UNIVERSITI PUTRA MALAYSIA

A TOOL FOR DETECTING AMBIGUITY IN SOFTWARE REQUIREMENT SPECIFICATION

ABDIRASHID ALI ISSE

FSKTM 2019 12

A TOOL FOR DETECTING AMBIGUITY IN SOFTWARE REQUIREMENT SPECIFICATION

By

ABDIRASHID ALI ISSE

This dissertation submitted to School of Graduate Studies, Universiti Putra Malaysia, in Fulfillment of the Requirement for the Degree of Master of Software Engineering

January 2019

DEDICATION

To:

This dissertation is dedicated to my beloved uncle (Abdi Isse Said) for his endless support and motivation (encouragement).

Abstract of dissertation presented to the Senate of Universiti Putra Malaysia in fulfillment of the requirement for the degree of Master of Software Engineering

A TOOL FOR DETECTING AMBIGUITY IN SOFTWARE REQUIREMENT SPECIFICATION

ABSTRACT

This thesis is about detecting ambiguities in software requirements' specification (SRS). Specifically, most of the software requirement documents are written in Natural languages (NLs). NLs are basically ambiguous. Ambiguity is a statement of requirements, which have more than one interpretation. However, Ambiguity can be considered as an issue of software requirement documents because it can lead the software developers to develop software, which is different what the customers' need. The aim of this research is to propose a tool which detects lexical, syntactic and syntax ambiguities in SRS. In this thesis, ambiguity words from the ambiguity handbook have been used to detect lexical ambiguity. In parallel, Parts of speech (POS) tagging technique has been applied to detect syntactic and syntax ambiguous. The proposed tool was evaluated in order to check its performance by comparing human detection capacity and the proposed tool. The aim of this evaluation also was to see if the humans face complexities in detecting ambiguity in SRS, and the result shows that the humans have difficulties detecting ambiguities in SRS compared to the proposed tool, particularly, lexical ambiguity and requirements that contains lexical, syntactic and syntax ambiguities in one sentence. The proposed tool can facilitate both software analysts and developers to detect the ambiguities in software requirements' specification easily.

Abstrak tesis yang dikemukakan kepada Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Sarjana Kejuruteraan Perisian

TOOL UNTUK MENGESAN KEKABURAN DALAM REQUIREMENT SPECIFICATION

ABSTRAK

Tesis ini adalah untuk mengesan kekaburan dalam s*oftware requirement specification* (SRS). Secara spesifik, kebanyakan dokumen keperluan perisian ditulis dalam *Natural Languanges* (NLs). NLs secara asasnya adalah kabur. Kekaburan berlaku apabila pernyataan keperluan mempunyai lebih satu daripada makna. Kekaburan merupakan satu isu dalam dokumen keperluan perisian kerana ia boleh menyebabkan pembangun perisian membangunkan perisian yang berbeza daripada keperluan pengguna. Tujuan kajian ini ialah untuk mencadangkan satu alat untuk mengesan kekaburan leksikal, sintaktik and sintak dalam SRS. Di dalam tesis ini, perkataan-perkataan kabur daripada *ambiguity handbook* digunakan untuk mengesan kekaburan leksikal. Pada masa yang sama, teknik penandaan *Parts of speech* (POS) telah digunakan untuk mengesaan kekaburan sintaktik and sintak. Prestasi alat yang dicadangkan akan dinilai dengan membuat perbandingan keupayaan mengesan kekaburan antara manusia dan alat yang dicadangkan. Penilaian ini juga bertujuan untuk melihat sekiranya manusia mengalami kesukaran dalam mengesan kekaburan dalam SRS berbanding alat yang dicadangkan, terutamanya, kekaburan leksikal dan pernyataan keperluan yang mengandungi kekaburan leksikal, sintaktik dalam satu ayat. Alat yang dicadangkan akan memudahkan penganalisis dan pembangun perisian untuk mngesan kekaburan dalam SRS dengan mudah.

ACKNOWLEDGEMENT

In the name of ALLAH, the Beneficent, the Compassionate, thanks and praise to God for giving me strength and patience to complete my duties successfully.

I would like to sincerely thank and express my deep thanks and gratitude to my supervisor Dr. Sa'adah Hassan for her endless support, guidance, correction, encouragement, advice and valuable observations in my Master's degree.

Also, I would like to acknowledge the support of my family and parents (especially to my uncle Abdi Isse said, who gave me fully financial support during this study), sisters and brothers. They always give motivation and pray for successfully graduation of this study.

Finally, I thanks the contribution of my friends such as Mohamed nor and Ali Olow Jimale (Somalia), Saiful Bahri Bin Hisamudin (Malaysia) for their great support and others who don't mention here their names I will never forgotten their supports, thanks allot.

Finally, to all UPM staff, thanks you for your facilitation. I would like to acknowledge to any individual who are not mentioned here for his/her irreplaceable helps and cooperation.

.

APPROVAL

Thesis submitted to the Senate of University Putra Malaysia and has been accepted as fulfillment of the requirement for the degree of Master of Software Engineering.

_____

Supervisor,

Sa'adah Hassan, PhD.

Department of Software Engineering and Information Systems

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

January, 2019

DECLARATION

I hereby confirm that:

- this thesis is my original work;
- quotations, illustrations and citations have been duly referenced;
- this thesis has not been submitted previously or concurrently for any other degree at any other institutions;
- intellectual property from the thesis and copyright of thesis are fully-owned by Universiti Putra Malaysia, as according to the Universiti Putra Malaysia (Research) Rules 2012;
- written permission must be obtained from supervisor and the office of Deputy Vice-Chancellor (Research and Innovation) before thesis is published (in the form of written, printed or in electronic form) including books, journals, modules, proceedings, popular writings, seminar papers, manuscripts, posters, reports, lecture notes, learning modules or any other materials as stated in the Universiti Putra Malaysia (Research) Rules 2012;
- there is no plagiarism or data falsification/fabrication in the thesis, and scholarly integrity is upheld as according to the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) and the Universiti Putra Malaysia (Research) Rules 2012. The thesis has undergone plagiarism detection software.

Signature: _____

Date:

Name and Matric No.:

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NLP	NATURAL LANGUAGE PROCESSING
UML	UNIFIED MODELING LANGUAGE
RE	REQUIREMENTS ENGINEERING
OOA	OBJECT ORIENTED ANALYSIS
NLs	NATURAL LANGUAGEs
POS	PART OF SPEECH
RQA	REQUIREMENTS QUALITY ANALYZER
SRS	SOFTWARE REQUIREMENTS SPECIFICATION
UPM	UNIVERSITI PUTRA MALAYSIA

CHAPTER 1: INTRODUCTION

1.1 Background

At the moment, software systems become the backbone of every organization such as: education, manufacturing, governments, social networking, insurance, banking and health care. Developing software system requires: time, cost, tools, idea, infrastructure and experts. Lack of correct requirements can cause the failure of the project.

The success of any software development depends on how it meets and answers the needs of its stakeholders (Cheng and Atlee, 2007). Requirement engineering (RE) becomes the backbone of establishing the needs of the stakeholders. RE is the process of understanding the system's requirements through predefined activities of eliciting the needs of the users by: gathering, analyzing, modeling, validating, documenting and managing the requirements. It involves a systematic process of defining, analyzing, modeling, evaluating and documenting the requirement of the system and the users in which the system will use (Inayat at el, 2015).

A payroll system was designed in 1970s, it was developed to store the last two digits of the year rather than four digits for the purpose of the saving the memory space. Conversely, the year of Millennium (year 2000) a bug happened. To fix that bug, hundreds of US dollars were spent (Nigam et al., 2012a).Moreover, ERP software failure in Jordan: the cause of this problem was due to unmatched the assumptions of the problem with system requirements. It means the software requirements engineers don't executed the system as expected and the issues of this system that is sizeable gap between the system requirements and assumptions was built in the project caused the loss of capital and immoral of the clients. A recent survey, participated more than 800 information systems managers showed that more than 60% of the projects failed different issues. Another example is automated airport baggage handling system failure: this

project failed to forecast how many carts are properly resulted in interruptions in picking up bags and that means this system failed to meet the requirements of the system and lastly, cause monthly of correction which caused exceeded the time and the cost (Ribeiro, 2014).

SRS document is a crucial document which contains system and user's requirements with their descriptions. It is very important since it describes the stakeholder's requirements together with the system modeling. Briefly, it is a contract among stakeholders to clarify the key contents about the software in order to design and develop (Anuar, Ahmad and Emran, 2015). Moreover, SRS is an agreement report as well as written by the stakeholders of the developed system.

There a lot of problems cause natural languages (NLs) in software requirements specifications (SRS) such as incomplete, incorrect inconstancies, and ambiguity (Shah and Patel, 2014). Furthermore, this study focus on ambiguity of natural language in SRS, specially three types of ambiguities which are lexical, syntax and syntactic.

NL is the human language which is used to describe software specification such as English language (Fockel and Holtmann, 2015). The majority of NL documents are ambiguous. (Sabriye and Zainon, 2017) Ambiguity is the possibility to understand a phrase or word in several different means. The purpose of this study is to propose a tool which detects ambiguity in software specification documents using words from ambiguity handbook as dictionary.

1.2 Problem statements

The failed software development projects have become commonplace, due to, incomplete and ambiguous software requirement specification (SRS), since most of the requirement documents written in natural language (NL), (Bano, M. 2015). Ambiguity considered as a challenge of the SRS than other requirements' defects which have more frequently results in misunderstanding (Bano, M. 2015). Detecting and addressing ambiguity during RE can be more cost-effective than fixing it at later stages of development. Multiple interpretations of the requirements can lead to incorrect implementation, especially in case of unacknowledged ambiguity (Haron and Ghani, 2015).

In addition to that, several researchers tried to solve the issues of the requirements specification ambiguities using different tools and techniques including UML (unified modeling language) based techniques, ontology based techniques and NLP based techniques (Sabriye and Zainon, 2017). However, until now, Syntactic and Syntax ambiguities are more prominent than other types of ambiguities such as lexical (Bano, M. 2015). Moreover, there is limited research tried to solve lexical, syntax and syntactic ambiguities using POS tagging technique and dictionary with ambiguity handbook.

1.3 Research objectives

To overcome the mentioned problems, the main objectives of this research are:

i) To formulate ambiguity detection based on the existing approaches

ii) To propose a tool for detecting requirement ambiguity in requirement specification

iii) To evaluate the effectiveness of the proposed tool

1.4 Scope of the study

The scope of this research is limited detecting syntax, syntactic and lexical ambiguities in SRS. This study consists of three phases: Preprocessing phase: which are uploading SRS documents from users in English as input, Processing phase: which focusing on processing the SRS documents by reads the NL text line by line and Post-processing phase: which displays the detected lexical, syntactic, and syntax ambiguities as output. In order to detect ambiguities in SRS, this research proposed a tool which detects lexical, syntactic and syntax ambiguities by using a POS tagging technique with dictionary. In addition to that, the research used ambiguity words from ambiguity handbook to mark lexical ambiguities in SRS by stored these words in a dictionary. In parallel, POS tagging technique was applied to detect syntactic and syntax ambiguities.

1.5 Dissertation organization

The remaining chapters of this research are ordered as follows:

CHAPTER 2 discusses general overview about requirement engineering and its activities. Also the chapter discusses how software requirements are documented in NL format. In addition to that, this unit also covered the related work of the study and listing Natural language Processing Tools for detecting ambiguity in software requirements and finally summarises the chapter.

CHAPTER 3 presents the overall methodology used to conduct this research in order to achieve the main goal of the research. It contains two main parts: first part provides the general overview of the methodology and divided into five phases by providing short explanations of each phase. Second parts, discusses in detail the components and how covered each phase.

CHAPTER 4 this unit also contains two main sections: explaining the framework of the proposed tool in this study and implementation section of the framework of the proposed tool.

CHAPTER 5 provides the evaluation of the study or research. This evaluation conducted used same documents into the human and the tool itself, in order to see the human capacity of detecting ambiguity in the SRS. Then the result from the human test compared the result generated the developed tool are takes on.

CHAPTER 6 gives explanation about the conclusion, future work and recommendations. It also provides the achievements of the study such: objectives, research questions and the limitations of the research are covered.

References

1. Bettini, L., & Crescenzi, P. (2015, July). Java-meets eclipse: An IDE for teaching Java following the object-later approach. In Software Technologies (ICSOFT), 2015 10th International Joint Conference on (Vol. 2, pp. 1-12). IEEE.
2. Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. Computers in human behavior, 51, 915-929
3. Sabriye, A. O. J. A., & Zainon, W. M. N. W. (2017, May). A framework for detecting ambiguity in software requirement specification. In Information Technology (ICIT), 2017 8th International Conference on (pp. 209-213). IEEE.
4. Anuar, U., Ahmad, S., & Emran, N. A. (2015, December). A simplified systematic literature review: Improving Software Requirements Specification quality with boilerplates. In Software Engineering Conference (MySEC), 2015 9th Malaysian (pp. 99-105). IEEE.
5. Fockel, M., & Holtmann, J. (2015, August). ReqPat: Efficient documentation of high-quality requirements using controlled natural language. In Requirements Engineering Conference (RE), 2015 IEEE 23rd International (pp. 280-281). IEEE.
6. Nigam, A., Nigam, B., Bhaisare, C., & Arya, N. (2012A, March). Classifying the bugs using multi-class semi supervised support vector machine. In *Pattern Recognition, Informatics and Medical Engineering (PRIME), 2012 International Conference on* (pp. 393-397). IEEE.
7. Ribeiro, C., Farinha, C., Pereira, J., & da Silva, M. M. (2014). Gamifying requirement elicitation: Practical implications and outcomes in improving stakeholders collaboration. Entertainment Computing, 5(4), 335-345.
8. Shah, T., & Patel, V. S. (2014). A review of requirement engineering issues and challenges in various software development methods. International Journal of Computer Applications, 99(15), 36-45.
9. Bano, M. (2015, August). Addressing the challenges of requirements ambiguity: A review of empirical literature. In Empirical Requirements Engineering (EmpiRE), 2015 IEEE Fifth International Workshop on (pp. 21-24). IEEE.
10. Haron, H., & Ghani, A. A. A. (2015). A Survey on Ambiguity Awareness towards Malay System Requirement Specification (SRS) among Industrial IT Practitioners. Procedia Computer Science, 72, 261-268.
11. Cheng, B. H., & Atlee, J. M. (2007, May). Research directions in requirements engineering. In 2007 Future of Software Engineering (pp. 285-303). IEEE Computer Society.
12. Laplante, P. A. (2017). Requirements engineering for software and systems. Auerbach Publications.
13. Elsaid, A. H., Salem, R. K., & Abdul-kader, H. M. (2015, December). Automatic framework for requirement analysis phase. In Computer Engineering & Systems (ICCES), 2015 Tenth International Conference on (pp. 197-203). IEEE.
14. Pandey, D., Suman, U., & Ramani, A. K. (2010, October). An effective requirement engineering process model for software development and requirements management. In Advances in recent technologies in communication and computing (artcom), 2010 international conference on (pp. 287-291). IEEE.

15. Georgiades, M. G., & Andreou, A. S. (2010, November). Automatic generation of a software requirements specification (SRS) document. In Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on (pp. 1095-1100). IEEE.
16. Takoshima, A., & Aoyama, M. (2015, December). Assessing the Quality of Software Requirements Specifications for Automotive Software Systems. In Software Engineering Conference (APSEC), 2015 Asia-Pacific (pp. 393-400). IEEE.
17. Gill, K. D., Raza, A., Zaidi, A. M., & Kiani, M. M. (2014, May). Semi-automation for ambiguity resolution in Open Source Software requirements. In Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on (pp. 1-6). IEEE.
18. Gause, D. C., & Weinberg, G. M. (1989). Exploring requirements: quality before design (pp. 152-164). New York: Dorset House Pub..
19. Sandhu, G., & Sikka, S. (2015, May). State-of-art practices to detect inconsistencies and ambiguities from software requirements. In Computing, Communication & Automation (ICCCA), 2015 International Conference on (pp. 812-817). IEEE.
20. Gleich, B., Creighton, O., & Kof, L. (2010, June). 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.
21. Hagal, M. A., & Alshareef, S. F. (2013, December). A systematic approach to generate and clarify consistent requirements. In IT Convergence and Security (ICITCS), 2013 International Conference on (pp. 1-4). IEEE.
22. Gulia, S., & Choudhury, T. (2016, January). An efficient automated design to generate UML diagram from Natural Language Specifications. In Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference (pp. 641-648). IEEE.
23. Beg, R., Abbas, Q., & Joshi, A. (2008, July). A method to deal with the type of lexical ambiguity in a software requirement specification document. In Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on (pp. 1212-1215). IEEE.
24. Shah, U. S., & Jinwala, D. C. (2015). Resolving ambiguities in natural language software requirements: a comprehensive survey. ACM SIGSOFT Software Engineering Notes, 40(5), 1-7.
25. Arendse, B. (2016). A thorough comparison of NLP tools for requirements quality improvement (Master's thesis).
26. Génova, Gonzalo, José M. Fuentes, Juan Llorens, Omar Hurtado, and Valentín Moreno. "A framework to measure and improve the quality of textual requirements." *Requirements engineering* 18, no. 1 (2013): 25-41.
27. Popescu, D., Rugaber, S., Medvidovic, N., & Berry, D. M. (2007, September). Reducing ambiguities in requirements specifications via automatically created object-oriented models. In Monterey Workshop (pp. 103-124). Springer, Berlin, Heidelberg.
28. Femmer, H., Fernández, D. M., Juergens, E., Klose, M., Zimmer, I. & Zimmer, J. Rapid Requirements Checks With Requirements Smells: Two Case Studies. Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, 2014. ACM, 10-19.
29. Korner, S. J., & Brumm, T. (2009, September). Resi-a natural language specification improver. In Semantic Computing, 2009. ICSC'09. IEEE International Conference on (pp. 1-8). IEEE.

30. Umber, A., Bajwa, I. S., & Naeem, M. A. (2011, July). NL-based automated software requirements elicitation and specification. In International Conference on Advances in Computing and Communications (pp. 30-39). Springer, Berlin, Heidelberg.