as computed by LSA) will not be taken into the next phase of the experiment. The basic K-means algorithm is used to confirm the groupings of the documents. K-means

University of Malaya

algorithm is a commonly used clustering algorithm with the aim to optimise an objective function (the distance) that is described by the equation:

$$E = \sum_{i=1}^{c} \sum\nolimits_{x \in C_i} d(x, m_i) \tag{4}$$

m

$\boldsymbol{m_i}$ is the centre of cluster $\boldsymbol{C_i}$, while $\boldsymbol{d(x,m_i)}$ is the Euclidean distance between point $\boldsymbol{x}$ and $\boldsymbol{m_i}$. Figure 5 is the algorithm for K-means (Hand et al., 2001):

```
1. Set a fixed number of clusters, c.
2. Randomly pick up a cluster centre.
3. Assign all points in the data set to the cluster whose centre is
the nearest (closest centroid).
4. Recompute the centres for each centroid.
5. Repeat the process in steps 3 and 4 until the centres stop
changing.
```

Figure 5.6 K-means clustering algorithm (Hand et al., 2001)

After running the K-means algorithm, a graph in Figure 5.7 is plotted to indicate the position of the reviews in the documents space.

University of Malaya

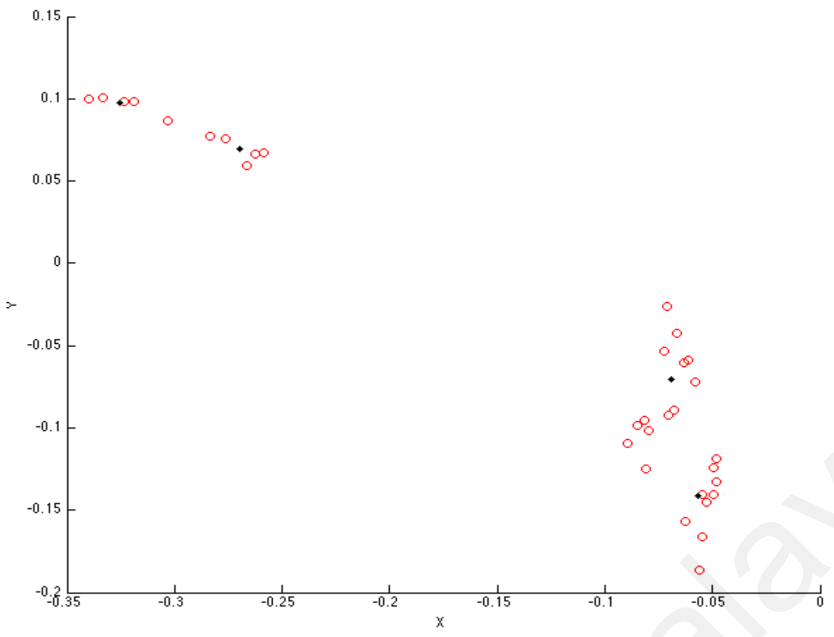


Figure 5.7 Position of reviews in document space that can be grouped into four categories

(for illustrative purposes, this figure only represent the first 32 software reviews)

This observation indicates that LSA is able to group related documents together based on the occurrences of terms that exist. This is especially true when using a reasonable large size of term-document-matrix and producing results that are very close to the categorisation made by human, in the *toptenreviews.com*.

5.4.2 Phase 3b: Extraction of phrases that represent features

In the related works such as (Alves et al., 2008) and (Weston et al., 2009), similar structure of requirement statements are compared because their research used standard requirement documents (i.e. use case specifications and SRS documents). However, when dealing with unstructured documents such as software reviews, measuring sentence similarities is not easily achieved. This is because reviews were written freely and did not follow sentence structure

University of Malaya

such as sentences that exist in SRS. In SRS, sentences are constructed in the form of Verb + Direct Object, for example, "The user shall *click* on the *Exit_Button* to terminate the application." With SRS, linguistic pattern in the extraction algorithm can specifically target on sentences consisting "shall, should, must, etc.", followed by verb and objects (*Exit_Button*), thus, the sentences can directly reflect the functional requirements of a system. However, when comparing to sentences in software reviews (freely written text), there is no standard linguistic pattern specified in the documents that could resemble the structure of functional requirements. This makes it hard to perform comparison towards sentences, in which there is no guarantee that sentences in review documents contain the "`shall statement`" that can represent functional requirements. Thus, in this thesis, the extraction of software features by focusing on selecting combination of noun, verb and adjectives in sentences is believed can bring out the underlying representation of characteristics of software systems or features, particularly statements that may relate to functional requirements.

In this research, the description of features is referred as "*a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems*" as described by (K. C. Kang, Cohen, Hess, Novak, & Peterson, 1990). The work in this thesis is focused on extracting the phrases (bigrams or trigrams words, i.e. combination of nouns, verbs, and adjectives) which is believed to bring out the user-visible characteristics of a system. In this context, for example, terms such as "`number recognition`", "`learning colors`", "`interactive tutorial`", "`multiplechoice quiz`" are considered as terms that represent product features. Figure 5.8 shows steps taken to extract features from reviews.

University of Malaya

```
  Step 1:       Remove Stop-words:

  %partial code to remove stop-words:

  important_words=[]
  for word in words:
    if word not in stopwords.words('english'):
     important_words.append(word)
     important_words    =    filter(lambda    x:    x    not    in
stopwords.words('english'), words)


  Step 2: Apply NLTK WordNet Lemmatization[13] (so that different
inflected word will be treated as one).

  Step 3: Extract the n-grams features by using the linguistic
tag[14] in Figure 5.9
```

Figure 5.8 Phase 3b Extraction of phrases that represent features

University of Malaya

Firstly in Step 1, the scraped texts underwent the stop-words removal process. In Step 2, the WordNet Lemmatizer is used to group different inflected words together. As mentioned earlier in Phase 2, lemmatisation process grouped together the different inflected forms of a word so they can be analysed as a single item. Then, in Step 3, linguistic tags are applied to extract n-gram features. Parts of Speech (POS) tagger provided in NLTK are used and the combination of phrases that occur in form of `<<adjective, noun>>` or `<<noun, adjective>>` AND `<<verb, adjective>>` or `<<adjective, verb>>` AND `<<verb, nouns, adjective>>` are extracted. All possible sequences or arrangements of verbs are

[13] http://textanalysisonline.com/nltk-wordnet-lemmatizer

[14] This configuration is extended from the python tutorial available at https://gist.github.com/shlomibabluki/5539628, last retrieved 1/12/2015

considered as well in this linguistic tagging selection. Table 5.5 describes the acronyms used in the linguistic tags to represent various forms of verbs, adjectives, and nouns:

Table 5-5 Acronyms Used in the Linguistic Tagging

#	Acronyms Used	Representing
1	NN	Nouns
2	NNP	Nouns, Plural
3	NNS	Nouns, Plural
4	JJ	Adjectives
5	VB	Verb Base Form
6	VBZ	Verb, 3rd person Singular Present Tense
7	VBD	Verb, Past Tense
8	VBN	Verb, Past Participle
9	VBG	Verb, Gerund, or Present Participle

Various categories of online learning software were chosen because the FENL approach can be evaluated against diverse words used in the review describing various features. For each review, the title of the review and the reviewed text are extracted. The FENL process is applied on the reviewed text. Three rounds of execution were completed in order to compare which combination of parts of speech tags that extract rather accurate features. Based on the understanding of acronyms in Table 5.5, the three different configurations of linguistic tags for the feature extraction process (in Step 3) will look like:

University of Malaya

```
cfg = {} #Simple Tagging (Verb-direct object)
cfg["VB+NNP"] = "NNP"

cfg = {} #NP Only (Noun Phrase Extraction)
cfg["NN+NN"] = "NNI"
cfg["JJ+NN"] = "NNI"

cfg = {} #FENL (Hybrid)
cfg["VB+NN"] = "NNP"
cfg["NN+JJ+VB"] = "NNP"
cfg["NNP+NN"] = "NNI"
cfg["JJ+NN"] = "NNI"
```

Figure 5.9 Different configuration tagging for feature extraction process

The first configuration, labelled as Simple Tagging in Figure 5.9 extract the verbs and nouns only, similar to the related work (Niu & Easterbrook, 2008) and (Mu et al., 2009) that focusing on Verb + Direct Object in the extraction of functional requirements profile of a software system. Meanwhile the second configuration in Figure 5.9, labelled as Noun Phrase Extraction (NP Only) applies the extraction approach by (Ferrari et al., 2013) that uses nouns and adjectives which is believed to bring out the components of a software system. Additionally, the noun phrase extraction was also mentioned in (Babluki, 2013) tutorial that demonstrates the process to extract the main topics from a sentence. To reflect the definition of features by (K. Kang et al., 1990), FENL will take a hybrid of both configurations.

From Figure 5.9, the tag NNP is used to label the combination of parts of speech category as a Normal Noun Phrase that may consist either Noun + Verb only, or Noun + Adjective + Verbs. The NNI is used to label a longer combination of parts of speech: NNP + NN means Normal Noun Phrases tagged earlier and additional nouns that occur afterwards. Another example is JJ + NN = NNI is used to tagged adjectives and Nouns that forms a longer Noun Phrase, labelled as NNI.

University of Malaya

The accuracy for these three configurations is then analysed and reported in Chapter 6. To compare the results of the automated approach, the truth data set created from the manual extraction performed by teachers and analysts are used.

5.4.3 Phase 3c: Grouping similar features

To group common features together, various clustering approach were used by related works (Hariri et al., 2013), (K. Chen et al., 2005), (Alves et al., 2008), (Yu et al., 2013) to group common features, as detailed out in Chapter 2 earlier. For example, Incremental Diffusive Clustering was used to cluster common features that exists within product listings (Hariri et al., 2013) and Hierarchical Agglomerative Clustering was applied in (K. Chen et al., 2005) to merge requirements to form feature trees.

The following subsection describes the use of K-Means algorithm to cluster similar features, followed by the application of LSA to trace back the origin of extracted features from the actual reviews.

5.4.3.1 Clustering common features with k-Means

The important parameter needed in K-Means clustering includes the number of clusters, the position or the distance between each feature, and how many replications are needed. To obtain the position of each term in the document space, the distance between each feature as computed by the cosine similarity metrics is used. Firstly, the *tfidf* is calculated for all features extracted. Then, the cosine similarity is obtained by finding the dot product between any two features:

University of Malaya

$$\text{Cosine Similarity (f1,f2)} = \text{Dot Product (f1,f2)} / ||f1|| * ||f2|| \qquad (8)$$[15]

In this case, the `f1`is the `tfidf` and `f2` is the `tfidf.T` values obtained. Figure 5.10 indicates the code snippet for finding the pairwise similarity in between extracted features:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vect = TfidfVectorizer(min_df=1)
tfidf = vect.fit_transform(["available idrive",
"available space dropbox",
"available window user web storage service",
"basic home plan",
"basic plan",
"basic storage option opendrive", …. ])
#continue for all the extracted features

result=(tfidf * tfidf.T).A  #finding similarity distance

import csv
b = open('cos.csv', 'w') #output to csv
a = csv.writer(b)
a.writerows(result)  #to export pairwise similarity distance
b.close()
```

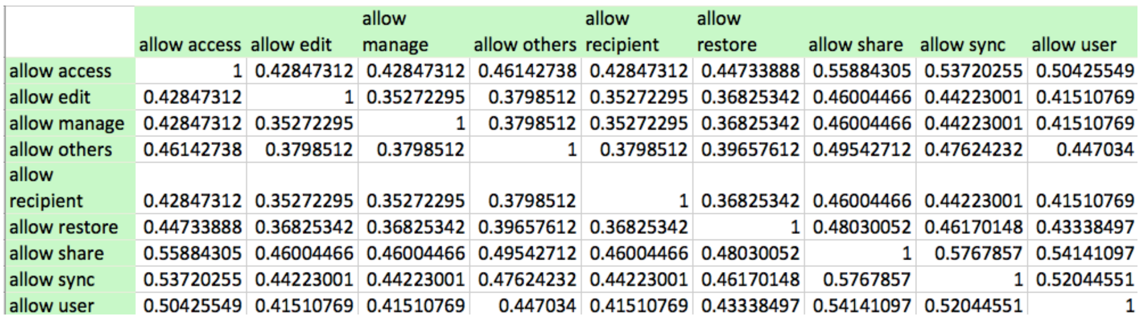
Figure 5.10 Finding similarity distance between extracted features

`TfidfVectorizer` converts a collection of raw documents to a matrix of tf-idf. The function `fit_transform( )` in scikitLearn[16] learn the vocabulary dictionary provided and return the term-document matrix.

As a result, pairwise similarity distance in between features is exported to spreadsheet (.csv file) as appear in Figure 5.11. Distance 0 means features are not related and distance of 1 indicates two features that are very related.

[15] *https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/*
[16] *http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html*

University of Malaya



	allow access	allow edit	allow manage	allow others	allow recipient	allow restore	allow share	allow sync	allow user
allow access	1	0.42847312	0.42847312	0.46142738	0.42847312	0.44733888	0.55884305	0.53720255	0.50425549
allow edit	0.42847312	1	0.35272295	0.3798512	0.35272295	0.36825342	0.46004466	0.44223001	0.41510769
allow manage	0.42847312	0.35272295	1	0.3798512	0.35272295	0.36825342	0.46004466	0.44223001	0.41510769
allow others	0.46142738	0.3798512	0.3798512	1	0.3798512	0.39657612	0.49542712	0.47624232	0.447034
allow recipient	0.42847312	0.35272295	0.35272295	0.3798512	1	0.36825342	0.46004466	0.44223001	0.41510769
allow restore	0.44733888	0.36825342	0.36825342	0.39657612	0.36825342	1	0.48030052	0.46170148	0.43338497
allow share	0.55884305	0.46004466	0.46004466	0.49542712	0.46004466	0.48030052	1	0.5767857	0.54141097
allow sync	0.53720255	0.44223001	0.44223001	0.47624232	0.44223001	0.46170148	0.5767857	1	0.52044551
allow user	0.50425549	0.41510769	0.41510769	0.447034	0.41510769	0.43338497	0.54141097	0.52044551	1

Figure 5.11 A snapshot of cosine similarity results (distance obtained in form of pairwise matrix) among features in PL7

K-Means clustering can be applied based on the distance obtained. Thus, the Matlab command for computing the K-Means clustering will look like:

```
[idx,c]=kmeans(cosPL7,10,'start','sample','replicates',100,'max
iter',1000,'display','final')
```

where :

`cosPL7` represents the cosine similarity distance obtained earlier
`x` is the number of clusters
`idx` is the cluster index
`c` is the centroid for the clusters

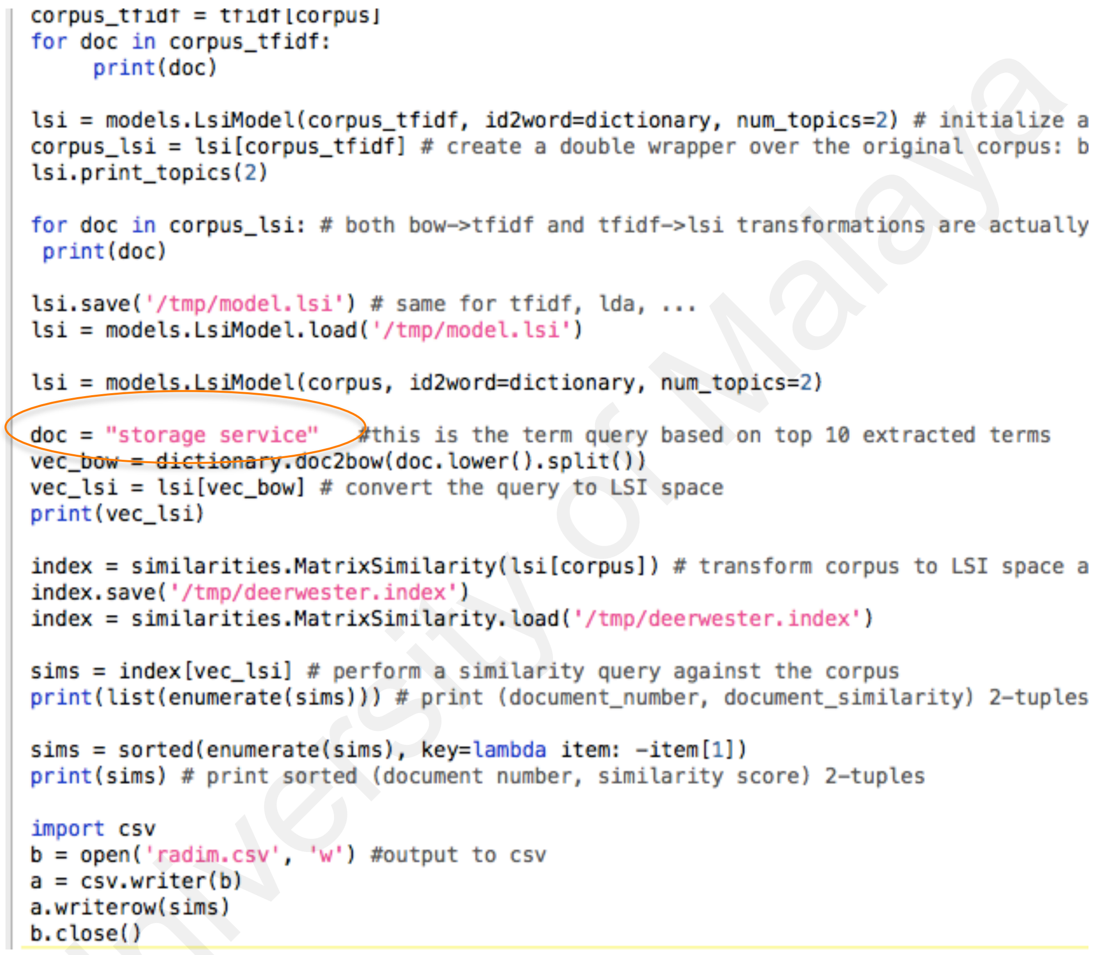
The `idx` value obtained represent the cluster index to indicate which cluster a feature belongs to. The result for this process is presented in Chapter 6.

5.4.3.2 Using LSA to trace the origin of feature clusters

The common features produced by K-Means if presented to the Domain Analysts can be more meaningful if it can be traced back from which sentence it actually comes from. The clustering result contains group of related features. These features can be input to the LSA

University of Malaya

algorithm as query, in which it will be matched with the raw text (raw reviews), to find the actual sentence it came from. The Latent Semantic Analysis implementation from Gensim[17] is adopted for this purpose. Figure 5.12 demonstrates the code snippet for implementing the query, to find term "`storage service`" the commonly used terms in Cluster4 (refer to output in Figure 6.7):



```
corpus_tfidf = tfidf[corpus]
for doc in corpus_tfidf:
     print(doc)

lsi = models.LsiModel(corpus_tfidf, id2word=dictionary, num_topics=2) # initialize a
corpus_lsi = lsi[corpus_tfidf] # create a double wrapper over the original corpus: b
lsi.print_topics(2)

for doc in corpus_lsi: # both bow->tfidf and tfidf->lsi transformations are actually
 print(doc)

lsi.save('/tmp/model.lsi') # same for tfidf, lda, ...
lsi = models.LsiModel.load('/tmp/model.lsi')

lsi = models.LsiModel(corpus, id2word=dictionary, num_topics=2)

doc = "storage service"   #this is the term query based on top 10 extracted terms
vec_bow = dictionary.doc2bow(doc.lower().split())
vec_lsi = lsi[vec_bow] # convert the query to LSI space
print(vec_lsi)

index = similarities.MatrixSimilarity(lsi[corpus]) # transform corpus to LSI space a
index.save('/tmp/deerwester.index')
index = similarities.MatrixSimilarity.load('/tmp/deerwester.index')

sims = index[vec_lsi] # perform a similarity query against the corpus
print(list(enumerate(sims))) # print (document_number, document_similarity) 2-tuples

sims = sorted(enumerate(sims), key=lambda item: -item[1])
print(sims) # print sorted (document number, similarity score) 2-tuples

import csv
b = open('radim.csv', 'w') #output to csv
a = csv.writer(b)
a.writerow(sims)
b.close()
```

Figure 5.12 LSA implementation in Gensim.

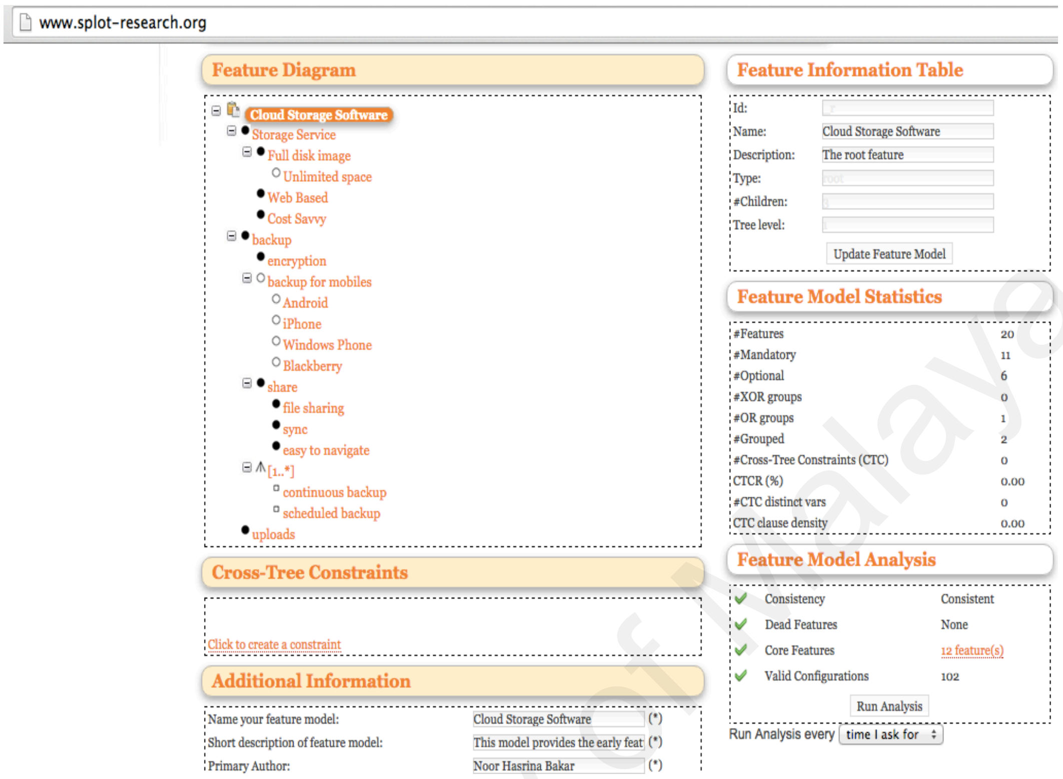
[17] https://radimrehurek.com/gensim/tutorial.html, last accessed 21st January 2015

The Latent Semantic Indexing, LSI or also known as Latent Semantic Analysis, LSA implementation in Gensim is adapted to find which sentence is related to the query. (Note that, the term LSA and LSI are used interchangeably, and LSA was applied earlier in this research to determine document relatedness in Phase 2). The idea behind LSA is that the algorithm will consider the semantic matching (based on the content of the corpus that are matched to the query) instead of using the simple keyword matching like the one in VSM. The result from the above query is a list of sentence number and the similarity measure, given 1 is perfect match and 0 is vice versa.

5.5 Phase 4: Formation of Feature Model

Up to this point, the FENL process has extracted software features and the K-Means is able to group similar features by using software reviews as the input. The grouping result performed in Phase 3b indicates possible combination of features that can be fed to domain analysts as early features in a similar product development. The groupings formed by the clustering algorithm are then passed to Phase 4 for a semi-automated process of constructing the Feature Model. The phrases extracted and grouped by the FENL can be manually transformed to a feature model. For example, from the `cloud storage service` (following example in Figure 5.12), the feature model in Figure 5.13 and feature tree in Figure 5.14 can be generated:

University of Malaya



www.splot-research.org

Feature Diagram

- Cloud Storage Software
  - ● Storage Service
    - ● Full disk image
      - ○ Unlimited space
    - ● Web Based
    - ● Cost Savvy
  - ● backup
    - ● encryption
    - ○ backup for mobiles
      - ○ Android
      - ○ iPhone
      - ○ Windows Phone
      - ○ Blackberry
    - ● share
      - ● file sharing
      - ● sync
      - ● easy to navigate
    - [1..*]
      - continuous backup
      - scheduled backup
  - ● uploads

Cross-Tree Constraints

Click to create a constraint

Additional Information

Name your feature model:	Cloud Storage Software	(*)
Short description of feature model:	This model provides the early feat	(*)
Primary Author:	Noor Hasrina Bakar	(*)

Feature Information Table

Id:	_r
Name:	Cloud Storage Software
Description:	The root feature
Type:	root
#Children:	3
Tree level:	1

Update Feature Model

Feature Model Statistics

#Features	20
#Mandatory	11
#Optional	6
#XOR groups	0
#OR groups	1
#Grouped	2
#Cross-Tree Constraints (CTC)	0
CTCR (%)	0.00
#CTC distinct vars	0
CTC clause density	0.00

Feature Model Analysis

✓	Consistency	Consistent
✓	Dead Features	None
✓	Core Features	12 feature(s)
✓	Valid Configurations	102

Run Analysis

Run Analysis every: time I ask for

Figure 5.13 Feature Model with SPLOT[18] tools

[18] http://www.splot-research.org/

University of Malaya

```
<!--
 This model was created online using SPLOT's Feature Model Editor
(http://www.splot-research.org) on Thu, Jan 21, 2016 - 5:43 AM
-->
<feature_model name="Cloud Storage Software">
<meta>
<data name="description">
This model provides the early features for cloud service software
</data>
<data name="creator">Noor Hasrina Bakar</data>
<data name="address"/>
<data name="email">noor.hasrina@gmail.com</data>
<data name="phone"/>
<data name="website"/>
<data name="organization">University of Malaya</data>
<data name="department"/>
<data name="date"/>
<data name="reference"/>
</meta>
<feature_tree>
:r Cloud Storage Software(_r) :m Storage Service(_r_9) :m Full disk
image(_r_9_10) :o Unlimited space(_r_9_10_11) :m Web Based(_r_9_12) :m Cost
Savvy(_r_9_13) :m backup(_r_15) :m encryption(_r_15_16) :o backup for
mobiles(_r_15_17) :o Android(_r_15_17_18) :o iPhone(_r_15_17_19) :o Windows
Phone(_r_15_17_20) :o Blackberry(_r_15_17_21) :m share(_r_15_23) :m file
sharing(_r_15_23_24) :m sync(_r_15_23_25) :m easy to navigate(_r_15_23_26) :g
(_r_15_29) [1,*] : continuous backup(_r_15_29_30) : scheduled
backup(_r_15_29_31) :m uploads(_r_32)
</feature_tree>
<constraints></constraints>
</feature_model>
```

Figure 5.14 Feature tree generated in SPLOT for the example Cloud Storage Service

This information can be beneficial to the domain analysts. This recommendation, although not 100% accurate, can help analysts to have the main features that exist in publicly available software reviews. The proposed FENL process is hoped to reduce the time spent for requirements engineers reading the entire customer reviews in order to find reusable software features.

Summary

This chapter describes the implementation of the FENL process, the proposed solution for Objective #3 presented earlier in Chapter 3. The implementation is inspired based on the findings reported in Chapter 2 and Chapter 4. This chapter firstly describes the data sets captured from *toptenreviews.com*. The experimental procedure to demonstrate the FENL

University of Malaya

process comprise of 4 main phases: Phase 1: Accessing requirements in Natural Language, Phase 2: Terms Extraction, Phase 3: Feature Identification and Phase 4: Formation of Feature Model. Details for all phases are then described in this chapter. Instead of reinventing the wheels, the implementation of various phases of the experiment for this research is made possible by combining numerous readily available and related open-source APIs resulting from other researchers in the area of information retrieval and software product lines engineering, and the adaptation is made clearly in this chapter. Next, in Chapter 6, the result and evaluation of the experiment conducted will be presented and discussed.

University of Malaya

CHAPTER 6: EXPERIMENTAL RESULTS AND DISCUSSIONS

Introduction

In Software Engineering, the aim of evaluation is to show that the proposed objectives were fulfilled. The purpose of the evaluation in this research comes in twofold: Firstly, evaluation is important to observe to what extent the proposed solution works or practical to solve the problem, and secondly, the evaluation is needed to compare the performance of the proposed approach versus the manual and other related methods in this research area. This chapter firstly describes the evaluation strategy used for the FENL experiment as presented in section 6.1. The result of the FENL experiment is presented in section 6.2, followed by some discussions in section 6.3. Lastly, a chapter summary is provided.

6.1 Evaluation Strategy

In order to compare the performance of information retrieval approach, comparisons are normally being done for two data sets: data extracted from a proposed approach are compared against a truth data set(Manning, Raghavan, & H. Schtze, 2008). Here, a set of ground truth data set is needed, in which ideally should come from a human-defined truth data. This ground truth data set is a supervised data set created that aims to represent the absolute truth, and the data is assumed to be true or it has been validated previously. More discussion on the description of ground truth data set in machine learning by James Kobeilus of IBM is available in (Kobielus, 2014). In this research, the ground truth data set used refers to the training data used in the experiment – the data that is prepared by expert (human), and is assumed to be true. The typical accuracy measures used in information retrieval, namely the Recall, Precision and F-Measure can only be calculated when two data sets are compared. For this research, the

University of Malaya

extracted features resulting from the proposed approach are compared against features being extracted manually (ground truth data set) using the same input - the software reviews. Since there is no truth data set that is available for the software reviews as far as we are aware of (up to the date this thesis is written), the ground truth data set is created for this experiment. The creation of ground truth data set is previously detailed out in Chapter 5.

6.1.1 Manual Extraction and Grouping

In order to evaluate the accuracy of the automated approach, results from manual feature extraction is needed too. This is important to determine how far the proposed approach performed when compared with the manual method. To assist with the manual extraction process, three alumni from the Kulliyyah of Engineering, International Islamic University of Malaysia were asked to perform the manual extraction and grouping process following the steps provided in Table 6.1. Since this evaluation is for experiment purposes with very limited time and resources, the author only able to get three people to help with the manual extraction and grouping. For replication with larger data set in the future, more people will be required to involve. These three alumni have zero knowledge pertaining to the current research project. Nevertheless, these three alumni are currently working as Java programmer at local software house, therefore they have some exposure about software development environment. The procedures which have been used by (Carreno & Windbladh, 2013) for conducting the manual classification of user reviews are adapted:

University of Malaya

Table 6-1 Steps for Manual Feature Extraction and Grouping

Steps	Action
1	Read reviews.
2	Highlight potential features and record into spreadsheets.
3	Manually group features that have similar meanings.
4	Construct Feature Model.

6.1.2 Evaluation Procedure

Previously, the conducted Systematic Literature Review has classified evaluation conducted to be viewed from four angles: context, evaluators, methods and measure used (Bakar et al., 2015). The evaluation conducted in this chapter is in the context of academia and involved evaluators from both industry and academics. From industry, software practitioners and teachers are involved in setting up the data set, while three Java programmers are engaged to participate in the manual extraction process. Lab experiment has been conducted for this research. As for the measure use, the evaluation of FENL employs Recall, Precision and F-Measure.

6.1.3 Phases in Evaluation

The evaluation of FENL is separated into two phases, measuring the accuracy of extraction result, and comparing the feature grouping results produced by clustering algorithms with the manual grouping. Figure 6.1 illustrates the steps taken in the evaluation process.

University of Malaya

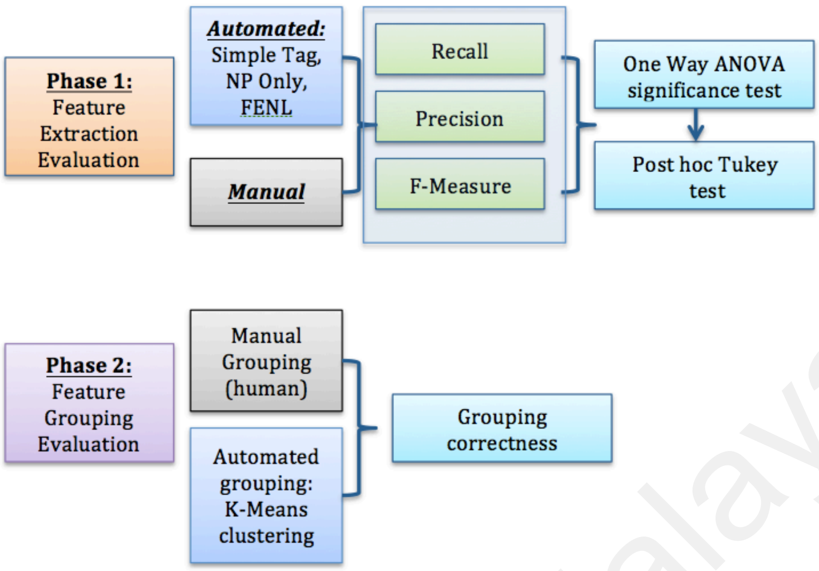


Figure 6.1 Phases in the Evaluation

In addition to using the Recall, Precision and F-Measure in evaluation, statistical measures such as One-Way ANOVA and Tukey post hoc test are used to determine the significance of the results obtained. FENL performance is measured through comparing three versions of automated approach (Simple, NP Only and FENL), benchmarking on the manual extraction approach. As for grouping common features, comparison in between the manually grouped features will be done towards the results produced by K-Means clustering in terms of number of features produced and correctness of features produced. Time taken for executing the semi-automated extraction process is recorded as well.

6.2 Results

This section presents the results obtained from the implementation of FENL in experimental setting by firstly reporting on the distribution of the truth data sets, followed by the results of the features extraction and grouping.

University of Malaya

6.2.1 Truth data set

The number of software reviews that has been used as input in Phase 1 is 72, subdivided into 9 categories. Each document length ranges from 91 to 440 sentences, while the total word lists extracted from all 72 software reviews is 10,766. Table 6.2 summarises the truth data sets used in this study. These datasets are selected roughly based on the expertise area of the teachers and practitioners involved in the experiment. (Note: same reviews were used by both Software Practitioner and teachers).

Table 6-2 Truth data sets of Software Reviews

Learning Software Subcategory	**# of software being reviewed**	**Length (# of sentences)**	**Word Lists ***
PL1: Preschool Learning	10	426	1998
PL2: Algebra Learning	10	296	1144
PL3: Language and Reading Software	3	91	725
PL4: Creative Writing Software	9	440	1140
PL5: Vacation Rental Software	10	185	709
PL6: Social Networking	5	206	883
PL7: Online Storage Service	5	271	852
PL8: Backup Services	10	144	1976
PL9: App Maker	10	261	1339
Total	**72**	**2,320**	**10,766**
** Total number of distinct words after removing stop-words, numbers, special characters etc.*			

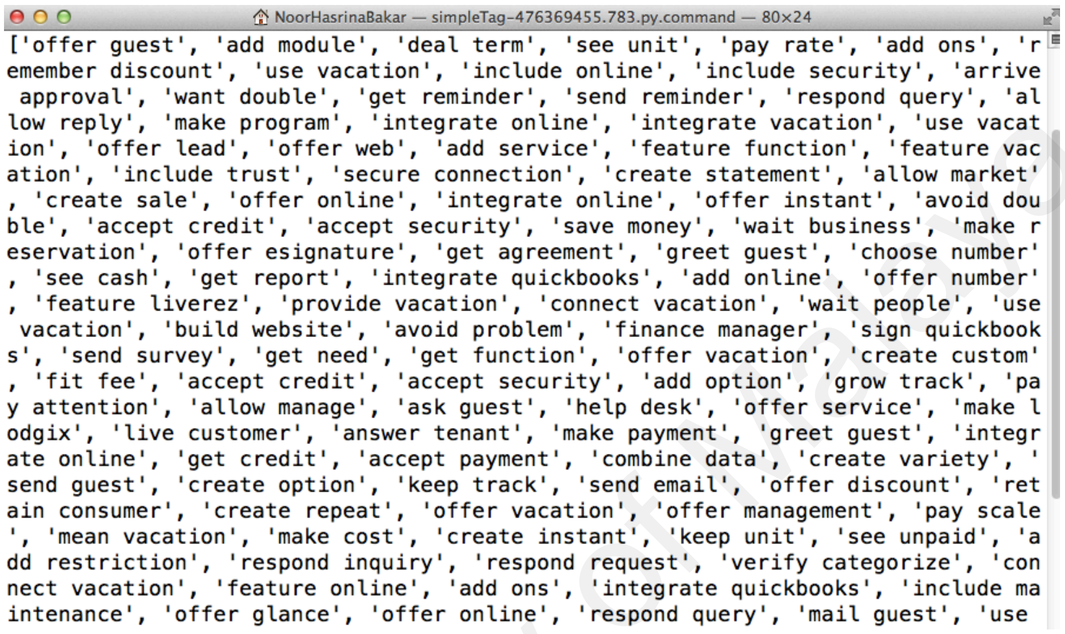
6.2.2 Feature Extraction Results

As described in Chapter 5, the experiment was conducted to compare three configurations: i) Verb + Direct object (Simple Tagging), ii) Noun Phrase tagging (NP Only), and iii) a combination of both (FENL). The extraction output and their accuracy are presented in below subsections.

University of Malaya

6.2.2.1 Extraction Output

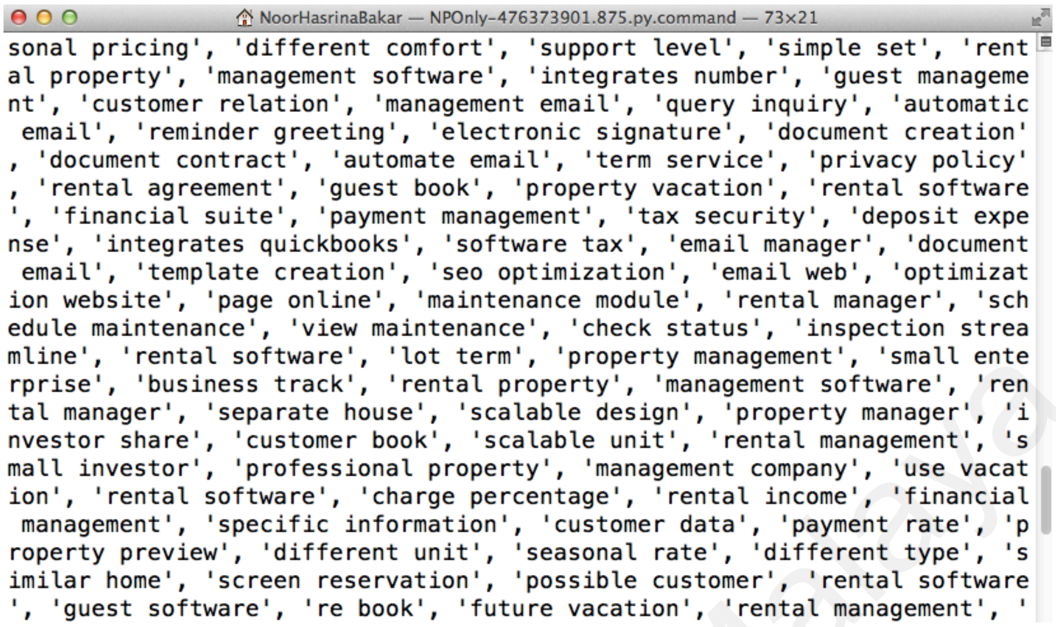
To illustrate the sample output from the three configurations, Figure 6.2, 6.3 and 6.4 presents sample phrases extracted based on the three different configurations for the case of PL5 – reviews for vacation rental software:


NoorHasrinaBakar — simpleTag-476369455.783.py.command — 80×24

```
['offer guest', 'add module', 'deal term', 'see unit', 'pay rate', 'add ons', 'r
emember discount', 'use vacation', 'include online', 'include security', 'arrive
 approval', 'want double', 'get reminder', 'send reminder', 'respond query', 'al
low reply', 'make program', 'integrate online', 'integrate vacation', 'use vacat
ion', 'offer lead', 'offer web', 'add service', 'feature function', 'feature vac
ation', 'include trust', 'secure connection', 'create statement', 'allow market'
, 'create sale', 'offer online', 'integrate online', 'offer instant', 'avoid dou
ble', 'accept credit', 'accept security', 'save money', 'wait business', 'make r
eservation', 'offer esignature', 'get agreement', 'greet guest', 'choose number'
, 'see cash', 'get report', 'integrate quickbooks', 'add online', 'offer number'
, 'feature liverez', 'provide vacation', 'connect vacation', 'wait people', 'use
 vacation', 'build website', 'avoid problem', 'finance manager', 'sign quickbook
s', 'send survey', 'get need', 'get function', 'offer vacation', 'create custom'
, 'fit fee', 'accept credit', 'accept security', 'add option', 'grow track', 'pa
y attention', 'allow manage', 'ask guest', 'help desk', 'offer service', 'make l
odgix', 'live customer', 'answer tenant', 'make payment', 'greet guest', 'integr
ate online', 'get credit', 'accept payment', 'combine data', 'create variety', '
send guest', 'create option', 'keep track', 'send email', 'offer discount', 'ret
ain consumer', 'create repeat', 'offer vacation', 'offer management', 'pay scale
', 'mean vacation', 'make cost', 'create instant', 'keep unit', 'see unpaid', 'a
dd restriction', 'respond inquiry', 'respond request', 'verify categorize', 'con
nect vacation', 'feature online', 'add ons', 'integrate quickbooks', 'include ma
intenance', 'offer glance', 'offer online', 'respond query', 'mail guest', 'use
```

Figure 6.2 is the sample output for extraction of PL5 (vacation rental software).

University of Malaya

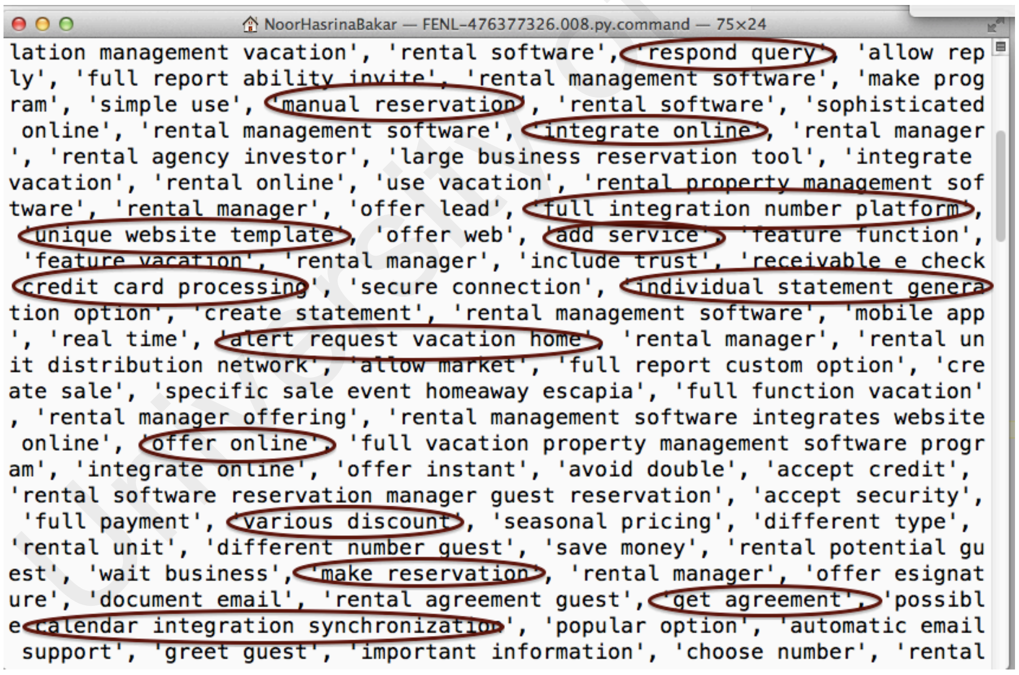


NoorHasrinaBakar — NPOnly-476373901.875.py.command — 73×21

```
sonal pricing', 'different comfort', 'support level', 'simple set', 'rent
al property', 'management software', 'integrates number', 'guest manageme
nt', 'customer relation', 'management email', 'query inquiry', 'automatic
 email', 'reminder greeting', 'electronic signature', 'document creation'
, 'document contract', 'automate email', 'term service', 'privacy policy'
, 'rental agreement', 'guest book', 'property vacation', 'rental software
', 'financial suite', 'payment management', 'tax security', 'deposit expe
nse', 'integrates quickbooks', 'software tax', 'email manager', 'document
 email', 'template creation', 'seo optimization', 'email web', 'optimizat
ion website', 'page online', 'maintenance module', 'rental manager', 'sch
edule maintenance', 'view maintenance', 'check status', 'inspection strea
mline', 'rental software', 'lot term', 'property management', 'small ente
rprise', 'business track', 'rental property', 'management software', 'ren
tal manager', 'separate house', 'scalable design', 'property manager', 'i
nvestor share', 'customer book', 'scalable unit', 'rental management', 's
mall investor', 'professional property', 'management company', 'use vacat
ion', 'rental software', 'charge percentage', 'rental income', 'financial
 management', 'specific information', 'customer data', 'payment rate', 'p
roperty preview', 'different unit', 'seasonal rate', 'different type', 's
imilar home', 'screen reservation', 'possible customer', 'rental software
', 'guest software', 're book', 'future vacation', 'rental management', '
```

Figure 6.3 Features extracted with NP Only

University of Malaya



NoorHasrinaBakar — FENL-476377326.008.py.command — 75×24

```
lation management vacation', 'rental software', 'respond query', 'allow rep
ly', 'full report ability invite', 'rental management software', 'make prog
ram', 'simple use', 'manual reservation', 'rental software', 'sophisticated
 online', 'rental management software', 'integrate online', 'rental manager
', 'rental agency investor', 'large business reservation tool', 'integrate
vacation', 'rental online', 'use vacation', 'rental property management sof
tware', 'rental manager', 'offer lead', 'full integration number platform',
 'unique website template', 'offer web', 'add service', 'feature function',
 'feature vacation', 'rental manager', 'include trust', 'receivable e check
credit card processing', 'secure connection', 'individual statement genera
tion option', 'create statement', 'rental management software', 'mobile app
', 'real time', 'alert request vacation home', 'rental manager', 'rental un
it distribution network', 'allow market', 'full report custom option', 'cre
ate sale', 'specific sale event homeaway escapia', 'full function vacation'
, 'rental manager offering', 'rental management software integrates website
 online', 'offer online', 'full vacation property management software progr
am', 'integrate online', 'offer instant', 'avoid double', 'accept credit',
'rental software reservation manager guest reservation', 'accept security',
 'full payment', 'various discount', 'seasonal pricing', 'different type',
'rental unit', 'different number guest', 'save money', 'rental potential gu
est', 'wait business', 'make reservation', 'rental manager', 'offer esignat
ure', 'document email', 'rental agreement guest', 'get agreement', 'possibl
e calendar integration synchronization', 'popular option', 'automatic email
 support', 'greet guest', 'important information', 'choose number', 'rental
```

Figure 6.4 Features extracted with FENL configuration.

Simple tagger extracts verb and direct objects, while the noun phrase tagger extracts the nouns phrase that can represent the components of the software system. When combined, the linguistic tags in Figure 6.4, FENL picked up some visible characteristics of a software system that might have been missed out by the first two methods. For example, features such as "`individual statement generation`" or "`calendar integration synchronization`" as extracted by FENL were not in the lists from the first two taggers. Features extracted by FENL in this example provide more information when compared with the previous two taggers. The details comparison on the number of features generated by each configuration is discussed in following sub section.

6.2.2.2 Extraction Accuracy

Figure 6.5 indicates the comparison between manual and the three automated extraction approaches in terms of number of features extracted for all nine categories of reviews.

University of Malaya

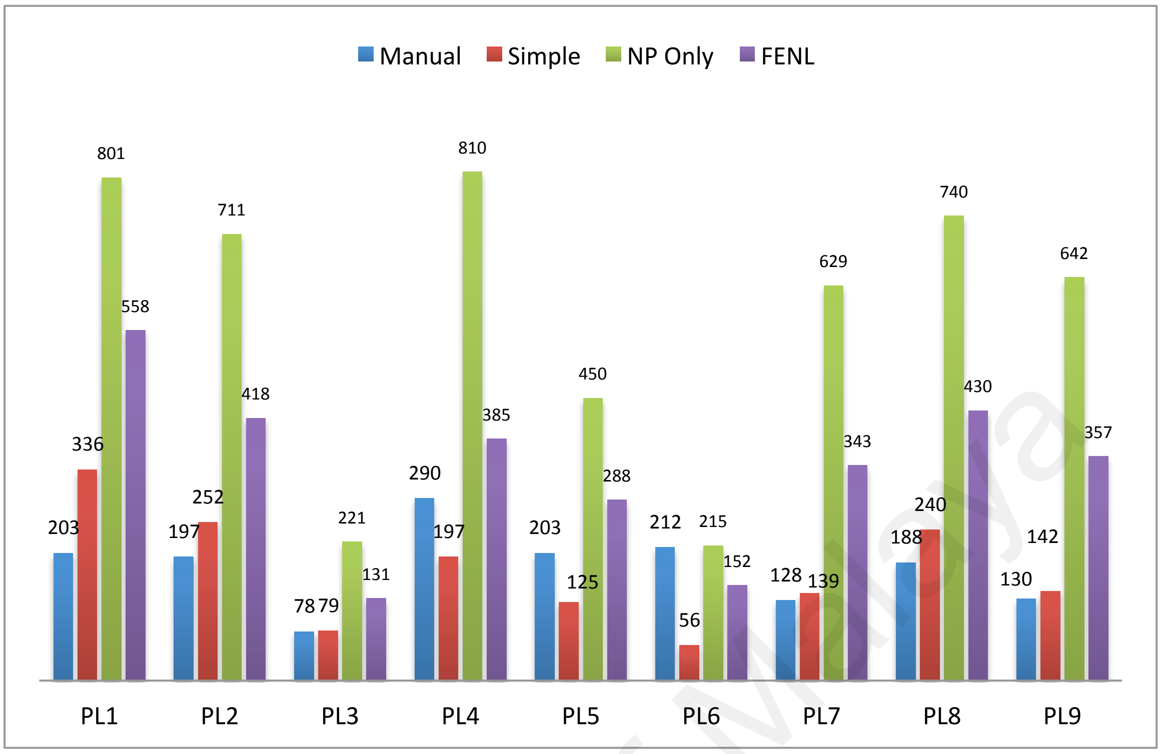


Figure 6.5 Number of features extracted by manual approach as compared to the automated approach.

All features extracted from online learning software (for all categories) are sorted by using pivot table features in Microsoft Excel. NP Only produces the biggest number of features across all product lines, because this approach extract noun phrases or objects that resides within the software reviews. When compared to Manual, simple tagging however produces slightly higher number of features for six product lines: PL1, PL2, PL3, PL7, PL8 and PL9. FENL performs steadily across all product lines (produces higher number of features if compared to manual and simple tagging, except for the case of PL6. Overall, it is observed that NP Only extract the highest number of features from the data set, and most of these does not represent features, but instead the objects mentioned in the reviews. From observing the NP Only results, not all the features are actually relevant, as it contains some noises - refer to Figure 6.3. For example, terms such as "`small enterprise`" and "`possible`

University of Malaya

`customer`" are noises, they did not represent software features. To compare the accuracy of the all extraction approach (manual and all three automated), the Recall, Precision and F-Measure are calculated based on the total features exist in the truth data set versus total features extracted by all of the approach. Recall, Precision, and F-Measure are calculated as follows:

$$Recall = \frac{No. of\ True\ Positives}{number\ of\ correct\ features} \tag{5}$$

$$Precision = \frac{No. of\ True\ Positives}{number\ of\ retrieved\ features} \tag{6}$$

$$F - Measure = 2\ X\ \frac{Precision\ X\ Recall}{Precision + Recall} \tag{7}$$

True Positives are obtained by calculating how many feature exists within the data set that are also extracted by the approach. Precision calculates the percentage of true positives over features retrieved by the approach. Recall calculates the percentage of true positives from the actual features that exist in the truth data set (correct features). In usual case, precision and recall are complementary measures and usually increasing one of them results in the decrease of the other (Poshyvanyk, Gethers, & Marcus, 2012). Figure 6.6, Table 6.7 and Table 6.8 present the results for Recall, Precision and F-Measure.

University of Malaya

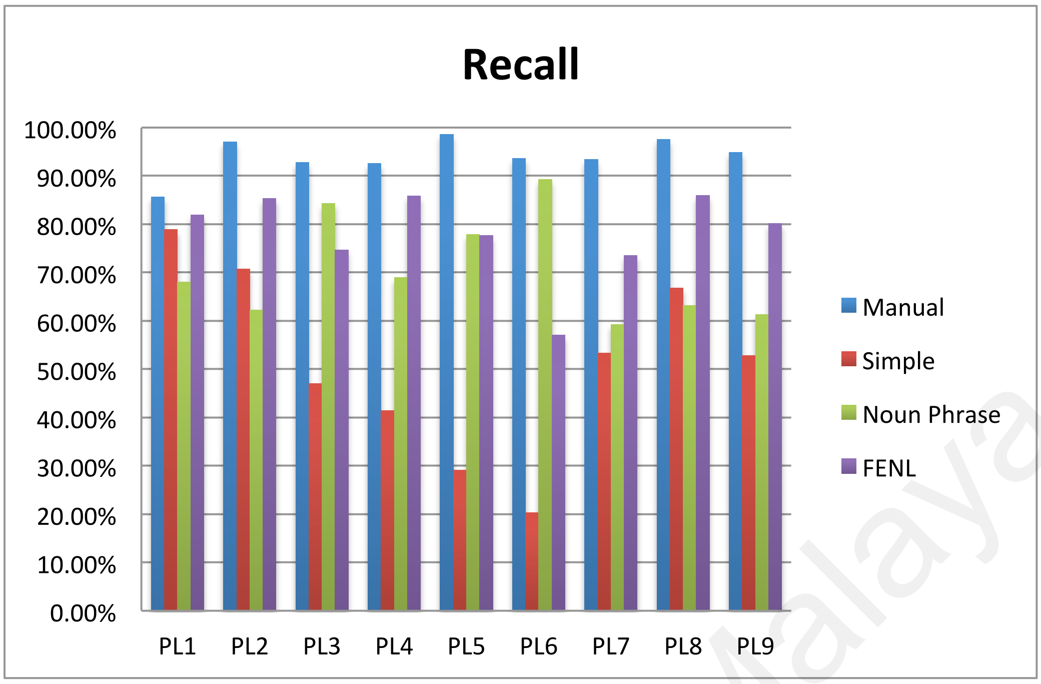


Figure 6.6 Recall results

Obtaining higher recall value indicates that an approach return most relevant results. From Figure 6.6, it is observed that FENL returned in between 57.06% to 85.85% recall result for all nine product lines. These make the average recall results for FENL to be at 78.03%, the highest among the three automated approach. When observing the recall result produced by the manual approach, lowest recall result is recorded by PL1 at 85.65% and by PL4 at 92.60%. This observation agrees with the problem statement specified earlier: manual requirements reuse process can decrease productivity, especially when dealing with larger data. (When referring to Table 6.2, PL1 consists of 426 sentences and PL4 consists of 440 sentences, categories that holds the most number of sentences).

University of Malaya

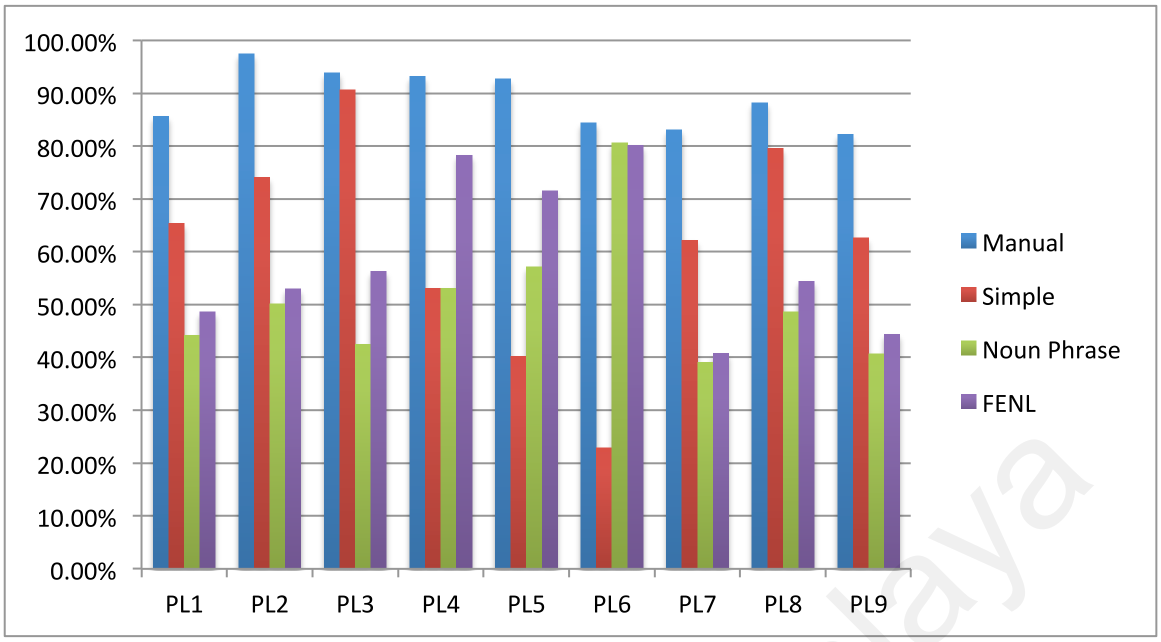


Figure 6.7 Precision results

Figure 6.7 presents the precision results. Higher precision value indicates that an approach returned significantly more relevant results than the irrelevant ones. From the results obtained, FENL recorded the highest precision result for PL6. PL2 records recall value at 85.31% but its precision is only at 53.04%. In PL2 case, the lower precision value is due to high number of false positives identified, in which FENL extracted some noises. The average precision result for FENL is slightly lower (58.63%).

F-measure combines the precision and recall with an equal weight, and this is shown in Figure 6.8.

University of Malaya

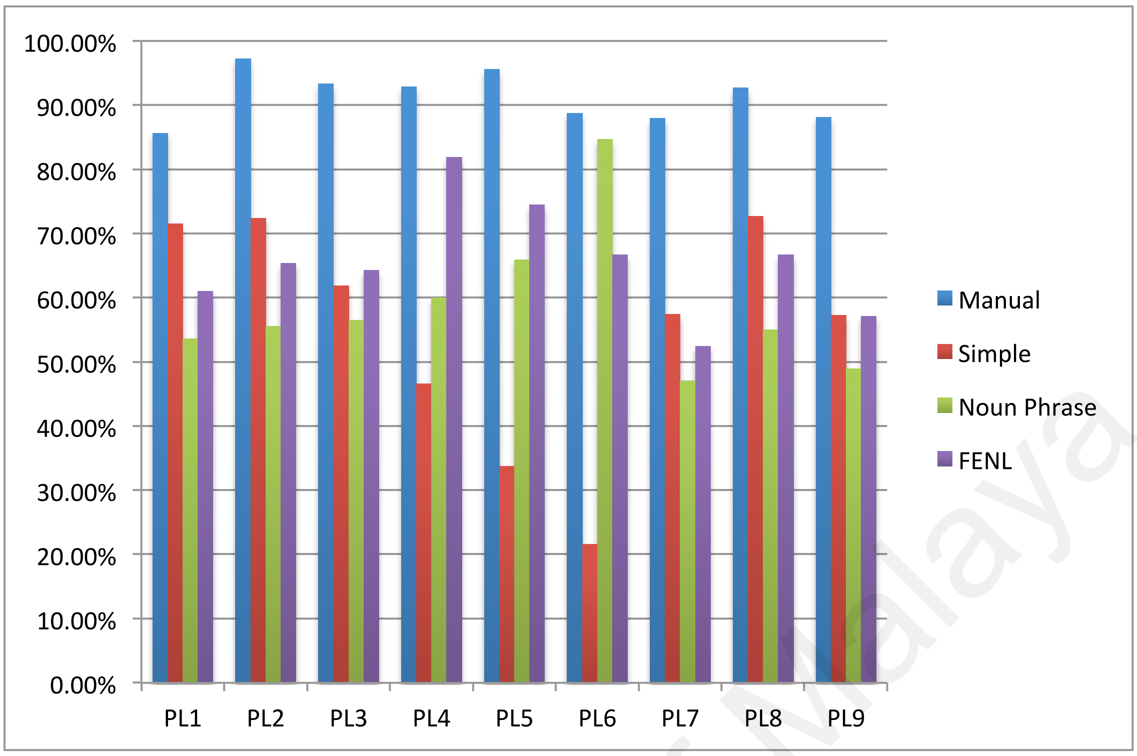


Figure 6.8 F-Measure results

University of Malaya

From F-Measure results obtained, FENL performs better than NP approach for all product lines except for PL8. thus making the average F-Measure for FENL approach to be at 65..56%, about 25% lower to manual approach.

Figure 6.9 depicted the overall performance for Manual and the three automated approach.

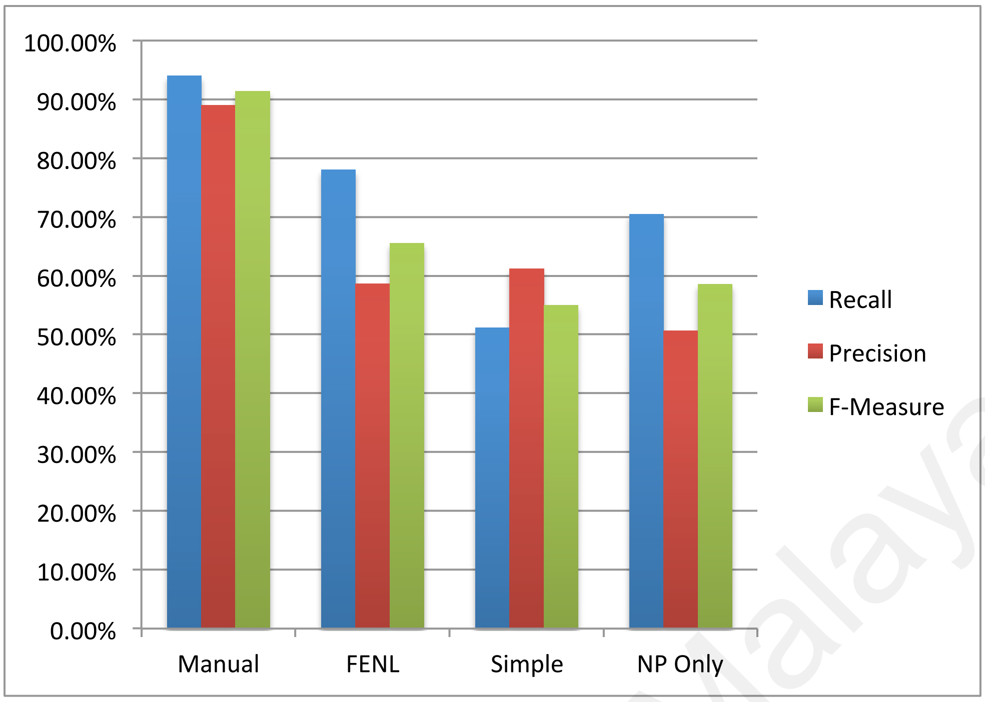


Figure 6.9 Average performance comparisons

With benchmarking on the manual approach, the performance for FENL is superior when compared to the other two automated approaches (looking at F-Measure average in Table 6.9).

To determine whether there was statistically significant difference between the means produced by different extraction methods, One Way ANOVA test has been conducted by using SPSS software and the results is reported in Table 6.3.

University of Malaya

Table 6-3 Result for One-Way ANOVA test

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Recall	Between Groups	8541.889	3	2847.296	19.482	.000
	Within Groups	4676.789	32	146.150		
	Total	13218.677	35			
Precision	Between Groups	7518.478	3	2506.159	12.169	.000
	Within Groups	6590.543	32	205.954		
	Total	14109.021	35			
FMeasure	Between Groups	7273.421	3	2424.474	17.883	.000
	Within Groups	4338.447	32	135.576		
	Total	11611.867	35			

Based on the sample data, there was statistically significant difference between groups as determined by one-way ANOVA at F(3,32)=19.482, p=0.000 for Recall, at F(3,32)=12.169, p=0.000 for precision and at F(3,32)=17.883, p=0.000 for F-Measure.

Table 6-4 Tukey-HSD post hoc test for Recall

		Subset for alpha = 0.05		
Method	N	1	2	3
Simple	9	51.1700		
Noun Phrase	9		70.4922	
FENL	9		78.0289	
Manual	9			94.0167
Sig.		1.000	.556	1.000

The Tukey post hoc test indicated that based on the sample data for Recall, the Simple and Manual method differs significantly from the other groups (p<0.05).

University of Malaya

Table 6-5 Tukey-HSD post hoc test for Precision

Method	N	Subset for alpha = 0.05	
		1	2
Noun Phrase	9	50.6989	
FENL	9	58.6322	
Simple	9	61.2167	
Manual	9		89.0011
Sig.		.418	1.000

Table 6-6 Tukey-HSD post hoc test for F-Measure

Method	N	Subset for alpha = 0.05	
		1	2
Simple	9	55.0144	
Noun Phrase	9	58.6011	
FENL	9	65.5644	
Manual	9		91.3622
Sig.		.239	1.000

Additionally, Table 6.5 and Table 6.6 shows that for Precision and F-Measure the Tukey post hoc with Manual extraction process differs significantly from the other three methods, at $p<0.05$. (For further observation on the results, See APPENDIX G for detail statistics obtained for One Way ANOVA test and Tukey HSD).

Performance in terms of time efficiency, the time taken to complete the extraction process is also recorded in Table 6.7. Note that, the time recorded is only for executing the feature extraction (Phase 3b).

University of Malaya

Table 6-7 Time taken FENL

Software Review	Execution time (in sec)
PL1: Preschool Learning	9.8
PL2: Algebra Learning	6.1
PL3: Language and Reading Software	5.5
PL4: Creative Writing Software	6.6
PL5: Vacation Rental Software	6.0
PL6: Social Networking	5.4
PL7: Online Storage Service	5.9
PL8: Backup Services	6.3
PL9: Apps Maker	6.1

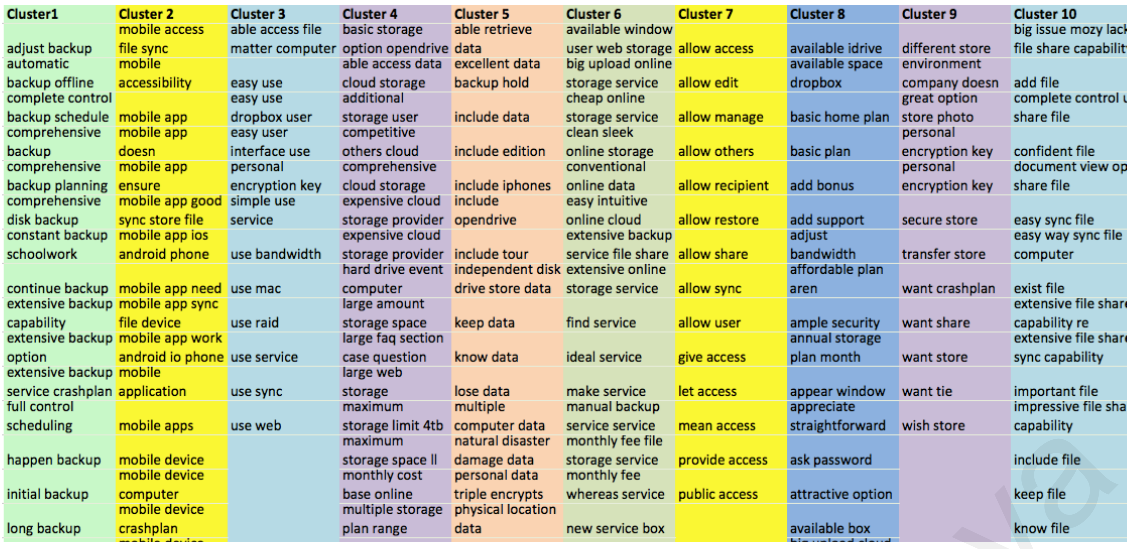
6.2.3 Feature Grouping Results

For this research experiment, the procedure for clustering similar features is explained in Chapter 5. The following subsection presents the features clustering results.

6.2.3.1 Clustering results to group common features

The `idx` obtained from K-Means algorithm indicates which cluster each of the features belongs to. For example, in case of PL7, Figure 6.7 shows the clustering output produced after executing the K-Means algorithm:

University of Malaya

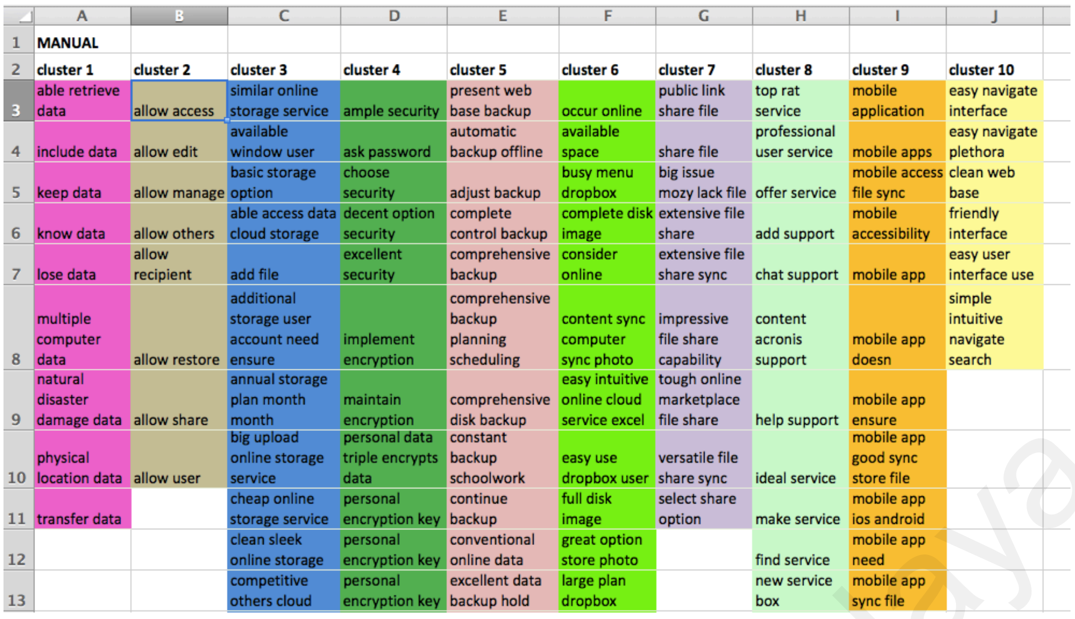


Cluster1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
adjust backup	mobile access file sync	able access file matter computer	basic storage option opendrive	able retrieve data	available window user web storage	allow access	available idrive	different store	big issue mozy lack file share capabilit
automatic backup offline	mobile accessibility	easy use	able access data cloud storage	excellent data backup hold	big upload online storage service	allow edit	available space dropbox	environment company doesn	add file
complete control backup schedule	mobile app	easy use dropbox user	additional storage user	include data	cheap online storage service	allow manage	basic home plan	great option store photo	complete control u share file
comprehensive backup	mobile app doesn	easy user interface use	competitive others cloud	include edition	clean sleek online storage	allow others	basic plan	personal encryption key	confident file
comprehensive backup planning	mobile app ensure	personal encryption key	comprehensive cloud storage	include iphones	conventional online data	allow recipient	add bonus	personal encryption key	document view op share file
comprehensive disk backup	mobile app good sync store file	simple use service	expensive cloud storage provider	include opendrive	easy intuitive online cloud	allow restore	add support	secure store	easy sync file
constant backup schoolwork	mobile app ios android phone	use bandwidth	expensive cloud storage provider	include tour	extensive backup service file share	allow share	adjust bandwidth	transfer store	easy way sync file computer
continue backup	mobile app need	use mac	hard drive event computer	independent disk drive store data	extensive online storage service	allow sync	affordable plan aren	want crashplan	exist file
extensive backup capability	mobile app sync file device	use raid	large amount storage space	keep data	find service	allow user	ample security	want share	extensive file shar capability re
extensive backup option	mobile app work android io phone	use service	large faq section case question	know data	ideal service	give access	annual storage plan month	want store	extensive file shar sync capability
extensive backup service crashplan	mobile application	use sync	large web storage	lose data	make service	let access	appear window	want tie	important file
full control scheduling	mobile apps	use web	maximum storage limit 4tb	multiple computer data	manual backup service service	mean access	appreciate straightforward	wish store	impressive file sha capability
happen backup	mobile device		maximum storage space ll	natural disaster damage data	monthly fee file storage service	provide access	ask password		include file
initial backup	mobile device computer		monthly cost base online	personal data triple encrypts	monthly fee whereas service	public access	attractive option		keep file
long backup	mobile device crashplan		multiple storage plan range	physical location data	new service box		available box		know file

Figure 6.10 Output from K-Means clustering for PL7

Features extracted by FENL are clustered with the K-means clustering. For example, in Figure 6.10, the result of K-Means clustering, in which most related features are grouped together reasonably, in a very short time. Manual grouping by human is very prone to mistakes (See Figure 6.11, the FENL features when they are grouped manually by human). Some features are correctly grouped together, yet some features are not. Moreover, the three alumni reported that performing the features grouping manually is very arduous. They did not complete the tasks given to group all nine product lines, but instead only 5 product lines is completed. On average, time taken to process each product line reaches more than 45 minutes.

University of Malaya



	A	B	C	D	E	F	G	H	I	J
1	MANUAL									
2	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	cluster 8	cluster 9	cluster 10
3	able retrieve data	allow access	similar online storage service	ample security	present web base backup	occur online	public link share file	top rat service	mobile application	easy navigate interface
4	include data	allow edit	available window user	ask password	automatic backup offline	available space	share file	professional user service	mobile apps	easy navigate plethora
5	keep data	allow manage	basic storage option	choose security	adjust backup	busy menu dropbox	big issue mozy lack file	offer service	mobile access file sync	clean web base
6	know data	allow others	able access data cloud storage	decent option security	complete control backup	complete disk image	extensive file share	add support	mobile accessibility	friendly interface
7	lose data	allow recipient	add file	excellent security	comprehensive backup	consider online	extensive file share sync	chat support	mobile app	easy user interface use
8	multiple computer data	allow restore	additional storage user account need ensure	implement encryption	comprehensive backup planning scheduling	content sync computer sync photo	impressive file share capability	content acronis support	mobile app doesn	simple intuitive navigate search
9	natural disaster damage data	allow share	annual storage plan month month	maintain encryption	comprehensive disk backup	easy intuitive online cloud service excel	tough online marketplace file share	help support	mobile app ensure	
10	physical location data	allow user	big upload online storage service	personal data triple encrypts data	constant backup schoolwork	easy use dropbox user	versatile file share sync	ideal service	mobile app good sync store file	
11	transfer data		cheap online storage service	personal encryption key	continue backup	full disk image	select share option	make service	mobile app ios android	
12			clean sleek online storage	personal encryption key	conventional online data	great option store photo		find service	mobile app need	
13			competitive others cloud	personal encryption key	excellent data backup hold	large plan dropbox		new service box	mobile app sync file	

Figure 6.11 Output from manual feature clustering for PL7

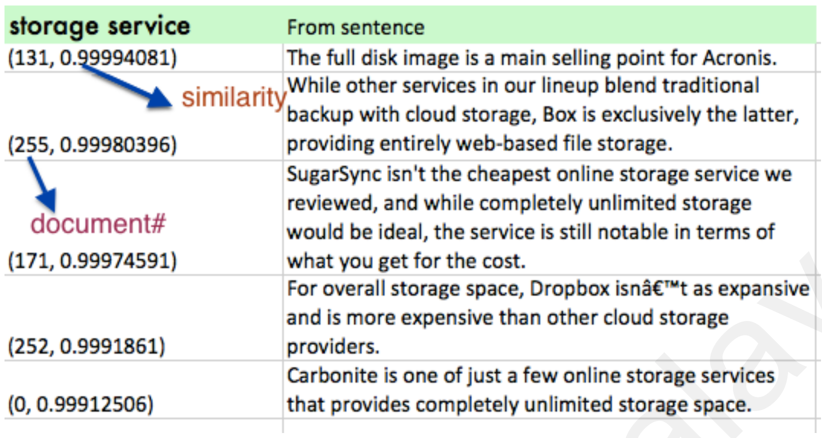
The person doing manual clustering was unable to complete the clustering tasks for all nine product lines and complained that this grouping tasks as too tedious. In terms of correctness, let us observe the sample grouping results presented in Figure 6.10 and Figure 6.11 for PL7. Looking at cluster2 (in Figure 6.10) and cluster9 (Figure 6.11). These two clusters grouped features related to mobile app, and most of the features related to mobile are successfully clustered by K-Means, and they as well appear in the result manually formed by human. This indicates that with the extracted features, normal clustering algorithms can produce cluster that is similar to the one done by human. However, more experiment with clustering algorithm is left for future work, as this thesis is focusing more on feature extractions.

6.2.3.2 Tracing features with Latent Semantic Analysis

The most commonly used terms from clustering results are used as query in LSI for tracing back the origin of extracted features. This task is needed, because if a requirement engineer is presented with set of features, they might not get full idea of the complete sentence of that

University of Malaya

feature. Therefore, querying back on where these features come from may provide added value to the feature extraction results. Figure 6.12 is a sample snapshot of how the related sentences are matched to the query (features provided).



similarity
document#

storage service	From sentence
(131, 0.99994081)	The full disk image is a main selling point for Acronis.
(255, 0.99980396)	While other services in our lineup blend traditional backup with cloud storage, Box is exclusively the latter, providing entirely web-based file storage.
(171, 0.99974591)	SugarSync isn't the cheapest online storage service we reviewed, and while completely unlimited storage would be ideal, the service is still notable in terms of what you get for the cost.
(252, 0.9991861)	For overall storage space, Dropbox isnâ€™t as expansive and is more expensive than other cloud storage providers.
(0, 0.99912506)	Carbonite is one of just a few online storage services that provides completely unlimited storage space.

Figure 6.12 Matching selected terms by using LSI

The extracted sentences that are matched with the query in LSI provide semantic similarity. The parameters in the bracket consist of document number, (in this case which sentence it came from), and the similarity value (how similar the sentence is to the feature being queried). For example, the sentence number 131 in Figure 6.12, “`The full disk image is a main selling point for Acronis`”. None of the keyword in the sentence overlapped with the term “`storage service`”. However, the term “`disk image`” from the sentence, if looking at its general definition provided by Wikipedia, “`disk image`” is defined as “*computer files containing the contents and structure of a disk volume or an entire data storage device, such as a hard disk drive, tape drive, floppy disk, optical disc or USB flash drive*”[19], which semantically related to the “`storage service`”.

University of Malaya

[19] https://en.wikipedia.org/wiki/Disk_image

6.3 Discussions on the Results

This section discusses the findings from the experiment conducted for this research. A discussion on the observation made towards the experiment results is provided followed by a discussion on the threat to validity pertaining to the data sets used in the experiment.

6.3.1 Discussions on the extraction results

As the final output from the experiment, the FENL process has extracted software features from the reviews. The phrases extracted by the FENL can be manually transformed to a feature model, i.e can be fed to SPLOT tools for semi-auto feature model construction (sample feature model constructed by SPLOT as presented in Chapter 5).

In the experiment conducted, FENL performs comparably with manual extraction approach, especially when observing the recall values produced. Higher recall values indicates the relevant features that are finally selected. Although FENL extracted some noises (irrelevant items that is indicated by lower precision values), it is important to note that there is an average of 76.59% of the relevant items which consists of actual features (recall).

The average recall, precision and F-measure results obtained by the FENL in comparison with related works that uses similar evaluation measure is presented in Table 6.8. There are three recent studies that reported similar evaluation results (Guzman & Maalej, 2014), (Carreno & Windbladh, 2013) and (Khan, Baharudin, & Khan, 2014). Other related works were not included in this comparison either because their approach did not present the evaluation results in terms of Precision, Recall and F-Measure (K. Chen et al., 2005) and (Ferrari et al., 2013) or they did not use the data set of similar nature, i.e. user reviews (for

University of Malaya

example (Niu & Easterbrook, 2008) , (Weston et al., 2009) and (Boutkova & Houdek, 2011) uses SRS documents as their input and not user reviews, thus comparison cannot be made).

Table 6-8 FENL Versus Related Works (Average Precision, Recall, and F-Measure)

Feature Extraction Approach by related works	Precision	Recall	F-Measure
Guzzman (2014)	0.582	0.520	0.549
Carreno and Windbladh (2013)	0.941	0.670	0.782
Khan et al. (2014)	0.790	0.717	0.752
FENL	0.586	0.780	0.656

From Table 6.10, FENL reported a lower F-Measure when compared with Khan et al. (2014) and Carreno and Windbladh (2013), but performed slightly better if compared to Guzzman's work. This result tells us that FENL approach performed comparably with related works.

Even though various clustering algorithms were mentioned in other related works such as (Alves et al., 2008), (Weston et al., 2009), and (Davril et al., 2013), this research only explore on the possibility to perform clustering with K-Means algorithm. However, experimenting with other clustering algorithms can be an interesting future work.

6.3.2 Discussions on the data set

Reviews that are compiled by experts are chosen in this experiment because the objective of the experiment is to demonstrate the extraction of software features that can aid the RR process. Hence, the focus was to extract the software features that may reside within the software review are believed to contain very minimal customers complaints and sentiments on a software product. This is different from the related works such as (Guzman & Maalej, 2014)

University of Malaya

or (Carreno & Windbladh, 2013) that uses first hand user reviews as the input, thus their extraction approach aiming to extract the user opinions about the software products on top of extraction of software features.

Due to limitation to access the SRS requirements, compilation of reviews containing software features available on the web can help domain analysts to gauge the idea about the features for a software product prior to development. For example, the reviews in the *toptenreviews.com* provide expert reviews of the top ten software (according to categories), which are compiled periodically. Referring to *toptenreviews.com*, the reviews consist of independent reviews by experts, which emphasised four important issues including hands-on use and evaluation, scoring and ranking, editorial independence, and updates. Hence, the reviews provided for each software in the *toptenreviews.com* are not the firsthand user feedback such as the user reviews from mobile apps used by related works (Guzman & Maalej, 2014), (Iacob & Harrison, 2013)and (Carreno & Windbladh, 2013). For example, (Guzman & Maalej, 2014) evaluated their approach with 32210 reviews from 7 apps, (Iacob & Harrison, 2013)) used 3279 reviews from 161 apps, while Carreno and Windbladh (2013) used data sets that contain 2651 reviews from 3 apps (as shown in Table 6.12). The first hand reviews from user feedback appeared in these three related works are important for developers who want to redesign the features in the product or improve/remove the current features with negative sentiments. However, the RR intention is not the main focus for their research. The use of compiled reviews in our case justifies the needs to focus on extracting software features for reuse.

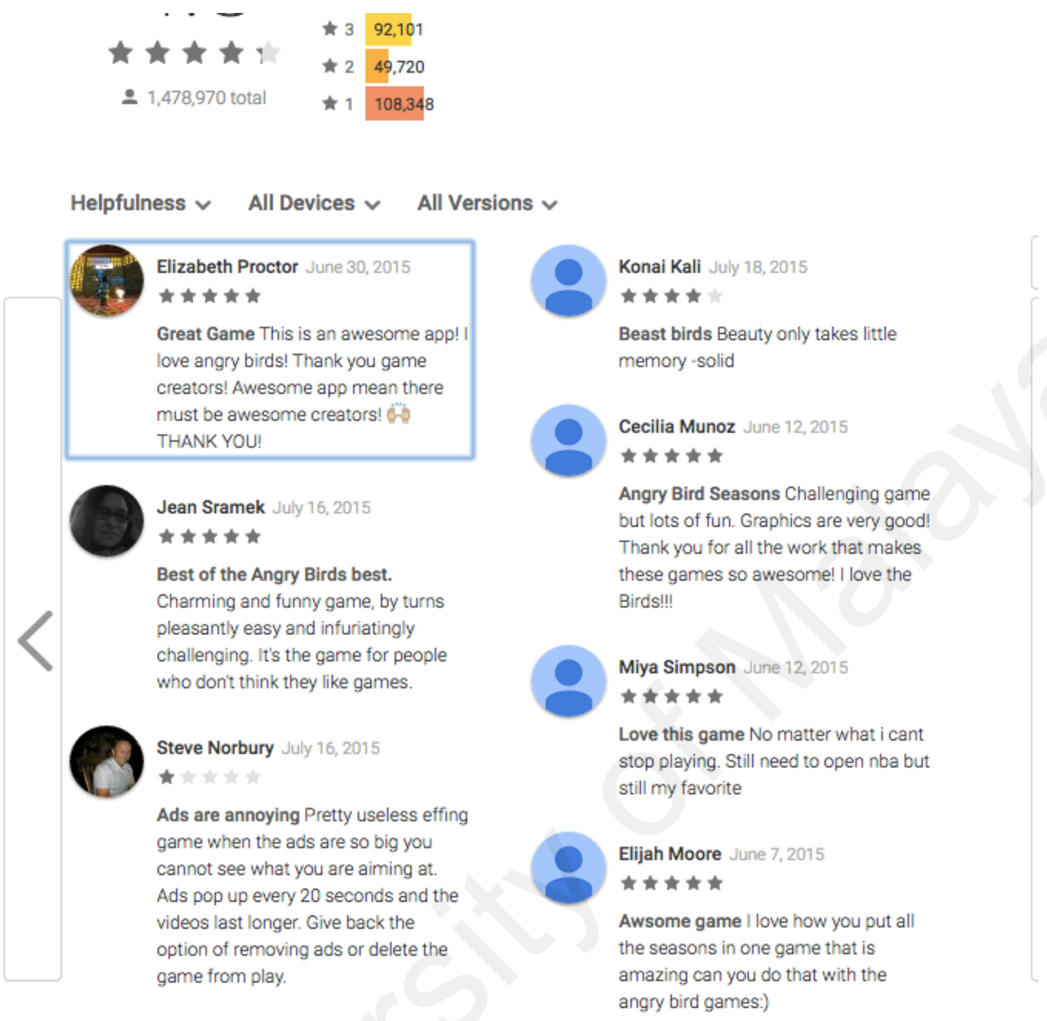
University of Malaya

Table 6-9 Data Sets Comparison

#	Authors	Software Types	No. of reviews	Research purpose
1	Guzman and Maleej (2014)	Mobile apps	32210	Identify new feature requests or improve current features for new releases
2	Iacob and Harrison (2013)	Mobile apps	2651	Identify new feature requests or improve current features for new releases
3	Carreno and Windbladh (2013)	Mobile apps	3279	Identify user comments to aid requirements engineers to revise the requirements for new releases

In this work, the length of the expert reviews extracted varies between the review documents: between the range of 400 words to 1200 words, each. When compared to the user comments used in the related works (Table 6.9), each comment is relatively shorter (of about 30 to 40 words each – see Figure 6.13 for sample mobile apps review, and Figure 6.14 as the sample of expert reviews used as the input to FENL process). This makes the data sets used in this research look comparably smaller compared to the related works. Furthermore, the data used in FENL experiment are reviews that have been compiled by experts, in which most sentiments and user complaints are already minimised.

University of Malaya



Elizabeth Proctor June 30, 2015
Great Game This is an awesome app! I love angry birds! Thank you game creators! Awesome app mean there must be awesome creators! 🙌 THANK YOU!

Jean Sramek July 16, 2015
Best of the Angry Birds best. Charming and funny game, by turns pleasantly easy and infuriatingly challenging. It's the game for people who don't think they like games.

Steve Norbury July 16, 2015
Ads are annoying Pretty useless effing game when the ads are so big you cannot see what you are aiming at. Ads pop up every 20 seconds and the videos last longer. Give back the option of removing ads or delete the game from play.

Konai Kali July 18, 2015
Beast birds Beauty only takes little memory -solid

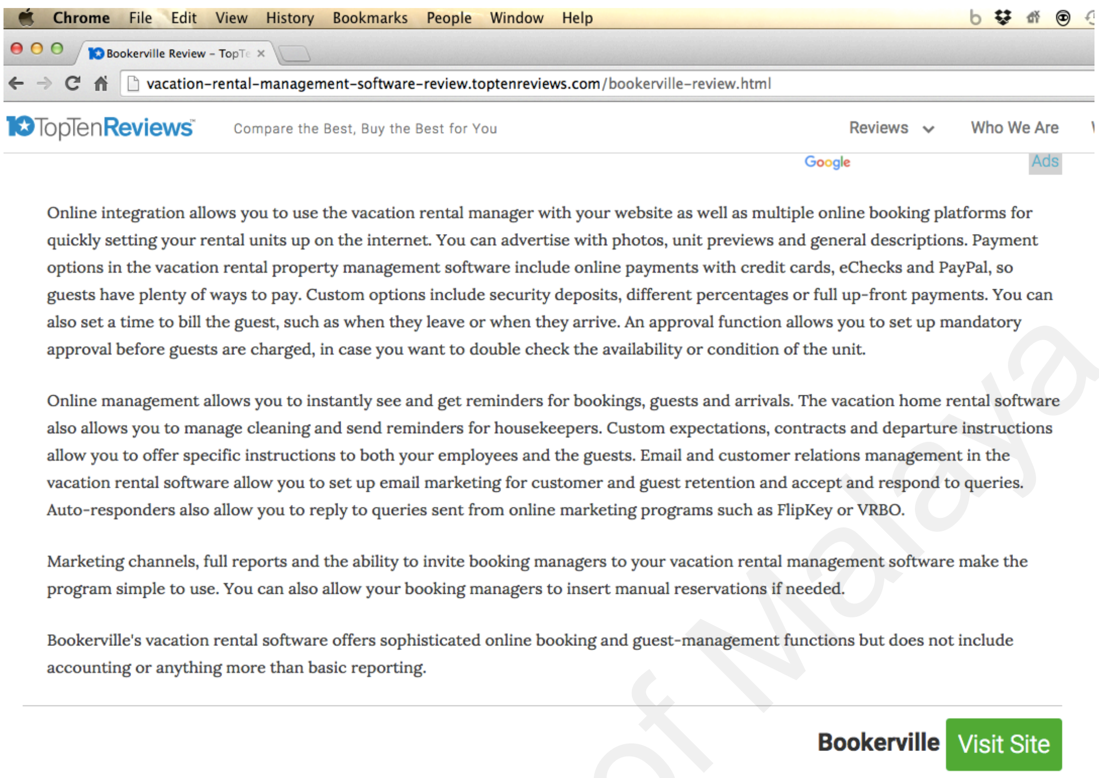
Cecilia Munoz June 12, 2015
Angry Bird Seasons Challenging game but lots of fun. Graphics are very good! Thank you for all the work that makes these games so awesome! I love the Birds!!!

Miya Simpson June 12, 2015
Love this game No matter what i cant stop playing. Still need to open nba but still my favorite

Elijah Moore June 7, 2015
Awsome game I love how you put all the seasons in one game that is amazing can you do that with the angry bird games:)

Figure 6.13 Sample user comments from mobile apps

University of Malaya


Chrome File Edit View History Bookmarks People Window Help
Bookerville Review - TopT
vacation-rental-management-software-review.toptenreviews.com/bookerville-review.html
TopTenReviews
Compare the Best, Buy the Best for You
Reviews
Who We Are
Google
Ads

Online integration allows you to use the vacation rental manager with your website as well as multiple online booking platforms for quickly setting your rental units up on the internet. You can advertise with photos, unit previews and general descriptions. Payment options in the vacation rental property management software include online payments with credit cards, eChecks and PayPal, so guests have plenty of ways to pay. Custom options include security deposits, different percentages or full up-front payments. You can also set a time to bill the guest, such as when they leave or when they arrive. An approval function allows you to set up mandatory approval before guests are charged, in case you want to double check the availability or condition of the unit.

Online management allows you to instantly see and get reminders for bookings, guests and arrivals. The vacation home rental software also allows you to manage cleaning and send reminders for housekeepers. Custom expectations, contracts and departure instructions allow you to offer specific instructions to both your employees and the guests. Email and customer relations management in the vacation rental software allow you to set up email marketing for customer and guest retention and accept and respond to queries. Auto-responders also allow you to reply to queries sent from online marketing programs such as FlipKey or VRBO.

Marketing channels, full reports and the ability to invite booking managers to your vacation rental management software make the program simple to use. You can also allow your booking managers to insert manual reservations if needed.

Bookerville's vacation rental software offers sophisticated online booking and guest-management functions but does not include accounting or anything more than basic reporting.

Figure 6.14 Sample expert reviews from Vacation Management Software

*(*Note that expert reviews are lengthier when compared to user comments)*

6.3.3 Threats to validity of findings

Unstructured reviews were extracted from the web as the input to FENL experiment in this research. These data are raw, unedited, and have not been used in the RR area, as far as this thesis is written. The reviews may also be out dated and new features have been added to the software at the time these reviews are being used in this research experiment. This factor is identified as internal threat to validity for this research experiment.

University of Malaya

In manual extraction, first, teachers and software practitioners manually extracted what constitute as software features from the selected reviews. Second, the extracted features are consolidated to form the ground truth data set. However, the extracted data might not 100% accurately describe the absolute truth for the data set produced, as human judgement can varies and very subjective. This may have some effects on the recall and precision measurement, in which introduces another threat to validity. Although selected teachers are domain subject matter experts in the subject of online learning software assigned to them, they however might not have the technical speciality in terms of identifying software features. Thus, to minimise this threat, features extracted by teachers are consolidated with features extracted by software practitioners. An increase in the quality of extracted features is observed when consolidating the features provided by both parties.

In the experiment, extracted reviews came from four learning categories and three other domains (vacation rental software, online storage software and social networking site). The first four software categories are dedicated for various age groups of users. For example, the software within preschool learning categories are dedicated to children aged six and below, and usually the design of user interface is different to cater to children, which may constitute bigger font sizes or colourful animations to attract the attention of children. When looking at the other categories of reviews such as algebra learning, in which the audience may range from older primary school to teenagers in secondary schools, thus the user interface should be a little more mature when compared to the user interface for the preschool software categories. In this research, the user interface requirements are not the main focus of the extraction. Since the main interest of this research is to extract software functionality (software features) by picking up the combinations of nouns, verbs, and adjective, thus the user interface

University of Malaya

requirements are not considered. Therefore, even though various learning categories for various age groups of users are used as the data set in this experiment, the author believe it will not give impact or difference to the features being extracted.

6.3.4 Integration of the FENL process

Our current implementation of semi-automated process, the FENL, is conducted in a laboratory setting and relies on three applications: Python, MicrosoftExcel, and Matlab. This implementation requires the researcher to have skills in Python programming, MicrosoftExcel, and Matlab programming. More people can benefit from this implementation better when the process is integrated into a single platform.

Summary

In this chapter, the results for FENL approach is presented and discussed. The results obtained from FENL approach is validated by measuring the precision, recall, and F-measure. One Way ANOVA test via SPSS was applied to the average precision, recall and F-Measure to test for their significance. The outcome from applying K-Means clustering on the related features extracted by FENL are also presented, and comparison are made visible for readers. At the end of the chapter, a comprehensive discussions covering threats to validity of the proposed approach is also presented. Next, in Chapter 7, conclusion and discussion on future works will be presented.

University of Malaya

CHAPTER 7: CONCLUSIONS AND FUTURE WORKS

Introduction

In this thesis the problem to reuse natural language requirements has been identified. The process of manual feature extraction in requirements reuse can be arduous, time consuming and error prone on the results. Moreover, there are lack of guidelines or support tools published in this area, which consequently impede the RR practice in Software Product Lines. The assumption was made that an implementation of automated approach could provide a better quality features extracted and expediting time to market (Clements & Northrop, 2002). This is especially true when common features that can be reused are easily extracted from existing requirements artefacts.

7.1 Research Aims and Methods

In order to fulfil the research aim, four research objectives were specified. Table 7.1 summarises the four research objectives and the methods employed in this research for fulfilment of the overall research aim.

Table 7-1 Revisiting Research Aims and Methods

#	Research Objectives	Methods	Presentation
1.	To identify available approaches in feature extraction from natural language requirements for requirements reuse.	Systematic Literature Review	Chapter 2
2.	To explore the current state of practice of requirements reuse	Preliminary Investigation through survey	Chapter 4
3.	To propose a feature extraction process for requirements reuse	Feature extraction process adapting the Natural Language Processing and machine learning techniques is demonstrated through lab	Chapter 5

University of Malaya

		experiment.	
4.	To evaluate the proposed approach	Recall, Precision and F-Measure along with statistical measures were used to evaluate the accuracy of the proposed approach	Chapter 6

In Chapter 2, a comprehensive systematic literature review was provided for the audience to understand the available approaches published in this area. The main conclusion from the Systematic Literature Review in Chapter 2 includes: (1) most related studies use Software Requirements Specifications (SRS) as inputs, but product descriptions, brochures, and user comments are also used due to practical reasons; (2) commonly the outputs from feature extraction process are feature diagrams, clustered requirements, keywords or direct objects; and (3) the extraction process can be divided into four phases: requirements assessment, terms extraction, features identification, and feature model formation.

Additionally, a preliminary survey was conducted to explore the state of practice for requirements reuse and this was presented in Chapter 4. Among important findings from this survey reveals that the unavailability of guidelines or support tools and the poor conditions of existing requirements are among the factors that hinder software practitioners from practicing systematic requirements reuse in Malaysia. Thus, most reuse activities are unplanned and happened on ad hoc basis.

In Chapter 5, the proposed feature extraction process was described. The proposed approach comprise of four phases: Accessing Natural Language Requirements, Terms Extractions, Feature Identifications and Formation of Feature Models. The Latent Semantic

University of Malaya

Analysis and clustering technique from machine learning was used in the FENL implementation.

To evaluate the proposed feature extraction process, a clearly laid out evaluation strategy was presented in Chapter 6. This strategy includes the setting up of manual extraction approach to form the truth data set for the purpose of accuracy comparisons towards the automated approach. Recall, Precision and F-Measure were used to measure the accuracy of the extracted approach against the truth data set. The evaluation results obtained indicated that FENL approach produces Recall results that is significant and performed comparably to manual approach.

7.2 Research Contribution

One of the main contributions of this research is the Systematic Literature Review findings that is useful to addition to the body of knowledge in the area of requirements reuse for software product lines (Bakar et al., 2015). Secondly, the preliminary investigation conducted through survey contributed to the software engineering research pertaining to the state of the practice for requirements reuse in Malaysian context (Bakar & Kasirun, 2014). The details implementation of the feature extraction approach presented in Chapter 5 and its demonstration through lab experiment can be used to guide practitioners to get started with requirements reuse. Besides, the ground truth data set created for this research experiment is made available online in case of future replication is needed. The limitations and threat to validity discussed at the end of Chapter 6 as well can acts as a guidelines for other researchers or practitioners who are interested to perform the feature extractions for requirements reuse problem.

University of Malaya

7.3 Future Work

The feature extraction experiment can be further experimented in a case study settings in industry to determine the applicability of the FENL approach when scale up. Additionally, in the near future, the FENL approach will be integrated in one open source platform, so that more parties can benefit from it.

The second part of FENL that involves grouping extracted features can be further explored. K-Means clustering algorithm is known to have fast convergence, and this property of K-Means might affect the performance of the feature grouping results. Experimenting with different clustering algorithm to determine which clustering algorithm could improve the feature grouping results can be explored in the near future. Another interesting continuation of this work would be to enhance the linguistic tagging of FENL so that it can as well extract the non-functional requirements such as user interface or security features of a software system. For this purpose, investigation on the structure of the non-functional requirements in natural language documents should be carried out.

University of Malaya

University of Malaya

REFERENCES

Abraham, B. Z., & Aguilar, J. C. (2007). Software Component Selection Algorithm Using Intelligent Agents. In *First KES International Symposium, KES-AMSTA 2007, Wroclaw, Poland, May 31– June 1, 2007. Proceedings. Lecture Notes in Computer Science (LNCS).* (pp. 82–91). Berlin: Springer-Verlag Berlin Heiderberg. doi:10.1007/978-3-540-72830-6_9

Acher, M., Cleve, A., Perrouin, G., Heymans, P., Vanbeneden, C., Collet, P., & Lahire, P. (2012). On extracting feature models from product descriptions. *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems - VaMoS '12*, 45–54. doi:10.1145/2110147.2110153

Agresti, W. W. (2011). Software Reuse: Developers' Experiences and Perceptions. *Journal of Software Engineering and Applications*, *04*(01), 48–58. doi:10.4236/jsea.2011.41006

Alves, V., Niu, N., Alves, C., & Valença, G. (2010). Requirements engineering for software product lines: A systematic literature review. *Information and Software Technology*, *52*(8), 806–820. doi:10.1016/j.infsof.2010.03.014

Alves, V., Schwanninger, C., Barbosa, L., Rashid, A., Sawyer, P., Rayson, P., … Rummler, A. (2008). An Exploratory Study of Information Retrieval Techniques in Domain Analysis. *2008 12th International Software Product Line Conference*, 67–76. doi:10.1109/SPLC.2008.18

Babluki, S. (2013). An Efficient Way to Extract the Main Topics from a Sentence. Retrieved from http://thetokenizer.com/2013/05/09/efficient-way-to-extract-the-main-topics-of-a-sentence/

Bagheri, E., & Ensan, F. (2013). Dynamic decision models for staged software product line configuration. *Requirements Engineering*, *19*(2), 187–212. doi:10.1007/s00766-013-0165-8

Bagheri, E., Ensan, F., & Gasevic, D. (2012). Decision Support for the Software Product Line. *Automated Software Engineering*, *19*(3), 335–377. doi:10.1007/s10515-011-0099-7

Bakar, N. H., & Kasirun, Z. M. (2014). Exploring Software Practitioners Perceptions and Experience in Requirements Reuse An Empirical Study in Malaysia. *International Journal of Software Engineering and Technology*, *1*(2), 33–42.

Bakar, N. H., Kasirun, Z. M., & Salleh, N. (2015). Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software*, *106*, 132–149. doi:10.1016/j.jss.2015.05.006

Barreto, F., Benitti, V., & Cezario, R. (2013). Evaluation of a Systematic Approach to Requirements Reuse. *Journal of Universal Computer Science*, *19*(2), 254–280.

Basili, V. R., & Rombach, H. D. (1994). Encyclopedia of Software Engineering. In *Encyclopedia of Software Engineering* (pp. 528–532). John Wiley & Sons.

Benavides, D., Segura, S., & Ruiz-cort, A. (2009). *Automated Analysis of Feature Models : A Detailed Literature Review*. Seville, Spain.

Benavides, D., Segura, S., & Ruiz-Cortes, A. (2010). Automated Analysis of Feature Models

University of Malaya

20 years later: A Literature Review. *Information Systems*, *35*, 615–636. doi:10.1016/j.is.2010.01.001

Bonin, F., Orletta, F. D., Venturi, G., & Montemagni, S. (2010). A Contrastive Approach to Multi – word Term Extraction from Domain Corpora. In N. C. (Conference C. and K. C. and B. M. and J. M. and J. O. and S. P. and M. R. and D. Tapias (Ed.), *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)* (pp. 3222–3229). Vallette, Malta: European Language Resources Association (ELRA).

Boutkova, E., & Houdek, F. (2011). Semi-automatic identification of features in requirement specifications. In *2011 IEEE 19th International Requirements Engineering Conference* (pp. 313–318). Trento, Italy: IEEE. doi:10.1109/RE.2011.6051627

Carreno, L. V. G., & Windbladh, K. (2013). Analysis of User Comments : An Approach for Software Requirements Evolution. In *International Conference of Software Engineering, ICSE 2013* (pp. 582–591). San Francisco, USA: IEEE. doi:10.1109/ICSE.2013.6606604

Casamayor, A., Godoy, D., & Campo, M. (2012). Functional grouping of natural language requirements for assistance in architectural software design. *Knowledge-Based Systems*, *30*, 78–86. doi:10.1016/j.knosys.2011.12.009

Chen, K., Zhang, W., Zhao, H., & Mei, H. (2005). An approach to constructing feature models based on requirements clustering. In *13th IEEE International Conference on Requirements Engineering (RE'05)* (pp. 31–40). La Sorbonne, France: IEEE. doi:10.1109/RE.2005.9

University of Malaya

Chen, L., & Ali Babar, M. (2011). A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology*, *53*(4), 344–362. doi:10.1016/j.infsof.2010.12.006

Chernak, Y. (2012). Requirements Reuse: The State of the Practice. In *2012 IEEE International Conference on Software Science, Technology and Engineering* (pp. 46–53). IEEE. doi:10.1109/SWSTE.2012.12

Clements, P., & Northrop, L. M. (2002). *Software product lines: practices and patterns*. Boston, MA, USA: Addison Wesley Professional.

Cleverdon, C. W. (1970). Evaluation of tests of information retrieval systems. *Journal of Documentation*, *26*, 55–67.

Cui, X., Potok, T. E., & Palathingal, P. (2005). Document clustering using particle swarm optimization. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE* (pp. 185 – 191). Pasadena, California: IEEE. doi:10.1109/SIS.2005.1501621

Czarnecki, K., Helsen, S., & Eisenecker, U. (2004). Staged Configuration Using Feature Models. In R. L. Nord (Ed.), *Third International Conference, SPLC 2004, Boston, MA, USA,* (pp. 266–283). Boston, USA: Springer Berlin Heidelberg. doi:10.1007/978-3-540-28630-1_17

Davis, F. D., Bagozzi, R. P., & Warshaw, P. R. (1989). User Acceptance of Computer Technology: A Comparison of Two Theoretical Models. *Management Science*, *35*(8), 982–1003.

University of Malaya

Davril, J.-M., Delfosse, E., Hariri, N., Acher, M., Cleland-Huang, J., & Heymans, P. (2013). Feature model extraction from large collections of informal product descriptions. *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013*, 290. doi:10.1145/2491411.2491455

Deerwester, S., Dumais, S. T., Furnas, G. W., & Landauer, T. K. (1998). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, *41*(6), 391.

Denger, C., Berry, D. M., & Kamsties, E. (2003). Higher Quality Requirements Specifications through Natural Language Patterns. In *Proceedings of the IEEE International Conference on Software—Science, Technology & Engineering (SwSTE'03)* (pp. 1–11).

Dit, B., Revelle, M., Gethers, M., & Poshyvanyk, D. (2013). Feature location in source code: a taxonomy and survey. *Journal of Software: Evolution and Process*, *25*, 53–95. doi:10.1002/smr.567

Dumitru, H., Gibiec, M., Hariri, N., Cleland-Huang, J., Mobasher, B., Castro-Herrera, C., & Mirakhorli, M. (2011). On-Demand Feature Recommendations Derived from Mining Public Software Repositories. In *International Conference on Software Engineering ICSE 2011* (pp. 181–190). Waikiki, Honolulu, HI, USA: IEEE. doi:10.1145/1985793.1985819

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9-10), 833–859. doi:10.1016/j.infsof.2008.01.006

Eriksson, M., Borstler, J., & Borg, K. (2006). Sotware Product Line Modeling Made Practical:

University of Malaya

An Example from Swedish Defense Industry. *Communications of the ACM*, *49*(12), 49 – 53.

Falessi, D., Cantone, G., & Canfora, G. (2010). A Comprehensive Characterization of NLP Techniques for Identifying Equivalent Requirements. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1–10). Balzano-Bolzen, Italy: ACM. doi:10.1145/1852786.1852810

Falessi, D., Cantone, G., & Canfora, G. (2013). Empirical Principles and an Industrial Case Study in Retrieving Equivalent Requirements via Natural Language Processing Techniques. *IEEE Transaction on Software Engineering*, *39*(1), 18–44.

Faulk, S. R. (2001). Product-Line Requirements Specification (PRS): an Approach and Case Study. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on* (pp. 48–55). IEEE. doi:10.1109/ISRE.2001.948543

Ferrari, A., Spagnolo, G. O., & Dell'Orletta, F. (2013). Mining commonalities and variabilities from natural language documents. In *Proceedings of the 17th International Software Product Line Conference on - SPLC '13* (p. 116). New York, USA: ACM Press. doi:10.1145/2491627.2491634

Finkelstein, A. (1988). Re-use of formatted requirements specifications. *Software Engineering Journal*, *3*(5), 186–197. doi:10.1049/sej.1988.0024

Frakes, W. B., & Fox, C. J. (1995). Sixteen Questions About Software Reuse. *Communications of the ACM*, *38*(6), 75–87.

University of Malaya

Frakes, W. B., & Kang, K. (2005). Software Reuse: Status and Future. *IEEE Transaction on Software Engineering*, *31*(7), 529 – 536.

Galster, M., Weyns, D., Tofan, D., Michalik, B., & Avgeriou, P. (2014). Variability in Software Systems — A Systematic Literature Review. *IEEE Transactions on Software Engineering*, *40*(3), 282–306.

Guzman, E., & Maalej, W. (2014). How Do Users Like This Feature ? A Fine Grained Sentiment Analysis of App Reviews. In *Requirement Engineering Conference 2014* (pp. 153–162). Karskrona, Sweden.

Hand, D., Mannila, H., & Smyth, P. (2001). *Principles of Data Mining* (Adaptive C.). MIT Press.

Hariri, H., Castro-Herera, C., Mirarkholi, M., Cleland-Huang, J., & Mobasher, B. (2013). Supporting Domain Analysis Through Mining and Recommending features from Online Product Listings. *IEEE Transactions on Software Engineering*, *39*(12), 1736–1752.

Hoenkamp, E. (2011). Trading spaces: on the lore and limitations of latent semantic analysis. In *ICTIR'11 Proceedings of the Third international conference on Advances in information retrieval theory* (pp. 40–51).

Huang, A. (2008). Similarity Measures for Text Document Clustering. In *New Zealand Computer Science Research Student Conference (NZCSRSC)* (pp. 1–8). Christchurch.

Hubaux, A., Tun, T. T., & Heymans, P. (2013). Separation of concerns in feature diagram languages. *ACM Computing Surveys*, *45*(4), 1–23. doi:10.1145/2501654.2501665

University of Malaya

Iacob, C., & Harrison, R. (2013). Retrieving and Analyzing Mobile Apps Feature Requests from Online Reviews. In *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference* (pp. 41–44).

IEEE Computer Society (1990). "IEEE Standard Glossary of Software Engineering Terminology". IEEE Standard. (n.d.).

Jiang, H., Nguyen, T. N., & Chen, I. (2008). Incremental Latent Semantic Indexing for Effective , Automatic Traceability Link Evolution Management. In *In Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering* (pp. 59–68). L'Aquila, Italy: IEEE Computer Society.

Jørgensen, M., & Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering*, *33*(1), 33–53.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). *Feasibility Study Feature-Oriented Domain Analysis (FODA).* Pittsburgh, PA. doi:10.1080/10629360701306050

Kang, K., Cohen, S., Hess, J., Novak, W., & Peterson, A. (1990). *Feature Oriented Domain Analysis (FODA) Feasibility Study*. Pittsburgh, PA.

Khan, K., Baharudin, B., & Khan, A. (2014). Identifying Product Features from Customer Reviews Using Hybrid Patterns. *International Arab Journal of IT*, *11*(3), 281–286.

Khurum, M., & Gorschek, T. (2009). A systematic review of domain analysis solutions for product lines. *Journal of Systems and Software*, *82*(12), 1982–2003.

University of Malaya

doi:10.1016/j.jss.2009.06.048

Kitchenham, B. A., & Charters, S. (2007). *Procedures for Performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report version 2.3, EBSE-2007-01*. Keele, UK. doi:10.1.1.122.3308

Knethen, A. Von, Paech, B., Kiedaisch, F., Houdek, F., Kaiserslautern, D.-, Ulm, D.-, & Ag, D. (2002). Systematic Requirements Recycling through Abstraction and Traceability. In *Requirements Engineering* (pp. 273–281).

Kobielus, J. (2014). The Ground Truth in Agile Machine Learning. *IBM Big Data Analytics*. Retrieved from http://www.ibmbigdatahub.com/blog/ground-truth-agile-machine-learning

Krueger, C. (1992). Software Reuse. *ACM Comput. Surv.*, *24*(2), 131–183.

Krueger, C. (2001). Easing the transition to software mass customization. In *International Workshop on Product Family Engineering* (pp. 282–293). Bilbao, Spain.

Krueger, C. (2002). Eliminating the Adoption Barrier. *IEEE Software*, (August), 29–31.

Kumaki, K., Tsuchiya, R., Washizaki, H., & Fukazawa, Y. (2012). Supporting commonality and variability analysis of requirements and structural models. In *Proceedings of the 16th International Software Product Line Conference on - SPLC '12* (pp. 115–118). New York, USA: ACM Press. doi:10.1145/2364412.2364431

Lam, W., McDermid, J. A., & Vickers, A. J. (1997). Ten steps towards systematic

University of Malaya

requirements reuse. *Requirements Engineering*, *2*(2), 102–113. doi:10.1007/BF02813029

Leedy, P. D., & Ormrod, J. E. (2013). *Practical Research Planning and Design* (11th ed.). Boston, MA, USA: Pearson Education Inc.

Lisboa, L. B., Garcia, V. C., Lucrédio, D., De Almeida, E. S., De Lemos Meira, S. R., & De Mattos Fortes, R. P. (2010). A systematic review of domain analysis tools. *Information and Software Technology*, *52*(1), 1–13. doi:10.1016/j.infsof.2009.05.001

Maiden, N. A., & Sutcliffe, A. G. (1992). Exploiting Usable Specifications Through Analogy. *Communications of the ACM*, *35*(4), 55–64. doi:10.1145/129852.129857

Maletic, J. I., & Valluri, N. (1999). Automatic software clustering via Latent Semantic Analysis. In *14th IEEE International Conference on Automated Software Engineering* (pp. 251–254). Cocoa Beach Florida: IEEE Comput. Soc. doi:10.1109/ASE.1999.802296

Manning, C., Raghavan, P., & H. Schtze. (2008). *Introduction to Information Retrieval*. Cambridge Univ. Press.

Marcus, A., & Maletic, J. I. (2003). Recovering documentation-to-source-code traceability links using latent semantic indexing. In *25th International Conference on Software Engineering, 2003. Proceedings.* (pp. 125–135). Ieee. doi:10.1109/ICSE.2003.1201194

Massonet, P., & Lamsweerde, A. Van. (1997). Analogical Reuse of Requirements Frameworks. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997.* (pp. 26–37).

University of Malaya

Mellarkod, V., Appan, R., Jones, D. R., & Sherif, K. (2007). A multi-level analysis of factors affecting software developers' intention to reuse software assets: An empirical investigation. *Information & Management*, *44*(7), 613–625. doi:10.1016/j.im.2007.03.006

Metzger, A., & Pohl, K. (2014). Software Product Line Engineering and Variability Management : Achievements and Challenges. In *FOSE* (pp. 70–84). Hyderabad, India.

Meyer, M. H., & Lehnerd, A. P. (1997). *The Power of Product Platform*. New York: Free Press.

Monzon, A. (2008). A Practical Approach to Requirements Reuse in Product Families of On-Board Systems. In *16th IEEE International Requirements Engineering Conference* (pp. 223–228). Barcelona,Catalunya, Spain: IEEE. doi:10.1109/RE.2008.19

Moros, B., Toval, A., Rosique, F., & Sánchez, P. (2012). Transforming and Tracing Reused Requirements Models to Home Automation Models. *Information and Software Technology*. doi:10.1016/j.infsof.2012.12.003

Mu, Y., Wang, Y., & Guo, J. (2009). Extracting Software Functional Requirements from Free Text Documents. In *2009 International Conference on Information and Multimedia Technology* (pp. 194–198). Ieee. doi:10.1109/ICIMT.2009.47

Neighbors, J. M. (1984). The Draco Approach to Constructing Software from Reusable Components. *IEEE Transactions on Software Engineering*, *SE-10*(5), 564–574. doi:10.1109/TSE.1984.5010280

Neill, C. J., & Laplante, P. A. (2003). Requirements Engineering : The State of the Practice.

University of Malaya

IEEE Software, *20*(6), 40–45. doi:10.1109/MS.2003.1241365

Nicolás, J., & Toval, A. (2009). On the generation of requirements specifications from software engineering models: A systematic literature review. *Information and Software Technology*, *51*(9), 1291–1307. doi:10.1016/j.infsof.2009.04.001

Niu, N., & Easterbrook, S. (2008). Extracting and Modeling Product Line Functional Requirements. *2008 16th IEEE International Requirements Engineering Conference*, 155–164. doi:10.1109/RE.2008.49

Niu, N., Savolainen, J., Niu, Z., Jin, M., & Cheng, J.-R. C. (2013). A Systems Approach to Product Line Requirements Reuse. *IEEE Systems Journal*, 1–10. doi:10.1109/JSYST.2013.2260092

Northrop, L. M., & Clements, P. C. (n.d.). A Framework for Software Product Line Practice, Version 5.0. *Software Engineering Institute, Carnegie Mellon University*. Retrieved April 25, 2015, from http://www.sei.cmu.edu/productlines/frame_report/index.html

Oliveira, T. C., Alencar, P., & Cowan, D. (2011). ReuseTool—An extensible tool support for object-oriented framework reuse. *Journal of Systems and Software*, *84*(12), 2234–2252. doi:10.1016/j.jss.2011.06.030

Paredes, C., & Fiadeiro, J. L. (1995). Reuse of Requirements and Specifications - A Formal Framework. In *Symposium on Software Reusability* (pp. 263–266). Seattle, WA USA: ACM. doi:0-89791 -739 -1/95/0004

Petticrew, M., & Roberts, H. (2006). *Systematic Reviews in the Social Sciences: A Practical*

University of Malaya

Guide. Maryland USA: Blackwell Publishing.

Pohl, K., Bockle, G., & Van der Linden, F. (2005). *Software Product Line Engineering*. Berlin/Heidelberg: Springer-Verlag. doi:10.1007/3-540-28901-1

Poshyvanyk, D., Gethers, M., & Marcus, A. (2012). Concept location using formal concept analysis and information retrieval. *ACM Transactions on Software Engineering and Methodology*, *21*(4), 1–34. doi:10.1145/2377656.2377660

Renault, S., Mendez-Bonilla, O., Franch, X., & Quer, C. (2009). PABRE : Pattern-Based Requirements Elicitation. In *Third International Conference on Research Challenges in Information Science, 2009. RCIS 2009.* (pp. 81–92). doi:10.1109/RCIS.2009.5089271

Robinson, W. N., & Woo, H. G. (2004). Finding Reusable UML Sequence Diagrams Automatically. *IEEE Software*, *21*(5), 60–67. doi:10.1109/MS.2004.1331304

Salleh, N., Mendes, E., & Grundy, J. (2011). Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. *IEEE Transaction on Software Engineering*, *37*(4), 509–525. doi:doi: 10.1109/TSE.2010.59

Salton, G., & Buckley, C. (1988). Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, *24*(5), 513 – 523. Retrieved from http://comminfo.rutgers.edu/~muresan/IR/Docs/Articles/ipmSalton1988.pdf

Salton, G., Wong, A., & Yang, C. S. (1975). Vector Space Model for Automatic Indexing. *Communication of ACM*, *18*(11).

University of Malaya

Shaw, M. (2002). What Makes Good Research in Software Engineering? *International Journal of Software Tools for Technology Transfer*, *4*(1), 1–7.

Shull, F., Singer, J, & Sjoberg, D. I. K. (2008). *Guide to Advanced Empirical Software Engineering*. London: Springer-Verlag London.

Slyngstad, O. P. N., Gupta, A., Conradi, R., Mohagheghi, P., Rønneberg, H., & Landre, E. (2006). An Empirical Study of Developers Views on Software Reuse in Statoil ASA. In *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering* (pp. 242–251). Rio de Janeiro, Brazil: ACM. doi:10.1145/1159733.1159770

Solemon, B., Shahibudin, S., & Abdul Ghani, A. A. (2008). Requirements Engineering Problems in 63 Software Companies in Malaysia Badariah Solemon. In *International Symposium on Information Technology, ITSim 2008* (pp. 1–6). IEEE. doi:10.1109/ITSIM.2008.4631911

Weston, N., Chitchyan, R., & Rashid, A. (2009). A Framework for Constructing Semantically Composable Feature Models from Natural Language Requirements. In *Proceeding of the 13th Software Product Lines Conference* (pp. 211–220). San Francisco, California, USA: Carnegie Mellon University.

Wohlin, C., & Prikladnicki, R. (2013). Systematic literature reviews in software engineering. *Information and Software Technology*, *55*(6), 919–920. doi:10.1016/j.infsof.2013.02.002

Yu, Y., Wang, H., Yin, G., & Liu, B. (2013). Mining and recommending software features across multiple web repositories. In *Proceedings of the 5th Asia-Pacific Symposium on*

University of Malaya

Internetware (pp. 1–9). New York, USA: ACM Press. doi:10.1145/2532443.2532453

University of Malaya

LIST OF PUBLICATIONS AND PAPERS PRESENTED

1. Bakar, N. H., Kasirun, Z. M., & Salleh, N. (2015). Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software*, *106*, 132–149. doi:10.1016/j.jss.2015.05.006
2. Bakar, N. H., Kasirun, Z. M., Salleh, N., & Jalab, H. A. (2016). Extracting features from online software reviews to aid requirements reuse. Applied Soft Computing Journal (ISI cited – Q1), Accepted 27th July 2016. http://dx.doi.org/10.1016/j.asoc.2016.07.048
3. Bakar, N. H., Kasirun, Z. M., & Jalab, H. A. (2014). Towards Requirements Reuse : Identifying Similar Requirements with Latent Semantic Analysis and Clustering Algorithms. In *Proc. of the Second Intl. Conf. on Advances In Computing, Communication and Information Technology- CCIT 2014*. (pp. 19–24). Birmingham, United Kingdom: Seek Digital Library - the REID. doi:10.15224/ 978-1-63248-051-4-20
4. Bakar, N. H., & Kasirun, Z. M. (2013). Empirical Survey on Requirements Reuse Practice Among Software Practitioners in Malaysia. In *MySEC2013*).
5. Bakar, N. H., & Kasirun, Z. M. (2014). Exploring Software Practitioners Perceptions and Experience in Requirements Reuse An Survey in Malaysia. *International Journal of Software Engineering and Technology*, *1*(2). Retrieved from http://se.cs.utm.my/ijset/index.php/ijset/article/view/26/20

University of Malaya

6. Bakar, N. H. (2013). Latent Semantic Analysis and Particle Swarm Optimization for Requirements Reuse in Software Product Line : Research Plan. *Doctoral Symposium in conjunction with Software Product Line Conference, SPLC 2013*. Tokyo, Japan.
7. Bakar, N. H., Kasirun, Z. M., Salleh, N., & Jalab, H. A. (2015). Terms Extractions : An Approach for Requirements Reuse. In *2nd International Conference on Information Science and Security (ICISS)*.

University of Malaya

APPENDIX A – LIST OF PRIMARY STUDIES SELECTED IN THE SYSTEMATIC LITERATURE REVIEW

ID	Author	Paper Title	Venue / Source
S1	Kumaki K., Washizaki, H., & Fukazawa, Y.	Supporting commonality and variability analysis of requirements and structural models	SPLC 12: 115 – 118 (Kumaki et al., 2012)
S2	Niu, N., Savolainen, J., Niu, Z., Jin, M., & Cheng, J.-R. C.	A systems approach to product line requirements reuse	IEEE Systems Journal: 1-10 (Niu et al., 2013)
S3	Ferrari A., Spagnolo, G., & Dell Orletta	Mining commonalities and variabilities from natural language documents	SPLC 13: 116 – 120 (Ferrari et al., 2013)
S4	Alves, V., Schwanninger, C., Barbosa, L., Rashid, A., Sawyer, P., Rayson, P., Pohl, C., & Rummler, A.	An exploratory study of information retrieval techniques in domain analysis	SPLC 08: 67 – 76 (Alves et al., 2008)
S5	Weston, N., Chitchyan, R., & Rashid, A.	A framework for constructing semantically composable feature models from natural language requirements.	SPLC 09: 211 – 220 (Weston et al., 2009)
S6	Chen, K., Zhang, W., Zhao, H., & Mei, H.	An approach to constructing feature models based on requirements clustering	RE 2005: 31 – 40 (K. Chen et al., 2005)
S7	Acher, M., Cleve, A., Perrouin, G., Heymans, P., Vanbeneden, C., Collet, P., & Lahire, P.	On extracting feature models from product descriptions	VaMOS 12: 45 – 54 (Acher et al., 2012)
S8	Hariri, N., Castro-Herrera, C., Mirakhorli, M., Cleland-Huang, J., & Mobasher, B.	Supporting domain analysis through mining and recommending features from online product listings	IEEE Trans. Software Engineering, *39*(12), December 2013. (Hariri et al., 2013)
S9	Mu, Y., Wang, Y., & Guo, J.	Extracting software functional requirements from free text documents	International Conference on Information and Multimedia Technology, 2009 (ICIMT '09). (Mu et al., 2009)
S10	Yu, Y., Wang, H., Yin, G., & Liu, B.	Mining and recommending software features across multiple web repositories	Internetware, October 2013. (Yu et al., 2013)

University of Malaya

S11	Bagheri, B., Ensan F., & Gasevic, D.	Decision support for the software product line domain engineering lifecycle	Automated Software Engineering, September 2012, *19*(3), 335-377. (Bagheri et al., 2012)
S12	Boutkova, E., & F. Houdek	Semi-automatic identification of features in requirement specifications	RE 2011: 313 – 318 (Boutkova & Houdek, 2011)
S13	Guzman, E., & Maalej, W.	How do users like this feature? A fine grained sentiment analysis of app reviews	RE 2014: 153 – 162 (Guzman & Maalej, 2014)

University of Malaya

APPENDIX B: INPUT (TYPES OF REQUIREMENTS) AND OUTPUT (FEATURES) FOR SELECTED STUDIES IN SYSTEMATIC LITERATURE REVIEW

Classification Dimensions for Requirements (Input) and Feature (Output) Types										
Selected Studies		Input					Output			
		Requirements / SRS Docs	Internal Documentation	Product Descriptions	Product Brochures	User Comments	Features (Keywords)	Feature Trees/Model	Verb Phrase/Direct Objects	Clustered Requirements
S1	[Kumaki, '12]	•								•
S2	[NanNiu, '08]	•							•	
S3	[Ferrari, '13]				•		•			
S4	[V. Alves, '08]	•						•		
S5	[Weston, '09]	•						•		
S6	[Chen, '05]	•						•		
S7	[Archer, '12]			•				•		
S8	[Hariri, '13]			•				•		
S9	[Yunhe, '09]	•							•	
S10	[Yu Wang, '13]			•						•
S11	[Bagheri, '12]	•								•
S12	[Boutkova, '11]		•				•			
S13	[Guzman, '14]					•				

University of Malaya

APPENDIX C: DATA EXTRACTION FORM

Study Info Data	
Study ID (S #)	
Date of the extraction	
Paper Title	
Author(s)	
Publication Type	
Name of the tools (if any)	
Source	
Answers to elements in RQs	
RQ1: What approaches are available to extract features from NL requirements?	Types of Input: Types of Output:
RQ1.1: How were the commonality and variability addressed? Which technique was used?	NLP/Information Theory Machine Learning/Data mining
RQ1.2: Availability of support tools, Automated/Semi-automated	Automated Semi-auto Manual
RQ2: Evaluation being performed	Context: Academia/Industry Procedure: Experiment/Case Study/Other: ________________________ Measure Used: Recall/Precision/F-Measure/Other: _______________
RQ 2.2: Domain Application	

University of Malaya

APPENDIX D: QUALITY ASSESSMENT CHECKLIST

Item	Answer
QA1: Was the article refereed? (Leedy & Ormrod, 2013)	Yes/No
QA2: Was there a clear statement of the aims of the research? (Dybå & Dingsøyr, 2008)	Yes/No/ Partially
QA3: Is there an adequate description of the context in which the research was carried out? (Dybå & Dingsøyr, 2008) For example, the problems that lead to the research are clearly stated, descriptions of research methodology used, study participants, etc.	Yes/No/ Partially
QA4: Was the data collection done very well? For example, did the evaluation done on proposed approach answer the research questions, did the paper provide a thorough discussion of the collected results? (Dybå & Dingsøyr, 2008)	Yes/No/ Partially
QA5: Were the testing results rigorously analysed? (Petticrew & Roberts, 2006) For example, are there any software metrics provided in evaluating the test results, is there any threat to validity being presented in the study, etc.	Yes/No/ Partially
QA6: Any practitioner-based guidelines on requirements reuse being produced? Lam et al. suggested that practitioners' guidelines including producing explicit documentation is important to prevent reuse misuse. (Lam et al., 1997)	Yes/No/ Partially
**Scores: Yes[1], No[0], Partially[0.5]*	

University of Malaya

APPENDIX E – COVER LETTER FOR PILOT SURVEY

6th March 2013

Dear Y.Bhg. Dato'/Prof./Assoc. Prof. Dr./Dr./Sir/Madam/Ms

Pilot Study: Requirements Reuse Practice Among Software Engineering Community In Malaysia

This pilot study is being conducted by a PHD candidate, Noor Hasrina Bakar – WHA110041 under supervision of Assoc. Prof Dr. Zarinah Kasirun from the Department of Software Engineering, FSKTM of University of Malaya. The purpose of this study is to investigate the current practice of requirements reuse among software engineering community in Malaysia. In this study, you will be asked to complete a paper based survey. The survey should take no longer than 15 minutes to complete.

You are selected because we need your software engineering expert opinion in confirming the questions specified in the survey. Please indicate your comments (if any) to each question pertaining to any related issue. Your participation in this pilot study is a crucial last step before the revised survey is distributed to the selected software practitioners in Malaysia.

We recognise the additional effort needed to respond to the revised survey, and we greatly appreciate your time and assistance. While you will not experience any direct benefits from participation, information collected in this pilot study may benefit the researcher to clarify any ambiguity from this survey questions.

We hope to receive your feedback no later than 22/3/2013.

If you have any questions regarding the survey or this research project in general, please contact Noor Hasrina Bakar at noor.hasrina@gmail.com / noorhasrina@siswa.um.edu.my / 012692-7506 or her advisor, Assoc. Prof Dr. Zarinah Kasirun at zarinahmk@um.edu.my.

Noor Hasrina Bakar,
PHD Candidate, Department of Software Engineering,
FSKTM University Malaya

University of Malaya

APPENDIX F – SURVEY QUESTIONNAIRE FOR EXPLORING THE STATE OF REQUIREMENTS REUSE PRACTICE IN MALAYSIA

SURVEY ON REQUIREMENTS REUSE PRACTICE AMONG SOFTWARE ENGINEERING COMMUNITY IN MALAYSIA

Thank you for agreeing to participate in this survey. This survey is conducted by the Department of Software Engineering, Faculty of Computer Science & IT, University of Malaya. We are interested in knowing your opinion about various issues regarding reuse of requirements in software development. Your thoughtful answers will provide important input to our research. It is very important to us that you answer all the questions to the best of your ability.

All information provided by you will be kept confidential and used for the purpose of research only. The overall purpose of this survey is to investigate the current practice of requirements reuse among software practitioners in Malaysia. We are aiming to:

1. collect the information regarding participants perceptions about reusing requirements in software development
2. gather information about current requirements reuse practice in participant's organisations, including tools support and related problems faced.

This survey consisted of 3 parts. Questions in PART A aim to collect demographic info / background of the survey respondents. Questions in PART B seek to understand respondent's perceptions about various requirements reuse related issues in software development. In PART C we would like to know your experience regarding reuse of requirements in your latest project.

PART A: DEMOGRAPHIC

Tell us about yourself:

1. Gender:

- ☐ Male
- ☐ Female

2. Age:

- ☐ 25 - 30
- ☐ 31- 40

University of Malaya

- ☐ 42-50
- ☐ 51-55

3. Educational Qualifications:

- ☐ Certificate
- ☐ Diploma
- ☐ Bachelor’s Degree
- ☐ Masters Degree
- ☐ PHD

4. Your job title:

- ☐ Project Manager
- ☐ Software Engineer
- ☐ Requirements Manager
- ☐ Business Analysts
- ☐ Software Engineering Expert / Consultant
- ☐ Other (please specify): ____

5. How long have you been with the current job?

- ☐ < 1 year
- ☐ Between 1 to 3 years
- ☐ Between 3 to 5 years
- ☐ **More than 5 years**

6. How many years of experience do you have in the following fields?

Requirements Engineering:

- ☐ 0 year
- ☐ < 1 year
- ☐ 1 – 3 years
- ☐ 3 – 5 years
- ☐ > 5 years

Project Management

- ☐ 0 year
- ☐ < 1 year
- ☐ 1 – 3 years
- ☐ 3 – 5 years
- ☐ 5 years

University of Malaya

Tell us about your organisation:

7. How big is your development team?

- ☐ 1 to 5 people
- ☐ 6 to 10 people
- ☐ 11 to 20 people
- ☐ 21 to 50 people

8. Which industry group that best describes your organisation? (you may choose more than one category)

- ☐ Software Development House
- ☐ IT Consultancy
- ☐ Financial / Banking
- ☐ Education
- ☐ Automobiles
- ☐ Business Services
- ☐ Health & Medical
- ☐ Energy
- ☐ Telecommunications
- ☐ Travel and Tourism
- ☐ Aerospace
- ☐ Other (please specify) ____

9. How does software requirements being presented in your organisation? (you may choose more than one category)

- ☐ Features
- ☐ Use Cases
- ☐ UML Diagrams
- ☐ Textual
- ☐ Other (please specify): ____

10. Your organisation is a:

- ☐ Government
- ☐ Semi-government
- ☐ Private Sector

11. Your organisation's span of operation:

University of Malaya

☐ Domestic
☐ Multinational

12. What kind of software development process model used in your organisation?

☐ Waterfall
☐ V-Model
☐ Incremental Process Model
☐ Agile
☐ Prototyping
☐ Other (please specify): ____

PART B: Perceptions about Requirements Reuse

In this part, we would like to understand your perception about various reuse-related issues.

Answer these questions on a scale from 1 to 7.
1 is Strongly Disagree. 7 is Strongly Agree.

	Reuse: Intention	1	2	3	4	5	6	7
1	Assuming I had access to reusable requirements, I intend to use them when developing future applications.							
2	Given that I have access to reusable requirements, I predict that I would make use of them when developing future applications.							
3	I intend to increase my use of reusable requirements in the future development of application.							

	Reuse: Benefits	1	2	3	4	5	6	7
1	Reusing existing requirements improves my job performance.							
2	Reusing existing requirements in my job increases my team's productivity.							
3	I believe, reusing existing requirements decreases maintenance costs at the later development stage.							
4	Requirements Reuse is not important in our organisation at this moment.							

University of Malaya

	Reuse: Ease of Use	1	2	3	4	5	6	7
1	I feel reusing requirements does not require a lot of mental effort.							
2	It is easier for me to understand the reusable requirement documents, as compared to developing new requirements when dealing with software solution from similar domain.							

	Reuse: Infrastructure My organisation has appropriate standardised process for:	1	2	3	4	5	6	7
1	Analysing a domain.							
2	Developing reusable codes and designs.							
3	Developing reusable requirements.							
4	Managing reusable requirements.							

	Reuse: Support Tools My organisation provide relevant training and support tools for:	1	2	3	4	5	6	7
1	Analysing a domain.							
2	Developing reusable codes and designs.							
3	Developing reusable requirements.							
4	Managing reusable requirements.							

PART C: Experience on Reusing Requirements

	Reuse of Requirements in the latest project	1	2	3	4	5	6	7
1	In our latest project, we somehow reuse the existing requirements.							
2	In our latest project, we reuse existing requirements from completed projects because:							

University of Malaya

	a. We are involved in Software Product Line development; thus reuse are systematically planned (SPL[21]).							
	b. It is just happened that the new product requirements requested by our customer are very similar to the one we have produced before (Clone and Own).							
	c. we are maintaining the prior releases, therefore the old documentations are now reused and improved (Software Maintenance).							
3	We never reuse existing requirements in our software development. Instead we always start new requirements engineering process for new projects.							

4	Requirements Reuse is Not Invented Here because:	1	2	3	4	5	6	7
	a. The project team did not feel that reuse is important and worth the effort.							
	b. The project management did not support requirements reuse.							
	c. The requirements developed in previous releases were incomplete (or do not exist), so it is impossible to reuse them.							
	d. The existing requirements were poorly structured, so it is difficult to identify which requirements can be reused.							
	e. The existing requirements were poorly structured, so it is difficult to identify which requirements can be reused.							
	f. Other reason (please specify):							

	Requirements Reuse Process in your organisation					
1	In our organisation, requirements, test cases or other assets within	(0%)	(25%)	(50%)	(75%)	(100%)

[21] The term SPL used here refers to software engineering methods, tools and techniques for creating a collection of similar software systems (family of systems) from a shared set of software assets using a common means of production, that were systematically planned for long term investments (Carnegie Mellon (SEI), 2003).

University of Malaya

	an average project are duplicated or commonly shared from other projects for about:							
		1	**2**	**3**	**4**	**5**	**6**	**7**
2	**In our software development, we follow a practitioner guidelines to reuse existing software requirements.**							

	Experience with Requirements Reuse tools (if any)	**Yes**			**No**			
1	**We use a support tool to assist our requirements reuse process.**							
2	**Please provide the name for the support tool used:**							
3	**We have problems with the requirements reuse tools used in our organisation.**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
4	**Write down any problem faced when using the requirements reuse tools at your organisation.**							
5	**Based on your experience working with requirements reuse tools, suggest the improvements needed to facilitate a better requirements reuse process in future.**							

University of Malaya

COMMENTS AND SUGGESTIONS:

Please provide your comments and suggestions on how we can improve the accuracy and readability of this survey. Your feedback is highly appreciated.

~~~THANK YOU~~~

University of Malaya
~~~

APPENDIX G – POST HOC TEST FOR ONE WAY ANOVA

Multiple Comparisons

Tukey HSD

Dependent Variable	(I) Method	(J) Method	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Recall	Manual	Simple	42.84667*	5.69892	.000	27.4062	58.2871
		Noun Phrase	23.52444*	5.69892	.001	8.0840	38.9649
		FENL	15.98778*	5.69892	.040	.5473	31.4282
	Simple	Manual	-42.84667*	5.69892	.000	-58.2871	-27.4062
		Noun Phrase	-19.32222*	5.69892	.010	-34.7627	-3.8818
		FENL	-26.85889*	5.69892	.000	-42.2993	-11.4185
	Noun Phrase	Manual	-23.52444*	5.69892	.001	-38.9649	-8.0840
		Simple	19.32222*	5.69892	.010	3.8818	34.7627
		FENL	-7.53667	5.69892	.556	-22.9771	7.9038
	FENL	Manual	-15.98778*	5.69892	.040	-31.4282	-.5473
		Simple	26.85889*	5.69892	.000	11.4185	42.2993
		Noun Phrase	7.53667	5.69892	.556	-7.9038	22.9771
Precision	Manual	Simple	27.78444*	6.76518	.001	9.4551	46.1138
		Noun Phrase	38.30222*	6.76518	.000	19.9729	56.6315
		FENL	30.36889*	6.76518	.000	12.0396	48.6982
	Simple	Manual	-27.78444*	6.76518	.001	-46.1138	-9.4551
		Noun Phrase	10.51778	6.76518	.418	-7.8115	28.8471
		FENL	2.58444	6.76518	.981	-15.7449	20.9138
	Noun Phrase	Manual	-38.30222*	6.76518	.000	-56.6315	-19.9729
		Simple	-10.51778	6.76518	.418	-28.8471	7.8115
		FENL	-7.93333	6.76518	.648	-26.2627	10.3960
	FENL	Manual	-30.36889*	6.76518	.000	-48.6982	-12.0396
		Simple	-2.58444	6.76518	.981	-20.9138	15.7449
		Noun Phrase	7.93333	6.76518	.648	-10.3960	26.2627
FMeasure	Manual	Simple	36.34778*	5.48891	.000	21.4763	51.2192
		Noun Phrase	32.76111*	5.48891	.000	17.8897	47.6325
		FENL	25.79778*	5.48891	.000	10.9263	40.6692
	Simple	Manual	-36.34778*	5.48891	.000	-51.2192	-21.4763
		Noun Phrase	-3.58667	5.48891	.914	-18.4581	11.2848
		FENL	-10.55000	5.48891	.239	-25.4214	4.3214
	Noun Phrase	Manual	-32.76111*	5.48891	.000	-47.6325	-17.8897
		Simple	3.58667	5.48891	.914	-11.2848	18.4581
		FENL	-6.96333	5.48891	.589	-21.8348	7.9081
	FENL	Manual	-25.79778*	5.48891	.000	-40.6692	-10.9263
		Simple	10.55000	5.48891	.239	-4.3214	25.4214
		Noun Phrase	6.96333	5.48891	.589	-7.9081	21.8348

University of Malaya

University of Malaya