Journal of Information Technology and Computer Science
Volume 5, Number 3, Desember 2020, pp. 279-292
Journal Homepage: www.jitecs.ub.ac.id

Automated Features Extraction from Software Requirements Specification (SRS) Documents as The Basis of Software Product Line (SPL) Engineering

M Syauqi Haris[1] , Tri Astoto Kurniawan[2] , Fatwa Ramdani[3]
[1,2,3]Computer Science Faculty of Brawijaya University Malang
[1]syauqi@student.ub.ac.id, [2] triak@ub.ac.id, [3] fatwaramdani@ub.ac.id

Received 13 July 2020; accepted 14 Desember 2020

Abstract. Extractive Software Product Line Engineering (SPLE) puts features on the foremost aspect in domain analysis that needs to be extracted from the existing system's artifact. Feature in SPLE, which is closely related to system functionality, has been previously studied to be extracted from source code, models, and various text documents that exist along the software development process. Source code, with its concise and normative standard, has become the most focused target for feature extraction source on many kinds of research. However, in the software engineering principle, the Software Requirements Specification (SRS) document is the basis or main reference for system functionality conformance. Meanwhile, previous studies of feature extraction from text document are conducted on a list of functional requirement sentences that have been previously prepared, not literally SRS as a whole document. So, this research proposes direct processing on the SRS document that uses requirement boilerplates for requirement sentence statement. The proposed method uses Natural Language Processing (NLP) approach on the SRS document. Sequence Part-of-Speech (POS) tagging pattern is used for automatic requirement sentence identification and extraction. The features are acquired afterward from extracted requirement sentences automatically using the word dependency parsing rules. Besides, mostly the previous studies about feature extraction were using non-public available SRS document that remains classified or not accessible, so this work uses selected SRS from publicly available SRS dataset to add reproducible research value. This research proves that requirement sentence extraction directly from the SRS document is viable with the precision value from 64% to 100% and recall value from 64% to 89%. While features extraction from extracted requirement sentences has a success rate from 65% to 88%.

Keywords: Software Product Line, Feature Extraction, Natural Language Processing

1. Introduction

Software Product Lines (SPL) arise as a new concept in software reuse [1]. It is also considered as a proven paradigm strategy on software reuse that makes it possible for the industry to reduce development costs up to 61%, shorten time-to-market but still maintain the product quality [2]. More than 50% of software industry practitioners just realized the importance of SPL after producing several products with an ad-hoc or single system development [3]. This condition, when the developer tries

to evolve those existing products into the Software Product Line (SPL), is called the extractive SPL method. In this process, the SPL core asset base will be built from the features that are extracted from existing products, while the feature itself is defined as a logical unit of behavior that is determined by a set of functional requirements [4] or abstraction from system functionality [5].

Studies on the feature extraction from existing systems mostly use the source code as the object or input of the extraction process [6][7][8][9][10][11]. While other studies also conducted to use models including a class diagram and use case diagram as the objects for the extraction process [12][13]. However, most software developer only measures their product quality on the released software product or the implementation result regardless of the original requirement [14], Therefore, software feature extraction from specification document is more suitable based on the software engineering perspective rather than model or source code to acquire the more valid feature. This is because the specification document is the basis of the validation and verification of system functionality in the software development process [15][16].

Currently, most studies on SPL feature extraction from Software Requirement Specification (SRS) document were processing the list of requirements that have been previously prepared, not SRS as a whole document [17][18][19]. As such, this method still needs expert intervention to manually separate requirement sentences from the SRS document that might be tedious and error-prone. In this research, we process directly from SRS documents that use requirement boilerplate to construct that requirement statements. Such requirements provide specific patterns to be processed using Natural Language Processing (NLP) approach.

This article is constructed using the following sections. Section 2 will situate related works that were previously conducted to show the state of the art in this research area. Section 3 will explain the methodology of this research and followed by section 4 that shows the research results. Section 5 will discuss the result measurement and opinion from the researcher to explain the logical manner. The last section will conclude this research along with the future work planning and suggestions for other researchers.

2. Related Works

Previous three studies on feature extraction from SRS document process requirement statement sentences that were already broken down into the list as they appear in the document. First, the statistical approach Term Frequency and Inverse Document Frequency (TF/IDF) is used in feature mining from the SRS document's functional requirement sentences. The research focused on several Mapping Rules (MRs) to identify the Semantic Model (SM) from each functional requirement sentence [17]. Second, Feature and Feature Relation Extraction (FFRE) tool for Eclipse plugin are also introduced to assist the feature model extraction process from the SRS document. This tool use NLP processing to identify actor, action, and object from each requirement sentence and heuristic processing afterward to determine which features are mandatory and which are optional [18]. The third research was done by collaborative researchers between academic and industry practitioners. The feature is defined as a higher-level abstraction of requirements or specifically as a cluster of requirements. In their framework, the extraction method for feature candidates is Latent Semantic Analysis (LSA) and Vector Space Model (VSM) while further classification is conducted using Hierarchical Agglomerative Clustering (HAC) [20].

Feature extraction researches working on natural language document were also done on product descriptions or brochures. First research initialized with the web-scrapping process on Softpedia for antivirus software, meaningful terms are then acquired using the TF-IDF method. The classification process is done afterward using a two-stage Spherical k-Means (SPK-Means) clustering algorithm to construct feature candidates, while the Frequent Pattern (FP) growth algorithm is used for feature naming [21]. Second research extract software feature commonality and variability mining on domain-specific natural language documents. The extraction process are done automatically using contrastive analysis of Natural Language Processing (NLP) approach to identify single and multi-word domain-specific terms [22].

Another Information retrieval (IR) with a system-oriented approach has been proposed for automatic identification of functional requirements from SRS documents. Orthogonal Variability Model (OVM) and Filmore's case theory are used to extract and characterize functional requirements. The functional Requirement Profile (FRP) concept is introduced in this research as a domain analysis method to give quick insight into system functionalities. FRP itself is represented as a "verb-do" pair of words that state user-visible system functionality. In the extraction process, this research method only able to parse the SRS document that already complies with the IEEE-830 document structure. This standardization enables parsing only on the specific section of the SRS document [23].

3. Basic Notions

3.1. SRS Documents and Requirement Boilerplate

Consistent use of language in the SRS writing will make a statement of requirements easy to understand and identify. A simple example is a use of the word "shall" which shows the existence of a statement of a requirement in a sentence. Another example is the choice of the words "shall", "should", and "may" which express different levels of priority to a requirement [4]. For determining the syntactical structure of a single requirement, the syntactical requirements template as a sentence blueprint was introduced [24] and illustrated as seen in Figure 1. This template is now commonly referred to as requirement boilerplates. While boilerplates itself mean the grammatical structure that is formulated to provide specific patterns to avoid complex and inconsistent patterns [25]. Furthermore, the usage of boilerplates for requirements will facilitate automated document analysis using Natural Language Processing (NLP).

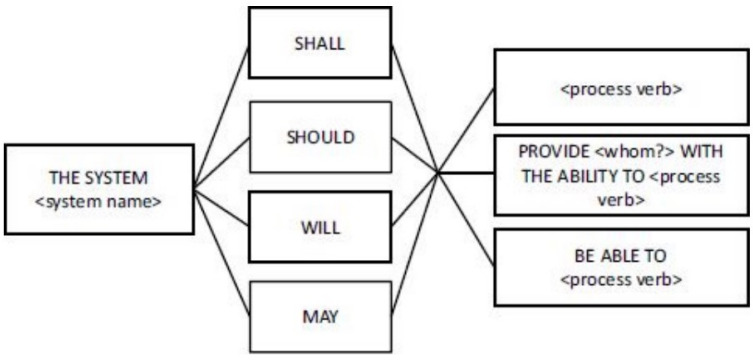

Figure 1. Requirements Boilerplate Template

3.2. Feature on Extractive Software Product Line (SPL)

Feature Model (FM) is a notation that is commonly used as an artifact resulting from the domain analysis process. Feature Model Diagram is considered capable of describing the similarities and variations of a set of features that can be applied to SPL [26]. The common process of making FM is manually done by an expert based on existing software product descriptions, whether structured or not. This process is error-prone and time-consuming [27]. Automatic FM diagram formation has been proposed in a study based on a feature catalog extracted from source code [8].

The extractive software product line approach using the requirements engineering (RE) and natural language as a source or object extraction that also called the requirements reuse (RR) [23]. The results of the requirements reuse process can be a list of features or a grouping of features from a single software product. In the context of SPL, the extraction results can be in the form of feature models (FM) of several software products as SPL core asset base, although in the process are still semi-automatic [19] [28]. The illustration of the extractive software product line is illustrated in Figure 2.

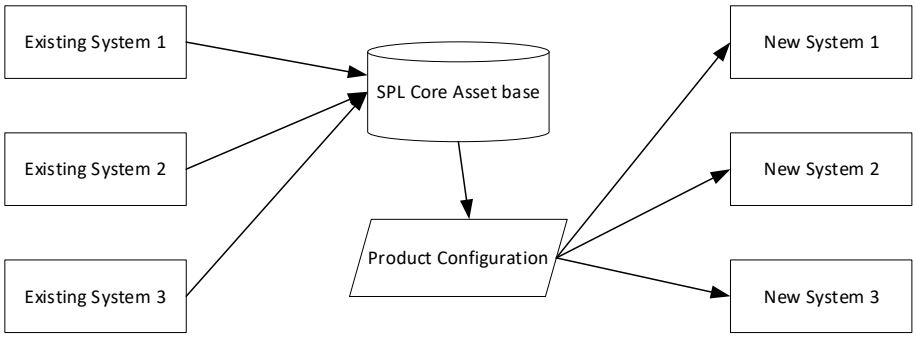

Figure 2. Extractive SPL Illustration

3.3. Natural Language Processing (NLP) for Information Extraction

Currently, NLP is widely used in studies working on human language documents, since there are many public NLP libraries to use i.e. Google SyntaxNet, Stanford CoreNLP, NLTK Phyton library, and spaCy [29]. NLP library provides several pipelines to process the documents as the input are illustrated in Figure 3. *Sentencizer* can split text document to sentences series, *Tokenizer* can split document or sentence to token series, token itself can be individual word or words phrase when text chunking method is applied, *Part-of-Speech (POS) tag* is code tagging for every token identified, it can be noun, verb, determiner, adjective, adverb, etc [25]. NLP library also can have a pre-trained model to add more capability on *Named Entity Recognizer (NER)* to automatically identify detected entity as person, organization, nation, etc. It also can have *Dependency Parser* for text chunking (phrase detection) or sentence boundary detection. Both NER and dependency parser is available in the spaCy NLP library [30].

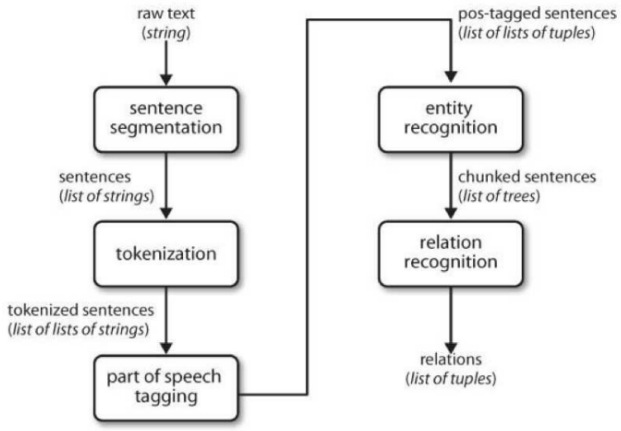

Figure 3. NLP Process Pipeline for Information Retrieval from Text Document [31]

At the moment, there are few limitations in published researches on feature extraction from natural language documents, i.e. unavailable tools for evaluation, restricted or limited input, irrelevant feature naming, non-reproducible result, and domain engineer intervention in the process [32]. While this research is aimed to produce a tool for automatically extracting software features directly from SRS documents without any human intervention in the process. The tool will be applied and tested using selected SRS from the Public Requirement Engineering (PURE) dataset [33] to justify its correctness.

4. Methodology

4.1. Research Questions

The research questions in this research are established as follows:

- RQ 1: What approach or technique needs to be constructed to automatically extract features directly from the SRS document?
- RQ 2: What rules of processing that need to be formulated to automatically extract requirement sentences and features?
- RQ 3: How accurate is the automated extraction method?

4.2. Data Acquisition

SRS documents that are used in this research selected from Public Requirement Engineering (PURE) dataset that is published in 2017 and publicly available on the internet [33]. Since this research focuses on automatic feature extraction from the SRS document, so this dataset is suitable to be used. Rationalization is also made to select certain documents that are possible to be checked manually for the analysis described in Figure 4.

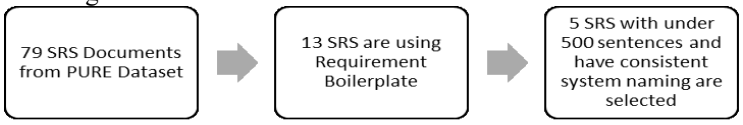

Figure 4. Dataset Rationalization Illustration

Based on 79 SRS documents in the dataset, there are 13 documents consistently using requirement boilerplate to express functional requirement sentence. Finally, this research only selects 5 SRS documents that have less than 500 sentences and have consistency in system naming. The rationalizations are needed to accommodate manual checking for the analysis presented in Figure 4 and five selected SRS documents are presented in Table 1.

Table 1. Selected SRS Documents

No	SRS Document Title	Number of Sentences
1	Crime & Criminal Tracking Network and Systems (CCTNS)	431
2	Digital Home System (DH)	404
3	e-Store	322
4	Laboratory Information System (LIS)	314
5	Puget Sound Enhancements (Puget)	495

4.3. Proposed Solution Framework

The feature extraction automation framework for SPL from SRS documents is the basic conceptual structure used in software development in this study. In this framework, the SRS dataset which consists of a complete SRS document is extracted to produce raw text, then the raw text is pre-processed to eliminate certain sentence variations noises and ready to be processed with NLP. NLP in this framework uses the SpaCy NLP library with a pre-trained model. The first process in NLP is sentencizer and tokenizer to break up preprocessed text documents into sentences consisting of tokens (separated words), while noun-chunking is used to recognize phrases so that they are not considered as separated words. The next process is implementing the POS tagging sequence pattern to automatically identify requirement sentences. Through a list of requirement sentences, an abstraction is needed to obtain system features. In this study, the word dependency parsing rules are proposed to retrieve only certain words that indicate the system function from each requirement sentence. An overview of the feature extraction automation framework from the SRS document is presented in Figure 5.

4.4. Text Pre-Processing

After obtaining text data from the SRS document, there are several preliminary processes for the text data to be processed properly by the NLP engine. In this process, variations in requirement boilerplate sentence implementations are simplified to facilitate the main verb recognition for further NLP processing. But, in this stage, these preprocesses are applied to all sentences in the text data. Sentence simplification rules are illustrated in Figure 6.

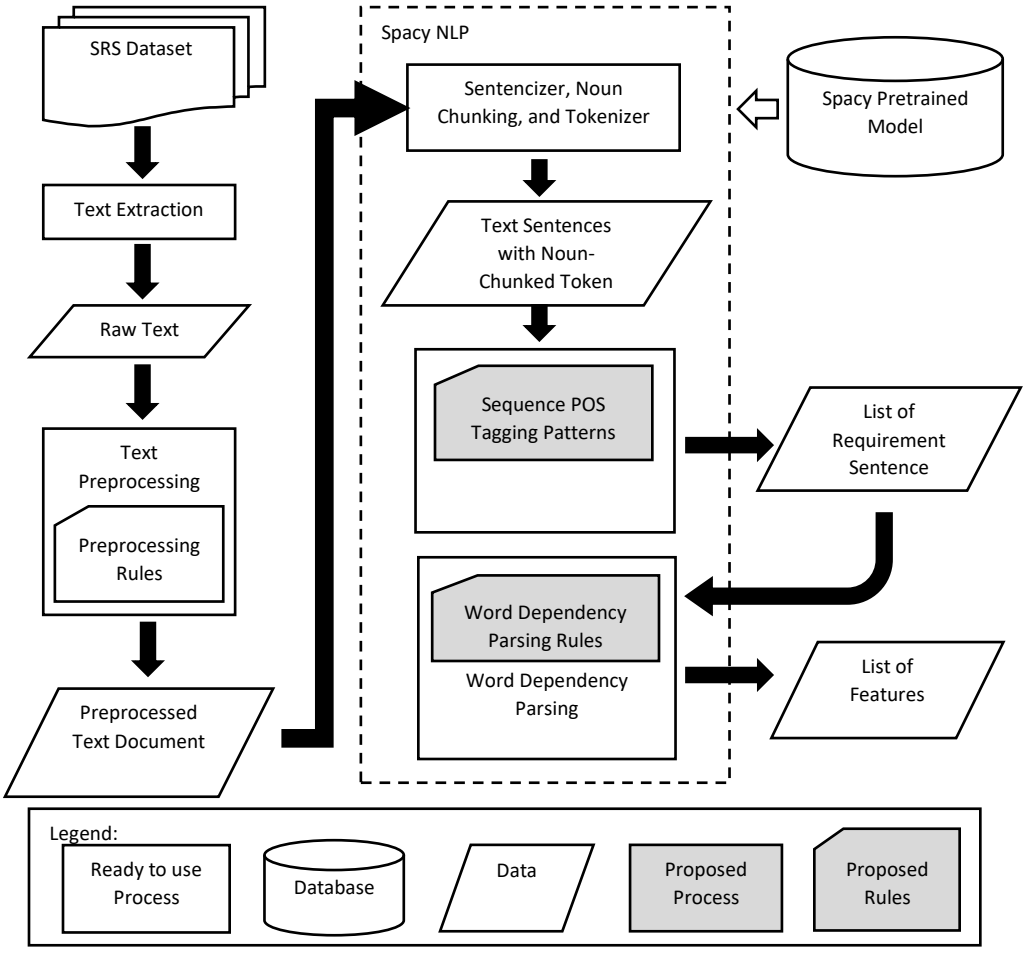

Figure 5. Feature Extraction Automation Framework from SRS Document

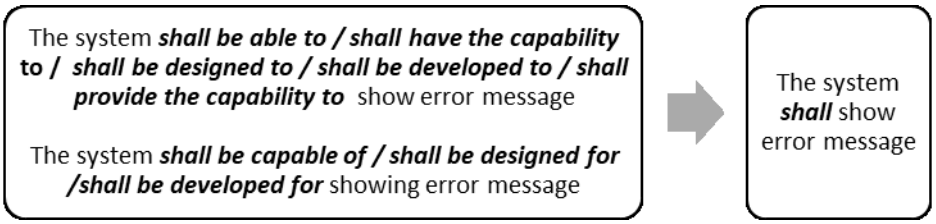

Figure 6. Text Simplification Applied on Text Document Illustration

4.5. Extracting Requirement Sentences from SRS Documents

Requirement sentences written with requirement boilerplate template has a sequence of special words or types of words, which can be used as a reference in determining patterns to recognize them. In this research, there are 2 (two) POS tag modifications, namely SYSNOUN as a system word marker or phrase containing the system word, and KEYVERB as a keyword marker for the keyword of requirement

boilerplate, i.e., the verb "shall". The illustration of the POS tagging result is presented in Figure 7.

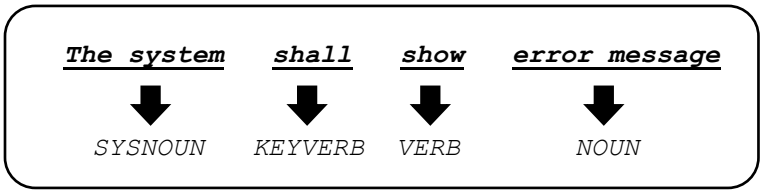

Figure 7. POS Tagging Illustration

Based on 5 (five) SRS documents selected from the PURE dataset, Crime & Criminal Tracking Network and Systems (CCTNS), Digital Home System (DH), e-Store, Laboratory Information System (LIS), and Puget Sound Enhancements (Puget), existing structure in requirement sentences of each document make it possible to formulate 5 (five) POS tagging sequence patterns as follows:

Pattern 1. *SYSNOUN-KEYVERB-VERB-NOUN*
This pattern is used to identify requirement sentences structure as follows:
- The system shall maintain the audit trail for as long as required. [CCTNS]
- The system shall authenticate user credentials to view the profile. [e-Store]
- The system shall log an error message to the external data file. [LIS]

Pattern 2. *SYSNOUN-KEYVERB-VERB-ADP-NOUN*
This pattern is used to identify requirement sentences structure as follows:
- The system shall run on multiple browsers. [CCTNS]
- The e-store system shall communicate to credit management system. [e-Store]
- The system shall clear up data if the user chooses to click the cancel button. [LIS]

Pattern 3. *SYSNOUN-KEYVERB-ADV-VERB-NOUN*
This pattern is used to identify requirement sentences structure as follows:
- The system shall optionally allow user to print the invoice [e-Store]

Pattern 4. *SYSNOUN-KEYVERB-ADV-VERB-ADP-NOUN*
This pattern is used to identify requirement sentences structure as follows:
- The system shall never include in the search result list any record which the user does not have the right to access. [CCTNS]
- The system shall automatically log out all customers after a period of inactivity. [e-Store]

Pattern 5. *SYSNOUN-KEYVERB-PART-VERB-NOUN*
This pattern is used to identify requirement sentences structure as follows:
- The system shall not include such cases in any count of search results, this level of security is normally appropriate for cases dealing with matters such as national security. [CCTNS]
- the system shall not leave any cookies on the customer's computer containing any of the user's confidential information. [e-Store]

In this research, we limit the identification only for system perspective requirement sentences, while the user perspective is excluded at the moment. Passive

form sentences also excluded since we follow the requirement boilerplates syntax guide that shown active form sentences only.

4.6. Extracting Features from Requirement Sentences

After the list of requirement sentences has been successfully extracted from the SRS document, further processing is needed to obtain an abstraction from the system functionality that is described in every sentence. Abstraction in a requirement boilerplate sentence is a combination of words or phrases that adequately represent the meaning of the whole sentence in the context of system functionality. This research assumes that there are no redundancies that need to be eliminated if several requirements sentences represent the same feature, one to one relationship between feature and requirement sentence is applied in this study.

In this stage, word dependency parsing is used to determine the position of a word or phrase in a sentence. The relation to word dependency parsing by spaCy is illustrated in Figure 8.

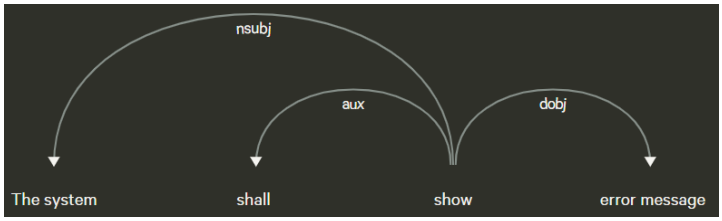

Figure 8. Word Dependency Parsing by SpaCy NLP Illustration

For every parsed sentence, it will always show the process verb of requirement sentences as a root of the dependency, which means that the process verb has no dependency on another word. While the other words are dependent on to process verb as a root. Based on the dependency parsing results for various requirement sentences in the dataset, there are 4 (four) rules that can be formulated to extract their features as follows:

Rule 1. *Verb of Root (dep: root) + Direct Object Noun of Root (dep: dobj)*
This rule is used to identify requirement sentences structure as follow:
*The system shall **show error message** to user.*

Rule 2. *Verb of Root (dep: root) + Preposition (prep) + Prepositioned Object Noun of Root (dep: pobj)*
This rule is used to identify requirement sentences structure as follow:
*The system shall **provide for authentication** to user.*

Rule 3. *(Rule 1 OR Rule 2) + Conjunctive (dep: conj) + Coordinating Conjunctives Noun (dep: cc)*
This rule is used to identify requirement sentences structure as follow:
*The system shall **show warning and confirmation message.***

Rule 4. *Negative Word (dep: neg) + (Rule 1 OR Rule 2 OR Rule 3)*
This rule is used to identify requirement sentences structure as follow:
*The system shall **not leave any cookies** from browsing activity.*

5. Results

After implementing sequence POS tagging patterns on SRS documents, this research automatically produces requirement sentences that are compared with actual requirement sentences from manual extraction. Precision and recall are computed afterward based on the number of results that are True Positive (TP), False Positive (FP), and False Negative (FN) from each SRS document processing. The comparison and accuracy calculations are presented in Table 2.

Table 2. Requirement Sentence Extraction Result Comparison

No	SRS	Actual	Extracted	TP	FP	FN	Precision	Recall
1	CCTNS	53	37	35	2	20	0.95	0.64
2	DH	24	33	21	12	5	0.64	0.81
3	e-Store	96	85	87	0	11	1.00	0.89
4	LIS	27	26	23	3	5	0.88	0.82
5	Puget	38	36	34	3	7	0.92	0.83

Feature extraction rules are applied on list requirement sentences that are already generated from the previous extraction as a continuous process without any refinements or intervention. Therefore, accuracy measurement for feature extraction is only presented as the percentage that shows the success rate of proposed rules to automatically extract features from requirement sentences. Features extraction results are presented in Table 3.

Table 3. Feature Extraction Result

No	SRS	Extracted Requirement Sentences	Extracted Feature	Success Rate (%)
1	CCTNS	37	30	81.08
2	DH	33	27	81.82
3	e-Store	85	56	65.12
4	LIS	26	23	88.46
5	Puget	36	27	75.00

6. Discussion

The findings of this study can be interpreted from the requirement sentence extraction result comparison in Table 2. This study proves the highest precision when applied to the e-Store SRS document with 100% precision value. It means that the proposed method, when it is applied to the e-Store SRS document, did not fail to exclude non-requirement sentences in the extraction process. In the same document, this study shows the highest recall value of 89%. It means that the proposed method for the e-Store SRS document succeeded in extracting 89% of all the requirement sentences that should be obtained. While for the feature extraction results, the success rate varies from the lowest 65% value to 88% value.

This study also found 64% precision value of Digital Home (DH) SRS document that shows 36% of false-positive (FP) extraction results and 64% recall value of Crime & Criminal Tracking Network and Systems (CCTNS) SRS document that shows 36% of false-negative (FN) extraction results. The cause of this extraction's failures can be identified to see the limitation of the proposed method. The examples

of these negative findings are presented in Table 4.

Table 4. Requirement Sentence Extraction Failure Example Analysis

Sentence	Result	Cause
The system owner director shall provide management and communication support (DH)	FP	Using of system phrase and boilerplate structure
The development of the dh system shall use methods and techniques such as … (DH)	FP	Using of system phrase and boilerplate structure
The user interfaces of the system shall comply with standard iso 9241 (CCTNS)	FP	Using of system phrase and boilerplate structure
The System must not exceed <xx> hours per <rolling three months period> (CCTNS)	FN	Using of symbol that not recognized
The System must be able to capture and store violations (CCTNS)	FN	Using conjunction of two process verbs
The system should be developed to be deployed in a 3-tier datacenter architecture (CCTNS)	FN	Using of passive sentence structure

Another finding of this research shows lowest feature extraction result success rate value of 65% on the e-Store and 75% on Puget SRS document. The extraction failure analysis is presented in Table 5.

Table 5. Feature Extraction Failure Example Analysis

Requirement Sentence	Feature	Cause
The system shall enable user to navigate between the search results (e-store)	Enable user	User perspective that is written similarly with system perspective
The system shall enable the user to enter their reviews and ratings (e-store)	Enable the user	User perspective that is written similarly with system perspective
The e store system shall communicate to credit management system (e-store)	commu nicate	“to credit” is detected as verb
The system shall open a pop-up window displaying information (e-store)	open	“a pop-up window” is not detected a noun phrase
The system shall allow actors to delete recorded clips (puget)	allow	User perspective that is written similarly with system perspective

These findings show the limitations of the proposed framework on certain cases, mostly caused by requirement sentence inconsistency and NLP library limitation on recognizing the words that have the same form for noun and verb or certain noun phrases that are failed to be recognized. But with requirement boilerplate template usage consistency for requirement sentence writing in the SRS document and avoiding the use of words that can cause ambiguity, the proposed method will work as expected.

7. Conclusion

This study gives a logical answer for all the research questions stated in subsection 4.1. The first research question about what techniques need to be

constructed to automatically extract features directly from the SRS document is answered by the proposed framework of automatic feature extraction using the NLP approach that is presented in Figure 5. Second research question about what rules of processing that need to be formulated to automatically extract features directly from the SRS document, is answered by sequence POS tagging patterns and word dependency parsing rules that are presented in subsection 4.5 and 4.6. The third research question on accuracy measurement is presented in the result section. First accuracy measurement on requirement sentence extraction from SRS document with precision value in the range of 64% to 100% and recall value in the range 64% to 89%. Second accuracy measurement on feature extraction from previously extracted requirement sentences with a success rate from 65% to 88%.

For the next study, technique extension is needed to be done to obtain further stage on domain engineering of SPLE. Classification on the list of features is needed to build a feature catalog as an intermediate artifact to build the feature model (FM). Mandatory features and optional features are also needed to be distinguished. Since there are many studies on automatic or semi-automatic feature model generation, so this research can be combined to produce a complete solution to automatically extract feature model from the SRS document directly.

References

[1] I. Sommerville, *Software Engineering 10th Edition*, Tenth Edition. Harlow: Pearson Education Limited, 2016.

[2] S. Al Busaidi and N. Kraiem, "Using Software Product Line Application in Enterprise Resources Planning Systems Systematic Literature Review," *Comput. Eng. Inf. Technol.*, vol. 06, no. 03, 2017.

[3] T. Berger *et al.*, "A survey of variability modeling in industrial practice," *Proc. Seventh Int. Work. Var. Model. Software-intensive Syst.*, vol. January, pp. 1–8, 2013.

[4] J. Dick, E. Hull, and K. Jackson, *Requirements Engineering*, Fourth Edi. Cham: Springer International Publishing, 2017.

[5] S. Apel, D. Batory, C. Kästner, and G. Saake, *Feature-Oriented Software Product Lines*. Springer-Verlag Berlin Heidelberg, 2013.

[6] R. AL-Msie'Deen, M. Huchard, A. D. Seriai, C. Urtado, and S. Vauttier, "Reverse engineering feature models from software configurations using formal concept analysis," *CEUR Workshop Proc.*, vol. 1252, pp. 95–106, 2014.

[7] F. Benbassat, P. Borba, and L. Teixeira, "Safe evolution of software product lines: Feature extraction scenarios," *Proc. - 2016 10th Brazilian Symp. Components, Archit. Reuse Software, SBCARS 2016*, pp. 11–20, 2016.

[8] M. A. Laguna and Y. Crespo, "A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring," *Sci. Comput. Program.*, vol. 78, no. 8, pp. 1010–1034, 2013.

[9] E. Stankov, M. Jovanov, and A. M. Bogdanova, "Source code similarity detection by using data mining methods," *35th Int. Conf. Inf. Technol. Interfaces, ITI 2013*, pp. 257–262, 2013.

[10] C. Kastner, A. Dreiling, and K. Ostermann, "Variability mining: Consistent semi-automatic detection of product-line features," *IEEE Trans. Softw. Eng.*,

vol. 40, no. 1, pp. 67–82, 2014.

[11] W. Fenske, J. Meinicke, S. Schulze, S. Schulze, and G. Saake, "Variant-preserving refactorings for migrating cloned products to a product line," *SANER 2017 - 24th IEEE Int. Conf. Softw. Anal. Evol. Reengineering*, pp. 316–326, 2017.

[12] W. K. G. Assunção, S. R. Vergilio, and R. E. Lopez-Herrejon, "Automatic extraction of product line architecture and feature models from UML class diagram variants," *Inf. Softw. Technol.*, vol. 117, no. September, 2020.

[13] M. Mefteh, N. Bouassida, and H. Ben-Abdallah, "Implementation and evaluation of an approach for extracting feature models from documented UML use case diagrams," *Proc. ACM Symp. Appl. Comput.*, vol. 13-17-Apri, pp. 1602–1609, 2015.

[14] R. S. Pressman and B. R. Maxim, *Software Engineering - A Practitioner's Approach*, Ninth Edition. New York: McGraw-Hill Education, 2020.

[15] R. F. Schmidt, *Software Engineering: Architecture-driven Software Development*, vol. 53, no. 9. Morgan Kaufmann, 2013.

[16] A. Mili and F. Tchier, *Software Testing Concepts and Operations*. New Jersey: John Wiley & Sons, 2015.

[17] M. Mefteh, N. Bouassida, and H. Ben-Abdallah, "Mining feature models from functional requirements," *Comput. J.*, vol. 59, no. 12, pp. 1784–1804, 2016.

[18] M. Hamza and R. J. Walker, "Recommending features and feature relationships from requirements documents for software product lines," *Proc. - 4th Int. Work. Realiz. Artif. Intell. Synerg. Softw. Eng. RAISE 2015*, pp. 25–31, 2015.

[19] N. H. Bakar, Z. M. Kasirun, and N. Salleh, "Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review," *J. Syst. Softw.*, vol. 106, pp. 132–149, 2015.

[20] V. Alves *et al.*, "An exploratory study of information retrieval techniques in domain analysis," *Proc. - 12th Int. Softw. Prod. Line Conf. SPLC 2008*, pp. 67–76, 2008.

[21] J. M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans, "Feature model extraction from large collections of informal product descriptions," *2013 9th Jt. Meet. Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng. ESEC/FSE 2013 - Proc.*, pp. 290–300, 2013.

[22] A. Ferrari, G. O. Spagnolo, and F. Dell'Orletta, "Mining commonalities and variabilities from natural language documents," *ACM Int. Conf. Proceeding Ser.*, pp. 116–120, 2013.

[23] N. Niu, J. Savolainen, Z. Niu, M. Jin, and J.-R. C. Cheng, "A Systems Approach to Product Line Requirements Reuse," *IEEE Syst. J.*, vol. 8, no. 3, pp. 827–836, Sep. 2014.

[24] K. Pohl and C. Rupp, *Requirements Engineering Fundamentals*, 2nd ed. Rocky Nook Inc., 2015.

[25] C. Arora, M. Sabetzadeh, L. C. Briand, and F. Zimmer, "Requirement boilerplates: Transition from manually-enforced to automatically-verifiable natural language patterns," *2014 IEEE 4th Int. Work. Requir. Patterns, RePa 2014 - Proc.*, pp. 1–8, 2014.

[26] J. Carbonnel, M. Huchard, and C. Nebut, "Modelling equivalence classes of feature models with concept lattices to assist their extraction from product descriptions," *J. Syst. Softw.*, vol. 152, pp. 1–23, 2019.

[27] M. Acher *et al.*, "On extracting feature models from product descriptions," *Proc. Sixth Int. Work. Var. Model. Software-Intensive Syst.*, vol. VaMoS '12, pp. 45–54, 2012.

[28] C. Palomares, C. Quer, and X. Franch, "Requirements reuse and requirement patterns: a state of the practice survey," *Empir. Softw. Eng.*, vol. 22, no. 6, pp. 2719–2762, 2017.

[29] F. Nasser, A. Al Omran, and C. Treude, "Choosing an NLP Library for Analyzing Software Documentation : A Systematic Literature Review and a Series of Experiments," 2017.

[30] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To Appear*, 2017.

[31] N. Lindblad *et al.*, "Natural Language Processing of Textual Requirements," *ICONS 2015 Tenth Int. Conf. Syst. Nat.*, vol. 9, no. 3, pp. 93–97, 2015.

[32] A. Sree-Kumar, E. Planas, and R. Clarisó, "Extracting software product line feature models from natural language specifications," *ACM Int. Conf. Proceeding Ser.*, vol. 1, pp. 43–53, 2018.

[33] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "PURE: A Dataset of Public Requirements Documents," *Proc. - 2017 IEEE 25th Int. Requir. Eng. Conf. RE 2017*, no. 1, pp. 502–505, 2017.