TEST Engineering & Management

November-December 2019
ISSN: 0193-4120 Page No. 3061- 3066

A Tool for Detecting Ambiguity in Software Requirements Specification

Abdirashid Ali Isse, *Sa'adah Hassan
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia
Serdang, Selangor, Malaysia
apdyrashid10@gmail.com, *saadah@upm.edu.my

Article Info
Volume 81
Page Number: 3061- 3066
Publication Issue:
November-December 2019

Article History
Article Received: *5 March 2019*
Revised: *18 May 2019*
Accepted: *24 September 2019*
Publication: *14 December 2019*

Abstract:
The main goal of requirements engineering is to establish software requirements specification (SRS). The requirements in SRS are mostly specified in natural languages (NL), therefore, one of the common problems of SRS is requirements ambiguity. The requirement is said as ambiguous when it has more than one interpretation, subsequently, can lead to requirements inconsistency and conflict. Besides, to detect ambiguous requirements manually is time-consuming and tedious process. Thus, this paper presents a tool called SRS Ambiguity Detector, that able to detect automatically the major types of ambiguity; lexical, syntactic and syntax ambiguity. This tool uses ambiguity words from the ambiguity handbook to detect lexical ambiguity, while, parts of speech (POS) tagging technique has been applied to detect syntactic and syntax ambiguities. Evaluation was conducted to assess the effectiveness, and the result has shown that the proposed tool able to identify more ambiguous requirements as compared to manual detection.

Keywords: *requirements engineering, ambiguity detection, SRS*

1. INTRODUCTION

Requirements engineering (RE) is a systematic process to elicit, analyse, model, and document requirements for the software to be developed. Software Requirements Specification (SRS)is the final output of RE and typically, the requirements in SRS are written in natural languages (NLs) format, such as English language (Fockel and Holtmann, 2015). However, major issue of NLs documents is ambiguous. Ambiguity can be defined as the possibility to understand a phrase or word in several different meanings (Gill et al., 2014). Therefore, it is crucial in RE because ambiguous requirements will be interpreted in different meanings for the same requirements, thus, it can cause requirements inconsistency and conflict. Moreover, performing ambiguity review against SRS is a time-consuming and tedious job. Especially when dealing with a large number of requirements. There are limited tools to support ambiguity detection, and the tools focuses on certain types of ambiguity only (e.g., Gleich et al., 2010; Umber et al., 2011; Sabriye and Zainon, 2017).

The purpose of this paper is to present a tool to help detecting ambiguity of natural language requirements in SRS document. The proposed tool focuses on three main types of ambiguities,

November-December 2019
ISSN: 0193-4120 Page No. 2732 - 2742

which are lexical, syntax, and syntactic. The following sections discuss on related work, continue with descriptions of the proposed tool, evaluation, and finally the conclusions.

2. RELATED WORK

Ambiguity in NLs SRS documents is unavoidable. The ambiguity in NLs can be categorized into: lexical, syntactic, syntax, semantic, and pragmatic ambiguities (Berry, et al., 2003). Lexical ambiguity is in which one word or phrase has several interpretations (Bano, 2015). For example, *the users of the system are administrators and customers. They need to log in to the system.* The term "they" is an ambiguous word because of unclear of reference: it can be either the *customers* or the *administrators*, or it can be both that need to log in to the system. While, syntactic ambiguity occur when the sentence contains vague words, such as, *quickly*, and *accurate* (Gleich et al., 2010). As for examples, "*the system must accept accurate information only*", is an ambiguous requirement because the term "accurate" are vague word that can cause syntactic ambiguity. Moreover, syntactic ambiguity also happens when a sentence contains more than one logical condition such as *and* and*or*. As for semantic or scope ambiguity, it appears when sentence has several understandings based on its scenario without containing lexical, syntax, and syntactic ambiguities and mostly occurs when the scope of all things include the scope of one thing (Sabriye and Zainon, 2017). For example: "*all citizens have myKad"*. This example can be: *"Every citizen has an individual myKad"*or*"All citizens have the same myKad"*. Pragmatic ambiguity is type of ambiguity focuses on the relationship between the meaning of the sentences and its environment or context. It depends on the context of the requirements, including the background of the reader. For example, two different readers with different background education can interpret the requirements into two different ways (Gleich et al., 2010). In addition, syntax ambiguity(Nigam, 2012), this ambiguity is a specific type of error, which occurs, if the sentence does not end with a full stop (.), or if the sentence is a passive voice: for example, "*the records of the customers must be documented*." is a syntax ambiguity because the doer of the action is missed.

Many researchers addressed ambiguity problems in NLs SRS documents using different natural language processing (NLP) based techniques and tools. For example, work done by Beg *et al*., (2008) proposed NLP based tool and technique for detecting lexical ambiguity in NLs SRS documents. The weakness of this tool is that it detects ambiguity efficiently if the requirements in NLs SRS document contain no more than 6 words. RESI(Korner and Brumm, 2009)isa tool developed to help software analyst to work on software requirements documents. The tool able to present tothe user the ambiguous requirements in the document and provides the possible meaning of each word. This feature will help analyst to make changes to the document easily. This tool used POS tagger technique in order to detect ambiguities. However, the main drawback of this tool is that it only focuses on nouns and verbs. While, Gleich*et al*. (2010) proposed an automated ambiguity detection tool to detect lexical and syntactic ambiguities, as well as provides an explanation about the source of the ambiguity. The tool refers to a list of ambiguity words in the ambiguity handbook in order to detect ambiguous words. Whereas, Gulia and Choudhury (2016) have proposed a tool that analysed requirements specification using standard POS tagger and parser tool to create sequence and activity diagrams. This tool helps users to generate different diagrams (i.e., activity and sequence diagrams) from the same requirements specification in order to reduce the ambiguity. Recent work is by Sabriye and Zainon(2017), in which they have developed a tool to detect two types of ambiguities: syntactic and syntax ambiguity using a part of speech (POS) tagging technique. A summary of related work on tools to detect ambiguity in NLs SRS documents is as shown in Table 1.

Published by: The Mattingley Publishing Co., Inc.

TEST Engineering & Management

November-December 2019
ISSN: 0193-4120 Page No. 3061- 3066

Table 1. A Summary of tools that support ambiguity detection in NL's requirements

Tool/Work	Type of Ambiguity	Technique	Features/Limitations
A method to deal with lexical ambiguity in SRS (Beg *et al*., 2008).	Lexical	single algorithm created in their research.	Limited to 6 words only
RESI Tool (Korner and Brumm, 2009).	noun and verb	POS tagging	- It presents the users a dialog system as soon as the document is ambiguity. - It provides the meaning of each noun and verb, so it easy analyst to change.
Automatic ambiguity detection tool in SRS (Gleich*et al*., 2010).	Lexical and Syntactic	POS tagging	- Only cover 2 types of ambiguity. - It also provides an explanation about the source of the ambiguity
SR-Elicitor (Umber, *et al*., 2011)	Lexical, Syntactic and Semantic	POS tagging	to record and automatically transform the natural language software requirements to a controlled representation using SBVR.
Ambiguity Detector Tool (Nigam *et. al*, 2012)	Lexical, Syntax and Syntactic	POS tagging and referring corpus of ambiguous words.	POS tagger is used for those words that matched only.
Automatic Ambiguity detector (SabriyeandZainon, 2017)	Syntax and Syntactic	POS tagging	Only cover 2 types of ambiguity.

Based on the study, we propose SRS Ambiguity Detector Tool to detect three main types of ambiguities, namely: lexical, syntax and syntactic. The common technique used for the detection, which is POS tagging technique, is also adopted in the proposed tool. The proposed tool aims to tackle the limitations of the previous work.

3. SRS AMBIGUITY DETECTOR TOOL

The scope of this research is limited to detect ambiguities in English language SRS. The proposed tool detects lexical, syntactic and syntax ambiguities by using a POS tagging technique and by referring ambiguity handbook (Berry, *et al.*, 2003). SRS ambiguity detector tool was developed using Visual basic studio and C# Programming language. This section discusses in detail on how the proposed tool detect ambiguities in NLs SRS. Figure 1 presents the flowchart of the proposed tool and the explanation of each step as follows:

Phase 1 – User can upload document (i.e., SRS) into the system. The document can be a collection of sentences or paragraphs in English language and in .txt format. User will be able to browse the uploaded document for confirmation. The system will read the document until reach end of document.

Phase 2 – Each word of the uploaded document is compared with words in Ambiguity Handbook (AHB) dictionary(Berry, *et al*., 2003).Dictionary is a repository that stores a large number of words from AHB is referred to detect lexical ambiguity. Word in the sentence that is match with the words inside dictionary will be counted and recognized as **lexical** ambiguity. At the same time, each sentence of the document are also marked to the *part of speech (*POS) tagger technique which assigns each word of the document the matching tag of POS tagger. POS tagging is a technique which assigns every word in a given sentences into a predefined corresponding parts of speech in English. This technique used to detect syntactic and syntax ambiguity. For **syntactic** ambiguity, it checks if the sentence marked with the POS tagger contains adjective or adverb. While, for **syntax** ambiguity, it check if the document tagged with POS tagger contains passive voice and if the sentence does not end with afull stop (.). The system will then compute the total number of lexical, syntactic and syntax

November-December 2019
ISSN: 0193-4120 Page No. 2732 - 2742

ambiguities. This system checks every word and sentences in the SRS document and marks each type of ambiguity with specific colour. In which, lexical ambiguities are marked as red, syntactic as blue, and syntax as green colour.

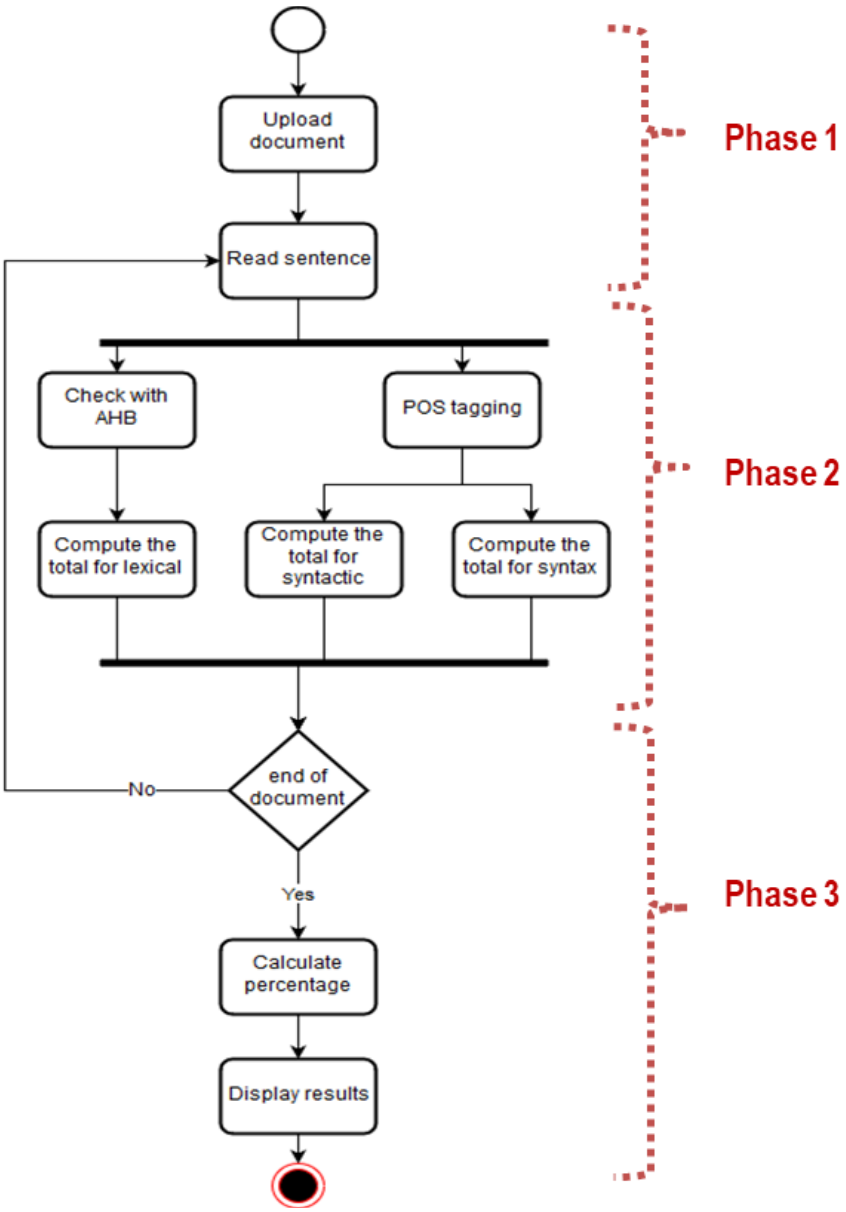

Figure 1.Theflowchart for SRS Ambiguity Detector

Phase 3 – The tool will read the next sentence and repeat the detection process until reach end of document. If end of document is reached, then it will calculate the total number and percentage of ambiguity detected for each type of ambiguity. The results (the detected ambiguous requirements) will be displayed in different colours accordingly, in which, red for lexical ambiguity, blue for syntactic ambiguity, and green for syntax ambiguity. The results are also displayed as graph. Figure 2 shows the screenshot of the proposed tool and an example of the output displayed.

November-December 2019
ISSN: 0193-4120 Page No. 3061- 3066

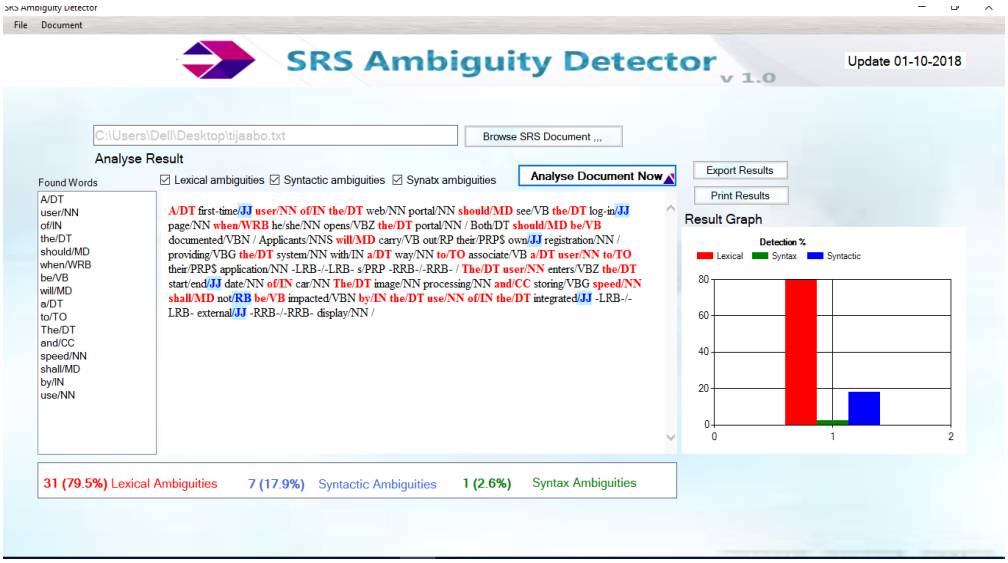

Figure 2. SRS Ambiguity Detector Tool

4. EVALUATION

Evaluation was carried out to assess the effectiveness of the proposed tool, in which, comparison was made between manual detection with automated detection using the proposed tool. The material used for this evaluation was based on Gleich *et al*., (2010). A dataset which contains 20 software requirements written in English that consists of 4 lexical, 4 syntactic, 4 syntax, 4 mixed of lexical, syntactic as well as syntax, and 4 unambiguous software requirements, has been used for the evaluation.

For manual detection, 33 participants, mostly software developer and software analyst, were involved. Explanation and examples of each type of ambiguity were given to the participants prior the evaluation. The dataset was given to all the participants and each participant need to determine whether each requirement in the dataset is lexical, syntactic, syntax, mixed of lexical, syntactic and syntax, or unambiguous. While, for automated detection, the same dataset have been uploaded into the tool. The tool automatically analysed the input and presents the output.

The results from manual detection showed an average 34% were detected correctly by the participants, while the tool managed to detect 100% all type of ambiguous requirements in the dataset. In addition, participants also mentioned that detecting ambiguity manually was very difficult, especially, when the requirement contains mixed types of ambiguities.

5. CONCLUSION

Requirements engineering is one of the crucial phase in software development life cycle. Wherein, the rest of software development phases depend on the requirements, and if the requirements are not correctly stated in the SRS, then it will give tremendous impact to the success of the software. One of the common issues is ambiguity in SRS that can causes different interpretations, which has ultimately; affects the quality of the software to be developed. This issue can be solved by detecting ambiguities at the early phase of software development. Therefore, a tool to detect ambiguities in SRS is presented to improve the quality of SRS. The SRS ambiguity detector tool was developed to detect the main types of ambiguities namely: lexical, syntax and syntactic. Besides, the tool will reduce the amount of effort required for ambiguity reviewing process. The results from the evaluation highlight the effectiveness of the proposed tool. For the next stage, we aim to add

November-December 2019
ISSN: 0193-4120 Page No. 2732 - 2742

more features to the tool that will facilitate the reviewing process.

6. ACKNOWLEDGEMENT

We wish to acknowledge financial support from Universiti Putra Malaysia.

7. REFERENCES

1. Bano, M. 2015. Addressing the challenges of requirements ambiguity: A review of empirical literature.In Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), 2015 IEEE (pp. 21-24).IEEE.
2. Berry, D. M., E. Kamsties, and M. M. Krieger.2003. From contract drafting to software specification: Linguistic sources of ambiguity. A Handbook.
3. Beg, R., Abbas, Q., and Joshi, A. 2008. A method to deal with the type of lexical ambiguity in a software requirement specification document. In First International Conference on Emerging Trends in Engineering and Technology, 2008. ICETET'08. (pp. 1212-1215). IEEE.
4. Fockel, M., andHoltmann, J. 2015.ReqPat: Efficient documentation of high-quality requirements using controlled natural language. In 23rd International Requirements Engineering Conference (RE), 2015 IEEE (pp. 280-281). IEEE.
5. Gill, K. D., Raza, A., Zaidi, A. M., andKiani, M. M. 2014.Semi-automation for ambiguity resolution in Open Source Software requirements. In 27th Canadian Conference on Electrical and Computer Engineering (CCECE), 2014 IEEE (pp. 1-6). IEEE.
6. Gleich, B., Creighton, O., andKof, L. 2010. Ambiguity detection: Towards a tool explaining ambiguity sources. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 218-232). Springer: Berlin, Heidelberg.
7. Gulia, S. and Choudhury, T. 2016.An Efficient Automated Design to Generate UML Diagram from Natural Language Specifications.6th International Conference on Cloud System and Big Data Engineering (Confluence), IEEE, pp.641-648.
8. Korner, S. J., andBrumm, T. 2009. Resi-a natural language specification improver.In International Conference on Semantic Computing, 2009.ICSC'09. IEEE (pp. 1-8). IEEE.
9. Nigam A, Arya N, Nigam B, Jain D, Tool for Automatic Discovery of Ambiguity in Requirements, *IJCSI International Journal of Computer Science*, vol. 9, no. 5, pp. 350-356, 2012.
10. Sabriye, A. O. J. A., andZainon, W. M. N. W. 2017.A framework for detecting ambiguity in software requirement specification.In the Proceeding of 8th International Conference on Information Technology (ICIT), pp. 209-213. IEEE.
11. Umber, A., Bajwa, I. S., andNaeem, M. A. 2011.NL-based automated software requirements elicitation and specification. In International Conference on Advances in Computing and Communications (pp. 30-39). Springer: Berlin, Heidelberg.