The Journal of Systems and Software 106 (2015) 132–149

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review

Noor Hasrina Bakar[a,b,*], Zarinah M. Kasirun[a], Norsaremah Salleh[c]

[a] Department of Software Engineering, Faculty of Computer Science & Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
[b] Department of ICT, Centre for Foundation Studies, International Islamic University Malaysia, 46350 Petaling Jaya Selangor, Malaysia
[c] Department of Computer Science, Kulliyyah of Information & Communication Technology, International Islamic University Malaysia, 53100 Jalan Gombak, Kuala Lumpur, Malaysia

ARTICLE INFO

Article history:
Received 17 April 2014
Revised 30 April 2015
Accepted 3 May 2015
Available online 9 May 2015

Keywords:
Feature extractions
Requirements reuse
Software product lines
Natural language requirements
Systematic literature review

ABSTRACT

Requirements for implemented system can be extracted and reused for a production of a new similar system. Extraction of common and variable features from requirements leverages the benefits of the software product lines engineering (SPLE). Although various approaches have been proposed in feature extractions from natural language (NL) requirements, no related literature review has been published to date for this topic. This paper provides a systematic literature review (SLR) of the state-of-the-art approaches in feature extractions from NL requirements for reuse in SPLE. We have included 13 studies in our synthesis of evidence and the results showed that hybrid natural language processing approaches were found to be in common for overall feature extraction process. A mixture of automated and semi-automated feature clustering approaches from data mining and information retrieval were also used to group common features, with only some approaches coming with support tools. However, most of the support tools proposed in the selected studies were not made available publicly and thus making it hard for practitioners' adoption. As for the evaluation, this SLR reveals that not all studies employed software metrics as ways to validate experiments and case studies. Finally, the quality assessment conducted confirms that practitioners' guidelines were absent in the selected studies.

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 and Clements, 2015). 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 and Northrop, 2002). When RR was planned systematically in the SPLE context, several studies (Eriksson et al., 2006; Monzon, 2008; Moros et al., 2013; Von 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 and Easterbrook, 2008), error-prone (Ferrari et al., 2013), and labour-intensive (Boutkova and Houdek, 2011) process, especially when dealing with large requirements.

In the following subsections, we will briefly describe the terms that bring together features extraction and RR in the SPLE context: requirements versus features, core assets development in SPLE, and the contributions of our work in SPLE.

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

* Corresponding author at: Department of Software Engineering, Faculty of Computer Science & Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia. Tel.: +60 126927506
E-mail addresses: noor.hasrina@gmail.com, noorhasrina@iium.edu.my (N.H. Bakar), zarinahmk@um.edu.my (Z.M. Kasirun), norsaremah@iium.edu.my (N. Salleh).

http://dx.doi.org/10.1016/j.jss.2015.05.006
0164-1212/

of software *requirements* in accordance with IEEE Standard Glossary of Software Engineering Terminology, page 62 in IEEE Computer Society (1990) 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 et al., 2003). This is because text is commonly used to convey information to communicate stakeholders' needs (Niu and Easterbrook, 2008). 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 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 and Toval, 2009).

Meanwhile, software feature is defined as a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems (Kang et al., 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.

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 and Kang, 2005). Niu and Easterbrook (2008) highlighted the basic tenets of extractive approach of software product lines (SPL) that include maximal reuse and reactive development, particularly for small and medium-sized enterprises.

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 (Chen and Ali Babar, 2011; Metzger and 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 and Easterbrook, 2008; Alves et al., 2008; Kumaki et al., 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 adequately covers these issues. Fig. 1 illustrates the scope of our SLR contribution in regard to other related works in 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 (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 Fig. 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 and 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 (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 sub-processes 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 Fig. 1 and its summary is presented in Section 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 and Ensan, 2013). Selection of desirable features that is now readily available should be gradually performed with ample interaction with the stakeholders, as described by Czarnecki et al. (2004) as staged configuration.

Various literature reviews have been published in the area of DE and AE (as numbered in Fig. 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.

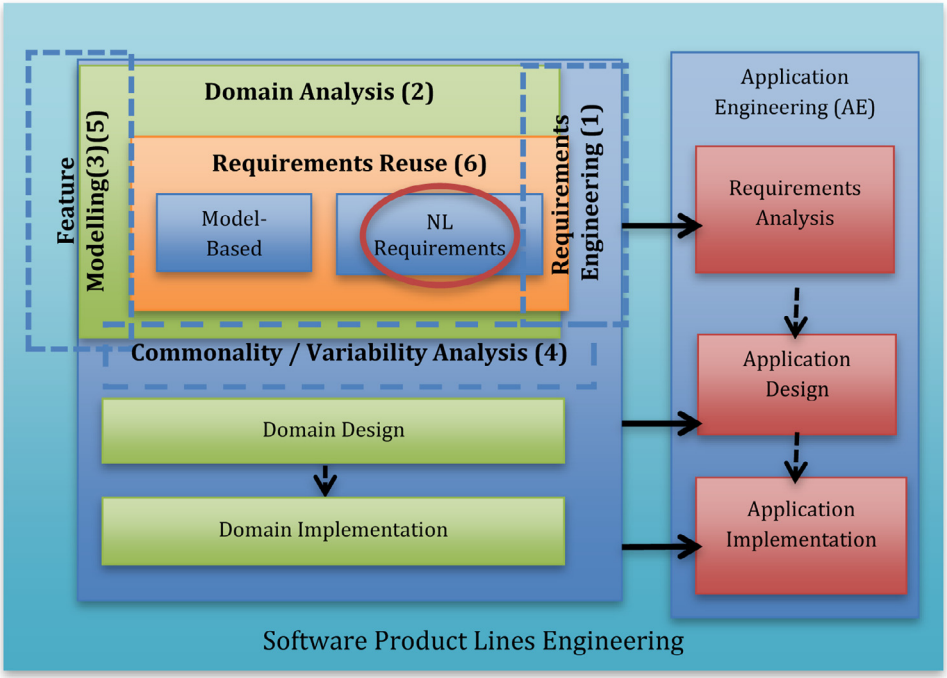


Fig. 1. Contribution of this SLR to SPLE.

The key contributions of this SLR are as follows:

- we offer 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
- we derive a number of key dimensions[1] 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, we have outlined 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.

Our 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, we summarise the related works. Section 3 reports the organisation of the SLR process: the research questions, search process, inclusion and exclusion criteria, and study quality assessment. Section 4 presents the results of this review based on the synthesis of the evidence. Section 5 provides a discussion of open issues and research implications, and lastly Section 6 provides the concluding remarks.

2. Related work

While conducting this review, we have also encountered other reviews related to areas that are close to RR in SPL, namely DA, requirements engineering (RE) in SPL, and automated feature modelling. This section provides a brief summary of the related studies.

2.1. Requirements engineering for software product lines: a systematic literature review (Alves et al., 2010)

Alves et al. (2010) reviewed the studies in the area of RE for SPL. This work aims to assess the research quality, synthesise evidence to provide suggestions on important implications for practice, and provide a list of open 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 sub-area 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 riskiest. Thus, future work is expected to combine the use of the extractive and reactive SPL adoption. Lastly, the authors conclude that future research should extend and improve the present research in an integrative manner (joint research and industry).

[1] 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 Dit et al. (2013) "Feature location in source code: A taxonomy and survey".

2.2. A systematic review of domain analysis solutions for product lines (Khurum and Gorschek, 2009)

Khurum and Gorschek (2009) conducted a review that covers a total of 89 primary studies in the DA solutions presented up until 2007. 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 solution in industry. However, the claims made were not supported by any qualitative or quantitative evidence.

2.3. Literature review on automated feature modelling (Benavides et al., 2010)

Benavides et al. (2010) provided a comprehensive literature review on the automated analysis of feature models for a period of 20 years (from 1990 to 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.

2.4. A systematic review of evaluation of variability management approaches in software product lines (Chen and Ali Babar, 2011)

Variability management (VM) is an important area in SPL (Northrop and Clements, 2015) and has been studied for almost 20 years since the early 1990s (Kang et al., 1990). The work in Chen and 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. 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.

2.5. Review on separation of concerns in feature diagram languages (Hubaux et al., 2013)

Hubaux et al. (2013) conducted a systematic review of separation of concerns in feature diagram languages. 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 detailed review and explanation of the techniques for composing concerns was also provided in this review.

2.6. Evaluation of a systematic approach to requirements reuse (Barreto et al., 2013)

Barreto et al. (2013) highlighted the reuse of requirement specifications by presenting a comparison of seven studies related to RR. 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 provides 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.

3. Review method

This section describes the process involved in conducting this SLR. Kitchenham and Charters (2007) 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. SLR provides a more systematic way to synthesise the research evidence by specifically using inclusion and exclusion criteria to set up the boundaries of evidence to be included in the review. In general, we are referring to Kitchenham and Charters' (2007) guidelines on performing SLR; however, we are also incorporating the guidelines on performing complementary snowballing search in locating articles to be included in the review (Wohlin and Prikladnicki, 2013) and we considered the recommendations on the importance to include manual target search on popular venues as appeared in (Jørgensen and Shepperd, 2007).

3.1. Formulating research questions

Petticrew and Roberts (2006) suggested that the formulation of research questions should focus on five elements known as PICOC.

Table 1 shows the *Population, Intervention, Comparison, Outcomes*, and *Context* of our research questions.

The primary focus of this SLR is to understand the available feature extraction approaches from the NL requirements to be reused in SPLE. In our SLR, we include all empirical studies presenting feature extraction approaches for NL requirements, specifically in the SPLE context.

Table 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

We do not include any comparison for feature extraction approaches in the PICOC, as it is not applicable to our research objectives.

Our SLR aims to answer the research questions (RQ) that are formulated based on the PICOC in Table 2.

3.2. Identification of relevant literature

Based on Kitchenham and Charters's (2007) guidelines, 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 and Prikladnicki, 2013) to minimise the possibility of missing important evidence. Additionally, Kitchenham and 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.

Our article search process is separated into three phases; Phase 1: online database search, Phase 2: complementary citation-based search, and Phase 3: manual target search.

3.2.1. Phase 1: online database search

Kitchenham and 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). We have used the Boolean OR to incorporate synonyms and alternative words. The Boolean AND was 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"))
```

We have searched through five databases that consist of Computer Sciences and Software Engineering articles: ACM, IEEE Xplore, ScienceDirect, Springer, and Scopus. In the initial selection, we applied the Inclusion and Exclusion criteria and removed irrelevant studies 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.

3.2.2. Phase 2: complementary citation-based search

In Phase 2, we used the citation-based search to find who cited the selected papers from Phase 1. We have looked at the references from each selected paper (backward snowballing) and listed down the titles that are relevant to our SLR. In addition, we also have looked on the Google Scholar to find out who have cited their papers (forward snowballing) and listed out the titles that look relevant to our SLR. 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 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.

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 and Shepperd, 2007). We have included manual target search from the most relevant venues in Software Engineering and Requirements Engineering fields in our article search process. 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 our topic (Alves et al., 2010; Benavides et al., 2010; 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. We have searched for all papers published in the selected venues starting from January 2000 up until December 2014.

3.3. Selection of studies

3.3.1. Inclusion and exclusion criteria

When conducting this review, we have 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.

Table 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?)

Table 3
Quality assessment (QA) checklists.

Item	Answer
QA1: Was the article refereed? (Leedy and Ormrod, 2010)	Yes/no
QA2: Was there a clear statement of the aims of the research? (Dybå and Dingsøyr, 2008)	Yes/no/partially
QA3: Is there an adequate description of the context in which the research was carried out? (Dybå and 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 and Dingsøyr, 2008)	Yes/no/partially
QA5: Were the testing results rigorously analysed? (Petticrew and 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

Our 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. We also found many articles mentioning 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

3.3.2. Data extraction plan

Data extraction plan is designed to accurately record the information obtained by the researchers from the primary studies (Kitchenham and 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

3.3.3. Study quality assessment

When designing the study quality assessment, we reused some of the questions in the published literature. Table 3 outlines six relevant criteria used to evaluate the quality of the selected studies, inspired by the quality assessment criteria for performing SLR used in Dybå and Dingsøyr (2008), Leedy and Ormrod (2010), Petticrew and Roberts (2006), Salleh et al. (2011) and the guidelines provided in Lam et al. (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 3 outlines these criteria.

The first author (Noor Bakar) was responsible for reading and completing the checklist for all the selected studies. As a way to validate the data extraction, the second author randomly selected 20% of the selected studies (in our case three papers were randomly picked by the second author). She then completed the QA checklist. Discrepancies found from the results were compared and discussed among all authors until a consensus is met.

The template used for the data extraction and quality assessment is available in Appendix C.

4. Results

In this section, we present the synthesis of evidence from our SLR. We begin with the analysis of the results from article searches, followed by the quality assessment results. Next, we present the answers to the main research questions from Table 2.

4.1. Results of article searches[2]

As mentioned in Section 3.2, we have divided our article searches 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.

4.1.1. 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. Fig. 2 illustrates the result on the number of articles retrieved from the online database searches.

4.1.2. Complementary citation-based search (snowball search)

Based on selected studies in Phase 1, we applied the backward and forward snowball searches. Firstly, with backward snowball search, we 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, we have 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

[2] The complete list of papers retrieved at each phase is available online at: https://www.dropbox.com/s/1f6hn38k6mtgq1k/SEARCH_RESULTS_SLR.xlsx?dl=0.

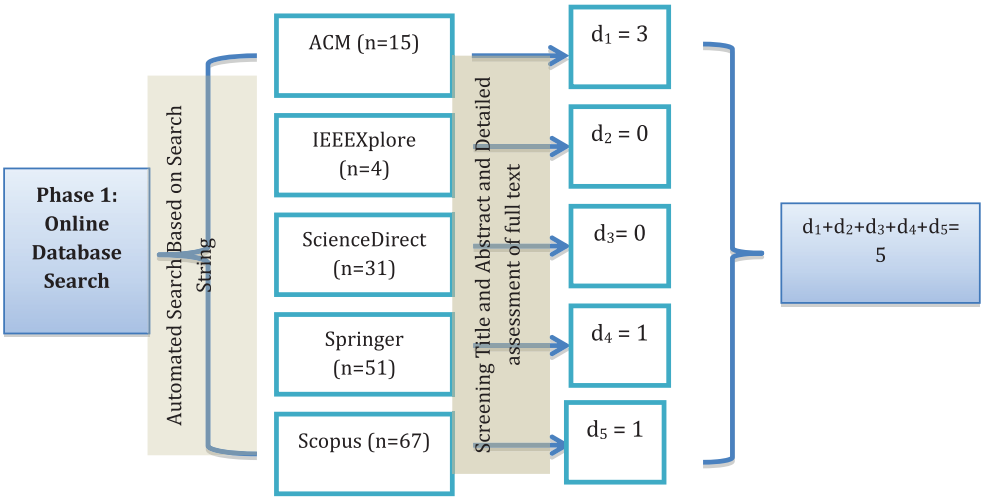


Fig. 2. Results of online database search.

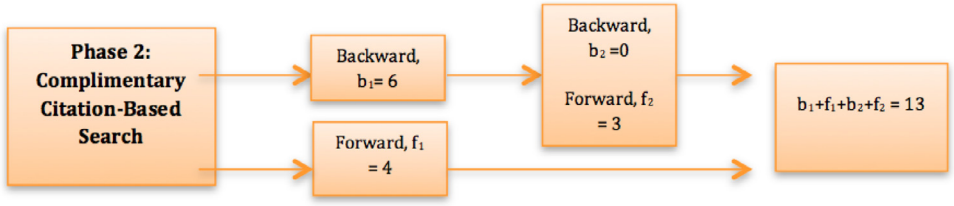


Fig. 3. Results of complimentary citation-based search.

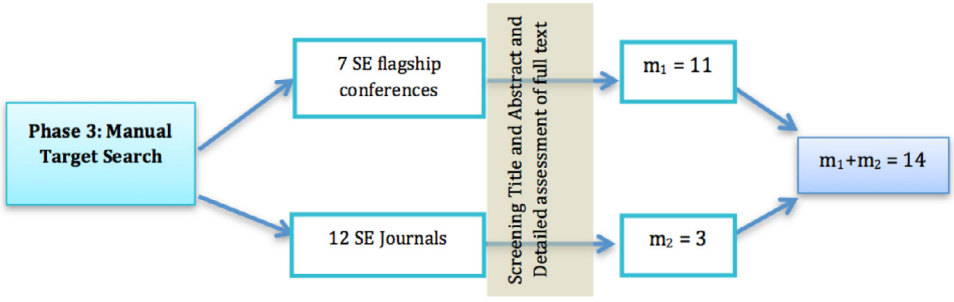


Fig. 4. Results of manual target search.

titles or abstracts and applying the inclusion criteria, we selected 13 additional papers.

Fig. 3 illustrates the result of the number of articles retrieved from the complimentary citation-based searches.

4.1.3. Manual target search

As mentioned previously, the manual target search was also performed to compliment automated search and snowball searches. We have used the popular venues to manually locate papers that were possibly not reached by the Phase 1 and Phase 2 searches. Fig. 4 illustrates the result of the manual target searches.

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

In total, we have collected 32 articles from the three phases of article searches. However, after removing duplicates, we are only left with 13 studies. 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, we only include the most recent publication or the most comprehensive version of the articles (see Appendix A for the complete list of selected primary studies).

4.1.4. 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 4.

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 et al. (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 synthesis, hence we only included the most comprehensive version of

Table 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

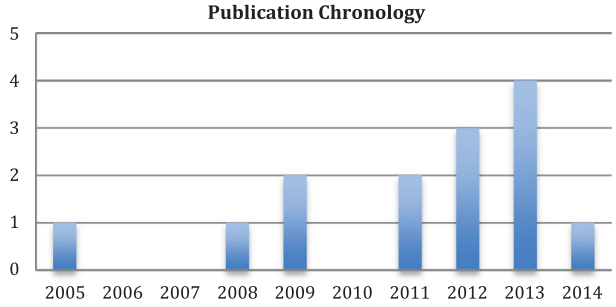


Fig. 5. Distribution of papers from 2005 to 2014.

the articles, in case of S8 (Hariri et al., 2013) is selected as primary study.

4.1.5. Publication chronology

The work on RR emerged as early as 1988, when Finkelstein published a paper in the Software Engineering Journal, entitled "Reuse 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 and Sutcliffe, 1992), a framework proposal on reuse of requirements and specification by Paredes and Fiadeiro (1995), and work by Massonet and Van Lamsweerde (1997). However, these works have been either 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 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, we identified that 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 symposium or workshop[3]). With this, we are confident that SPLE research topic has already achieved certain maturity for research publications, which potentially have published some works related to our interest. Thus, we have used the year 2000 as the starting point for our automated searches for articles in databases. Unfortunately, we only found one study that is relevant to our RQs, which was published five years later (in 2005) and appeared in Requirements Engineering conference (Chen et al., 2005). Other relevant studies appear from 2008 onwards. Based on this, we have used 2005 as the year to start our complimentary manual searches. Thus, it becomes clear to us that 2005 marks the emergence of the interest in feature extractions from NL requirements for SPLE. Fig. 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.

4.2. Quality assessment results

We used a score scale of 0–6: 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 5–6). Most studies (11 studies) achieved the score of more than 4, which 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, we 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.

4.3. 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 et al., 2010). Thus, there is a need for automated or semi-automated approach to cater to this extraction process. In this section, we examined the available approaches that extract the features from textual requirements 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 5.

4.3.1. 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.

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.

4.3.2. 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.

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

Table 5
Research questions and dimensions in reporting the review.

Research question	Dimension	Example	Refer to section
RQ1: What approaches were available to extract features from natural language requirements?	Types of input:	SRS documents [S1,S2,S4,S5,S6] Product descriptions/product brochures [S3,S7,S8,S9] User Comments [S13]	4.3.1
	Types of output:	Features [S3] [S13], feature tree/feature model [S4,S5,S6,S7,S8], verb-phrase [S2], clustered requirements [S1,S8,S11]	4.3.1 *(see Appendix B for classification dimension for input and output)*
RQ1.1: How were the commonality and variability addressed? Which technique was used?	Processes used	Text pre-processing: • 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 7)	4.3.2
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]	4.3.3
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]	4.3.4
RQ 2.2: Domain application	Domain application:	Automarker assignment [S2,S9] SmartHome [S4,S5] Antivirus [S8] Wiki [S7] MobileApps [S13]	

4.3.2.1. Extracting common features: NLP approaches. To classify the approaches used in extracting common features from NL requirements, we used the characterisation proposed by Falessi et al. (2010, 2013). Table 6 details out the types of NLP approaches across the selected studies in this review.

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 Sections 4.3.2.1.1–4.3.2.1.4 can be found in text.[4]

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 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.

Text pre-processing. Text pre-processing 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 pre-processing. 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 "verb-direct object" relations (Niu and Easterbrook, 2008; Niu et al., 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. (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

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

Table 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 pre-processing	(i) Tokenisation		/					/	/				/	/
	(ii) Part of speech tagging		/	/			[a]	/	/	/	/		/	/
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.
[a]S6 did not specify any NLP techniques, but used clustering algorithm.

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 pre-processing 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 et al., 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 pre-processed 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.

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.

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. (2010, 2013) categorised the Similarity Metrics into two categories: Vector Similarity Metrics (Dice, Jaccard, and 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 et al. (2005).

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. (2012) 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. Additionally, the NLP Toolkit provided to aid with Python programming is mentioned by S13 during the text pre-processing 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]

Thesaurus-based. 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 (2014) used WordNet lemmatiser from NLTK to group different inflected forms of words with similar part of speech tags (semantically equal but syntactically different). This step reduces the number of feature descriptors that needed to be inspected at later stages.

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

Table 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

4.3.2.2. Extracting common features: clustering approaches and more. We have 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 7). Other approaches that are beyond clustering such as Latent Dirichlet Allocation, Propositional Logic, and more are also listed in Table 7.

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.

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 et al., 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 and Maalej, 2014) to group features that tend to co-occur in the same user reviews of various mobile apps.

4.3.2.3. 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 (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 $V = \{C_1 \cup C_2 \ldots \cup C_n\} \backslash C$. 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 denote 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. 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. Five variability patterns were found: mandatory, optional, dead feature, multivalue, and real value.

Bagheri and Ensan (2013) 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 demands further explanation on how it can be done as the paper did not further elaborate on how to handle variants feature extraction.

4.3.3. 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.

4.3.4. 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.

4.3.4.1. 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. Fig. 6 indicates these distributions.

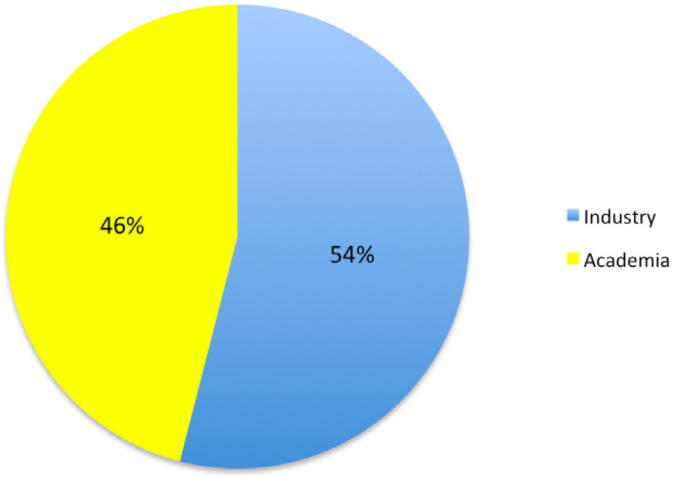
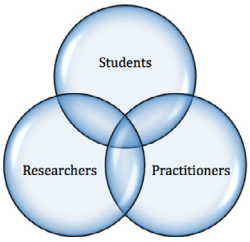


Fig. 6. Evaluation context.



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 ∩ Researchers	*= {S3, S5, S8, S9, S11}*
Students ∩ Researchers	*= {S1, S2, S10}*
Students ∩ Practitioners	*= 0*

Fig. 7. Evaluators.

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–practitioners' collaboration and four studies reported the collaboration between students and researchers. Fig. 7 illustrates the summary of this result.

4.3.4.2. 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.

4.3.4.3. 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 8 details out the metrics used by eight studies.

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:

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

Table 8
Measure used in selected studies.

Metrics	List of studies
Purity[a]	S8
Entropy[b]	S4, S10
Recall and Precision[c]	S2, S8, S9, S10,S11,S12, S13
F-measure	S9, S10, S11, S12, S13

[a] C.D. Manning, P. Raghavan, and H. Schutze. (2008). "Introduction to Information Retrieval". Cambridge University Press.
[b] 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.
[c] Source: http://en.wikipedia.org/wiki/Precision_and_recall.

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 and 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:

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

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 8, 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.

4.3.4.4. 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).

5. Discussion

This section firstly presents a discussion on the implications of this study (Section 5.1). Later in Section 5.2, we will discuss 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 5.3.

5.1. SLR study implications

5.1.1. SRS documents as the main input to the extraction process

We found that SRS or requirements documents 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

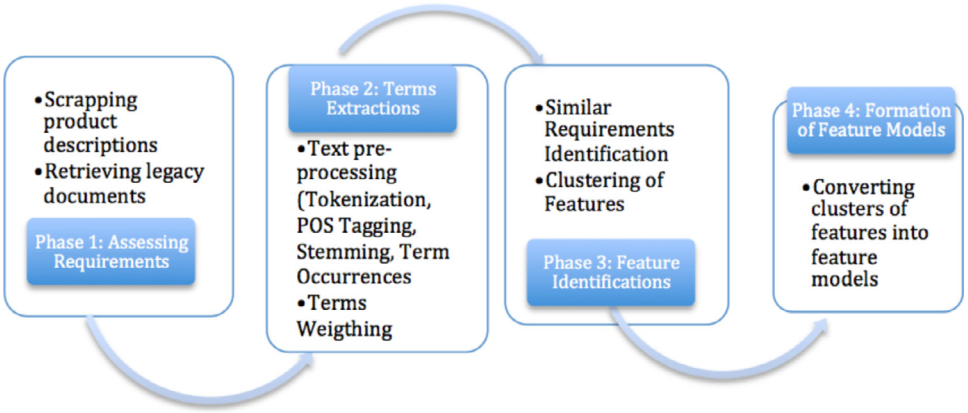


Fig. 8. Feature extraction process for requirements reuse.

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 (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 unaccessibility can be the contributing factors to the lower number of publication in the RR research area. In fact, we have reported this observation in our earlier publication in Bakar and Kasirun (2014). In case where SRS documents are not available, product descriptions from brochures and user comments were used as 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; Guzman and Maalej, 2014).

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 Fig. 8 and its details pertaining to input, process, and output are presented in Table 9. This process can be interpreted as taxonomy since it provides detailed granularity on the available processes and approaches, which are very useful for practitioners and researchers interested in this area to guide future research, development, and implementation.

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 scrapped using open source scrapping 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 9
Input, process, and output for feature extraction process.

Phase	Input	Process and 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	Scrapping (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 pre-processing (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 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 (Davril et al., 2013; Yu et al., 2013; Casamayor et al., 2012)), *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.

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 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, 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 and Pohl (2014) as open research challenges. More discussions on handling and measuring variability will be presented in Section 5.1.5 of this paper.

Pertaining to subjects used or the evaluators in the evaluation, we found out that all studies involve researchers (researchers or research students) as the evaluators in the evaluation (see Fig. 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, we can understand that research-industry collaboration requires additional effort, while most researchers which are postgraduate students tend to have very limited time frame in their research.

5.1.4. Practitioners' guidelines and support tools

We have included a question in the Quality Assessment 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 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 1990–2009 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.

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.

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 (Chen and 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 in Section 5.1.3 of this paper and also reported in a recent SLR publication in 2014, pertaining to variability in software systems (Galster et al., 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 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.

5.2. The reduction in number of selected studies

We have selected 13 studies ranging from 2005 to 2014 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 and Gorschek (2009). Benavides et al. (2009) included 53 studies when reviewing about automated feature modelling while the authors in Chen and 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 and Gorschek, 2009; Benavides et al., 2009; Chen and 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), Von Knethen et al. (2002), and Robinson and Woo (2004). We also excluded the studies regarding extractions of source code such as in Marcus and Maletic (2003) or selection of components for reuse in SPL (Abraham and Aguilar, 2007), or RR through pattern (Renault et al., 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 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.

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. Our 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. We address the issue of bias in study selection through multiphase search approaches (online database, snowballing, and manual search on targeted journals and conferences) that help to minimise the possibility of missing evidence.

6. Conclusion

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 systematically review and synthesise the evidence of how to extract features from NL requirements for reuse in SPLE. To fill this gap, we have conducted a systematic literature review for feature extraction approaches from NL requirements for reuse in SPLE. We have selected 13 primary studies from searching the literature through three main phases: automated database search, complimentary citation-based search, and manual target search. We have outlined inclusion and exclusion criteria for selecting the primary studies, which were meant to answer our main research questions.

We have answered the main research questions, and importantly the result is presented in Section 4.3. Our 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) we identified that 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.

We believe the findings of this study can supply important contribution to the practitioners and researchers as it provides them with useful information about the different aspects of RR approaches. For practitioners, our SLR has categorised the 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 in 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.

Acknowledgements

This research is funded by the Ministry of Higher Education Malaysia with Research Grant# FP050/2013A, with Assoc. Prof. Dr. Zarinah Kasirun from the Department of Software Engineering, FSKTM of the University of Malaya as the principal investigator. The first author's study is sponsored by the International Islamic University Malaysia. The authors would like to thank the reviewers and associate editors for the insightful comments and suggestions made to this paper. We would also like to thank the staff at the International Islamic University Malaysia and the University of Malaya for the support given in completing this systematic literature review.

Appendix A. Selected studies.

ID	Author	Paper title	Venue/Source
S1	Kumaki K., Washizaki, H., and 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., and 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., and 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., and 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., and 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 (Chen et al., 2005)
S7	Acher, M., Cleve, A., Perrouin, G., Heymans, P., Vanbeneden, C., Collet, P., and 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., and Mobasher, B.	Supporting domain analysis through mining and recommending features from online product listings	IEEE Transaction of Software Engineering, *39*(12), December 2013. (Hariri et al., 2013)
S9	Mu, Y., Wang, Y., and 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., and Liu, B.	Mining and recommending software features across multiple web repositories	Internetware, October 2013 (Yu et al., 2013)
S11	Bagheri, B., Ensan F., and 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., and F. Houdek	Semi-automatic identification of features in requirement specifications	RE 2011: 313–318 (Boutkova and Houdek, 2011)
S13	Guzman, E., and Maalej, W.	How do users like this feature? A fine grained sentiment analysis of app reviews	RE 2014: 153–162 (Guzman and Maalej, 2014)

Appendix B. Input (types of requirements) and output (features)

	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]					•				

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	
Quality assessment checklist	
Item	Answer
QA1: Was the article refereed? (Leedy and Ormrod, 2010)	Yes/no
QA2: Was there a clear statement of the aims of the research? (Dybå and Dingsøyr, 2008)	Yes/no/partially
QA3: Is there an adequate description of the context in which the research was carried out? (Dybå and 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 and Dingsøyr, 2008)	Yes/no/partially
QA5: Were the testing results rigorously analysed? (Petticrew and 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].

References

Abraham, B.Z., Aguilar, J.C. Software component selection algorithm using intelligent agents, in N.T. Nguyen, et al. (Ed.), Lecture Notes in Computer Science (LNCS). Berlin/Heiderberg: Springer-Verlag, 2007, pp. 82–91.

Acher, M., Cleve, A., Perrouin, G., Heymans, P., Vanbeneden, C., Collet, P., Lahire, P., 2012. On extracting feature models from product descriptions. In: Proceedings of Sixth International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS'12), pp. 45–54.

Alves, V., Niu, N., Alves, C., Valença, G., 2010. Requirements engineering for software product lines: a systematic literature review. Inf. Softw. Technol. 52 (8)), 806–820.

Alves, V., Schwanninger, C., Barbosa, L., Rashid, A., Sawyer, P., Rayson, P., Pohl, C., Rummler, A., 2008. An exploratory study of information retrieval techniques in domain analysis. In: 2008 12th International Software Product Line Conference, pp. 67–76.

Bagheri, E., Ensan, F., 2013. Dynamic decision models for staged software product line configuration. Requir. Eng. 19 (2), 187–212.

Bagheri, E., Ensan, F., Gasevic, D., 2012. Decision support for the software product line. Autom. Softw. Eng. 19 (3), 335–377.

Bakar, N.H., Kasirun, Z.M., 2014. Exploring software practitioners perceptions and experience in requirements reuse an empirical study in Malaysia. Int. J. Softw. Eng. Technol. 1 (2).

Barreto, F., Benitti, V., Cezario, R., 2013. Evaluation of a Systematic Approach to Requirements Reuse. J. Univ. Comput. Sci. 19 (2), 254–280.

Basili, V.R., 1994. Goal question metrics paradigm. Encyclopedia of Software Engineering.

Benavides, D., Segura, S., and Ruiz-cort, A. Automated analysis of feature models: a detailed literature review, December 2009.

Benavides, D., Segura, S., Ruiz-Cortes, A., 2010. Automated analysis of feature models 20 years later: a literature review. Inf. Syst. 35, 615–636.

Bonin, F., Orletta, F.D., Venturi, G., Montemagni, S., 2010. A contrastive approach to multi-word term extraction from domain Corpora. In: Seventh International Conference on Language Resources and Evaluation (LREC 2010), pp. 3222–3229.

Boutkova, E., Houdek, F., 2011. Semi-automatic identification of features in requirement specifications. In: 2011 IEEE 19th International Requirements Engineering Conference, pp. 313–318.

Boutkova, E., Houdek, F., 2011. Semi-automatic identification of features in requirement specifications. In: 2011 IEEE 19th International Requirements Engineering Conference, pp. 313–318.

Casamayor, A., Godoy, D., Campo, M., 2012. Functional grouping of natural language requirements for assistance in architectural software design. Knowl. Based Syst 30, 78–86.

Chen, K., Zhang, W., Zhao, H., Mei, H., 2005. An approach to constructing feature models based on requirements clustering. In: 13th IEEE International Conference Requirements Engineering, pp. 31–40.

Chen, L., Ali Babar, M., 2011. A systematic review of evaluation of variability management approaches in software product lines. Inf. Softw. Technol. 53 (4), 344–362.

Clements, P., Northrop, L.M., 2002. Software Product Lines: Practices and Patterns. Addison Wesley Professional, Boston, MA, USA.

Cui, X., Potok, T.E., Palathingal, P., 2005. Document clustering using particle swarm optimization. In: Proceedings of Swarm Intelligence Symposium, 2005 (SIS 2005). IEEE, pp. 185–191.

Czarnecki, K., Helsen, S., Eisenecker, U., 2004. Staged configuration using feature models. In: Software Product Lines Conference.

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. In: Proceedings of 2013 9th Joint Meeting of Foundations of Software Engineering (ESEC/FSE 2013), p. 290.

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. J. Softw. Evol. Process 25, 53–95.

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, p. 10.

Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. 50 (9–10), 833–859.
Eriksson, M., Borstler, J., Borg, K., 2006. Sotware product line modeling made practical: an example from Swedish defense industry. Commun. ACM 49 (12), 49–53.
Falessi, D., Cantone, G., Canfora, G., 2010. A comprehensive characterization of NLP techniques for identifying equivalent requirements. In: ESEM.
Falessi, D., Cantone, G., Canfora, G., 2013. Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. IEEE Trans. Softw. Eng. 39 (1), 18–44.
Faulk, S.R., 2001. Product-line requirements specification (PRS): an approach and case study. In: Proceedings of Fifth IEEE International Symposium on Requirements Engineering, 2001, pp. 48–55.
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 (SPLC'13), p. 116.
Finkelstein, A., 1988. Re-use of formatted requirements specifications I " /," . Softw. Eng. J 186–197.
Frakes, W.B., Kang, K., 2005. Software reuse: status and future. IEEE Trans. Softw. Eng. 31 (7), 529–536.
Galster, M., Weyns, D., Tofan, D., Michalik, B., Avgeriou, P., 2014. Variability in software systems—a systematic literature review. IEEE Trans. Softw. Eng. 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.
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 Trans. Softw. Eng. 39 (12), 1736–1752.
Hoenkamp, E., 2011. Trading spaces: on the lore and limitations of latent semantic analysis. In: Proceedings of the Third international conference on Advances in information retrieval theory (ICTIR'11), pp. 40–51.
Huang, A., 2008. Similarity measures for text document clustering. In: New Zealand Computer Science Research Student Conference (NZCSRSC) April.
Hubaux, A., Tun, T.T., Heymans, P., 2013. Separation of concerns in feature diagram languages. ACM Comput. Surv. 45 (4), 1–23.
IEEE Computer Society, 1990. IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard.
Jørgensen, M., Shepperd, M., 2007. A systematic review of software development cost estimation studies. IEEE Trans. Softw. Eng. 33 (1), 33–53.
Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. Feature oriented domain analysis (FODA) feasibility study. Pittsburgh, PA, 1990.
Khurum, M., Gorschek, T., 2009. A systematic review of domain analysis solutions for product lines. J. Syst. Softw. 82 (12), 1982–2003.
B.A. Kitchenham and S. Charters, Procedures for performing systematic literature reviews in software engineering: EBSE Technical Report version 2.3, EBSE-2007-01. Keele, UK, 2007.
Krueger, C.W., 1992. Software Reuse. ACM Comput. Surv. 24 (2), 131–183.
Krueger, C.W., 2001. Easing the transition to software mass customization. In: International Workshop on Product Family Engineering, Bilbao, Spain, October 2001, pp. 282–293.
Krueger, C., 2002. Eliminating the Adoption Barrier. IEEE Softw (July/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 16th International Software Product Line Conference (SPLC'12), vol. 1, p. 115.
Lam, W., McDermid, J.A., Vickers, A.J., 1997. Ten steps towards systematic requirements reuse. Requir. Eng. 2 (vol. 2), 102–113.
Lam, W., McDermid, J.A., Vickers, A.J., 1997. Ten steps towards systematic requirements reuse. In: International Conference of Requirements Engineering, 2.
Leedy, P.D., Ormrod, J.E., 2010. Practical Research Planning and Design, ninth ed. Pearson Education Inc, p. 10.
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. Inf. Softw. Technol. 52 (1), 1–13.
Maiden, N.A., Sutcliffe, A.G., 1992. Exploiting usable specifications through analogy. Commun. ACM 35 (4), 55–64.
Marcus, A., Maletic, J.I., 2003. Recovering documentation-to-source-code traceability links using latent semantic indexing. In: Proceedings of 25th International Conference on Software Engineering, 2003, pp. 125–135.
Massonet, P., Van Lamsweerde, A., 1997. Analogical reuse of requirements frameworks. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997, pp. 26–37.
Metzger, A., Pohl, K., 2014. Software product line engineering and variability management: achievements and challenges. In: FOSE, pp. 70–84.
Meyer, M.H., Lehnerd, A.P., 1997. The Power of Product Platform. Free Press, New York.
Monzon, A., 2008. A practical approach to requirements reuse in product families of onboard systems. In: 16th IEEE International Requirements Engineering Conference, pp. 223–228.
Moros, B., Toval, A., Rosique, F., Sánchez, P, 2013. Transforming and tracing reused requirements models to home automation models. Inf. Softw. Technol 55 (6), 941–965.
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.
Neighbors, J.M., 1984. The Draco approach to constructing software from reusable components. IEEE Trans. Softw. Eng. SE-10 (5), 564–574.
Nicolás, J., Toval, A., 2009. On the generation of requirements specifications from software engineering models: a systematic literature review. Inf. Softw. Technol. 51 (9), 1291–1307.
Niu, N., Easterbrook, S., 2008. Extracting and modeling product line functional requirements. In: 2008 16th IEEE International Requirements Engineering Conference, pp. 155–164.
Niu, N., Savolainen, J., Niu, Z., Jin, M., Cheng, J.-R.C., 2013. A systems approach to product line requirements reuse. IEEE Syst. J. 1–10.
L.M. Northrop and P.C. Clements, 2015. A framework for software product line practice, Version 5.0. Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu/productlines/frame_report/index.html [accessed 25.04.15].
Paredes, C., Fiadeiro, J.L., 1995. Reuse of requirements and specifications—a formal framework—. SSR'95. ACM, pp. 263–266.
Petticrew, M., Roberts, H., 2006. Systematic Reviews in the Social Sciences: A Practical Guide. Blackwell Publishing, Maryland USA.
Pohl, K., Bockle, G., Van der Linden, F., 2005. Software Product Line Engineering. Springer-Verlag, Berlin/Heidelberg.
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.
Robinson, W.N., Woo, H.G., 2004. Finding reusable UML sequence diagrams automatically. IEEE Softw 21 (5), 60–67.
Salleh, N., Mendes, E., Grundy, J., 2011. Empirical studies of pair programming for CS/SE teaching in higher education: a systematic literature review. IEEE Trans. Softw. Eng. 37 (4), 509–525.
Von Knethen, A., 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.
Weston, N., Chitchyan, R., Rashid, A., 2009. A framework for constructing semantically composable feature models from natural language requirements. In: Software Product Lines Conference, pp. 211–220.
Wohlin, C., Prikladnicki, R., 2013. Systematic literature reviews in software engineering. Inf. Softw. Technol. 55 (6), 919–920.
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 Internetware (Internetware'13), pp. 1–9.

Noor Hasrina Bakar received her BSc (Information Technology) from Marquette University in Milwaukee, Wisconsin, USA in 1998, and MSc (Computer Science) from University of Malaya in 2009. She holds a lecturing position in the Centre for Foundations Studies of International Islamic University Malaysia, IIUM. She is currently a full time PhD scholar in the Software Engineering Department at the Faculty of Computer Science and Information Technology, University of Malaya. Her current research is in the area of requirements reuse in software product lines. She also has 12 years teaching experience at the university foundation levels in Computer Science area.

Zarinah M. Kasirun received her BSc (CS) and MSc (CS) from National University of Malaysia (UKM) in 1989 and 1993, respectively. She received her PhD from University of Malaya in 2009. Currently she is an Associate Professor in Software Engineering Department at the Faculty of Computer Science and Information Technology, University of Malaya. She has vast experience in teaching and published many academic papers in conferences and journals. She actively supervises many students at all levels of study—Bachelor, Master and PhD. Her research interest includes requirements engineering, requirements visualization, software metrics and quality and Software Product Line Engineering.

Norsaremah Salleh is an Assistant Professor and the former Head of Computer Science Department at International Islamic University Malaysia (IIUM). She received her PhD in Computer Science from the University of Auckland, New Zealand. Her research interests include the areas of empirical software engineering (SE), evidence-based software engineering, computer science/software engineering education, and social network sites research.