Extraction of Essential Requirements from Natural Language Requirements

N. A. Moketar[1], M. Kamalrudin[1, 2]
[1]Innovative Software System and Services Group,
[2] Institute of Technology and Management and Entrepreneurship.
Universiti Teknikal Malaysia Melaka, HangTuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia
massila@utem.edu.my

***Abstract*— Requirement is a formal expression of user's need. It is the main foundation of any software development project. Natural language (NL) is often used to express and write system requirements specifications as well as user requirements. However, there is a very high probability that more than half natural language requirements can be ambiguous, incomplete and inaccurate. A software engineer can miss-interpret the natural language requirements and can generate an erroneous software model, which finally will lead to project failure. Earlier, we have introduced a prototype tool that provides natural language requirements authoring facilities and consistency checking to assist requirement engineers when working with informal and semi-formal requirements. However, the tool has pattern limitation to support the extraction of the essential requirements from the NL requirements. Therefore this study is aimed to enhance the accuracy and scalability of the tool to capture the essential requirements from the NL requirements. Our approach is to implement lexical analysis and embed an English lexical database where it will serve as a thesaurus in the tool. This tool is expected to be able to find the synonym of the extracted phrases (essential requirements) in the database to match it to the essential interaction pattern (phrases and expressions) in the library. Our future work will focus on the next phase of requirements engineering, which is requirements validation.**

***Index Terms*— About; Essential Use Case; Natural Language; Requirements Engineering; Synonym Extraction.**

I. INTRODUCTION

Requirements are the essential ingredient that represents the user's need and expectation of the intended software. It is often express and written in natural language [1][2]. Although natural language requirements are universal and flexible, it is error-prone due to both ambiguities and complexities of natural language [3]. It can be easily miss-understood and miss-interpreted by the requirements engineers and development team. This can produce an erroneous and poor quality software model, which eventually will lead to software project failure. Therefore, it is important to ensure its correctness as it forming the basis for the system life cycle.

In this paper, we introduced the new enhancement of our tool to enhance its accuracy and scalability in extracting essential requirements from natural language requirements. Our approach is to implement lexical analysis and embed an English lexical database where it will serve as a thesaurus in the tool. This tool is expected to be able to find the synonym of the extracted phrases (essential requirements) in the database to match it to the essential interaction pattern (phrases and expressions) in the library. The presentation of this paper is organized as follows: While section one presents the introduction, section two presents the background and motivation behind this study. Section three discusses our proposed approach and expected result. This is followed by section four that provides the related works on managing requirements ambiguity. Finally, the conclusions and future works are summarized in section five.

I. BACKGROUND AND MOTIVATION

We use the Essential Use Cases (EUCs) a semi-formalized model to translate and represent the natural language requirements in our studies [3]–[10]. An EUC is a simplified structured narrative to represent the user and system interaction without the need to describe a user interface in details. It takes the form of a dialogue between the user and the system, and are organized into an interaction sequence. The aim is to support better communication between the developers and the stakeholders via a technology-free model and to assist better requirements capture. Compared to a conventional UML use case, an equivalent EUC description is generally shorter and simpler as it only comprises the essential steps (core requirements) of intrinsic user interest.

Figure 1 shows an example of a textual natural language requirement (left-hand side) and an example EUC (right-hand side) capturing this requirement. On the left is the textual natural language requirement from which important phrases are extracted (highlighted). From each of these, a specific key phrase (essential requirement) called an abstract interaction is abstracted and is shown in the EUC on the right, and categorized as user intentions and system responsibilities. This assists to abstract the requirements for specific technologies. For example, the requirement of login information either user need to type in the login information or using biometrics as an identification tool are transformed to a more abstract expression of a requirement called —"identify self".

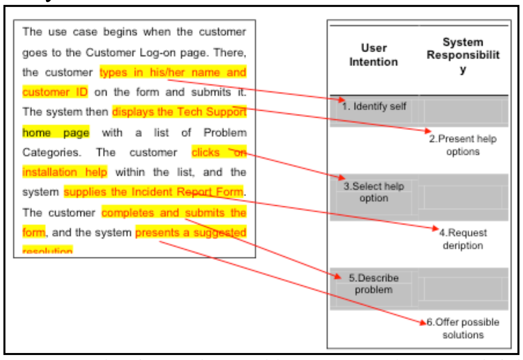
The use case begins when the customer goes to the Customer Log-on page. There, the customer types in his/her name and customer ID on the form and submits it. The system then displays the Tech Support home page with a list of Problem Categories. The customer clicks on installation help within the list, and the system supplies the Incident Report Form. The customer completes and submits the form, and the system presents a suggested resolution

Figure 1: Example of textual natural language requirements (left-hand side) and example of EUC (right-hand side)

Previously we have developed a lightweight prototype tool called MaramaAI to support the automatic extraction and modeling of EUC from a set of textual requirements [3][5][6]. It provides authoring facilities for textual requirements and checking the consistency of these requirements. Then, we introduced TestMEReq tool [8][9][11] as an enhancement of MaramaAI to assist requirements engineers to validate the requirements with client-stakeholder through the generated abstract tests and mock-up user interface. For these, we have developed a pattern library to store the collection of essential use cases. The pattern library consists of phrases describing abstract interactions to be identified, and they are extracted from the natural requirements. The extracted phrases are compared with the stored abstract interaction terminology in the database, which gained from various domain scenarios. Here is where the tool is lacking with an intelligent search-based method to identify potential essential interaction from the extracted NL requirements to match with the abstract interaction pattern in the database. This issue will create an incomplete EUC model, which eventually will influence the design and development of the system. For example, there are only four essential interactions (words/phrase) that match to the abstract interaction of "search item" in the current essential interaction library as shown in Table 1. If the users write the words such as "seek for a book" or "look for a CD" in their requirements scenario, the system will not be able to find and display the matching abstract interaction to these scenarios. The other limitation of the tool is that the users need to have a basic knowledge of EUC in order to write a good narrative (scenario) of the requirements. The narrative in the EUC is to be expressed in the language of the application domain and users.

Motivated from these limitations, this new study is aimed to overcome the issues in order to enhance the accuracy and scalability of the tool to capture the essential requirements from the NL requirements. Our approach is to implement lexical analysis and embedding an English lexical database (WordNet) where it will serves as a thesaurus in the tool. This tool is expected to be able to find the synonym of the extracted phrases (essential interaction/requirements) in the textual requirements to match it to the essential interaction pattern (phrases and expressions) in the library.

Table 1
Example of Abstract Interaction and Essential Interaction Patterns

Abstract Interaction	Essential Interaction (Words/Phrase)
Search item	Search of CD
	Search book
	Search for item
	Search for book

II. Related Works

The main goals for any tool for identifying and measuring ambiguities in natural language requirement specification are: (1) to identify which sentences in a natural language requirements specification are ambiguous and, (2) for each ambiguous sentence, to help the user to understand why it is ambiguous, so that he can remove the ambiguity from the sentence, and thus improve the natural language requirement specification.

Kamalrudin et al [3] have developed a light-weight prototype tool to support the extraction of Essential Use Case (EUC) models from natural language requirements and support for traceability and consistency management. The tool allows users to capture their requirements and generate Essential Use Case automatically. A collection of essential use case interactions is stored in a database. The database consists of phrases describing abstract interactions to be identified and they are extracted from the natural language requirements. The extracted phrases are compared with the stored abstract interaction terminology in the database, which gained from various scenario domains. Here is where the tool is lacking in term of ambiguity checking where it is unable to identify ambiguous words from the extracted natural language requirements.

A research by [12] has presented a reliable tool for ambiguity detection compare to an average human analyst and also able to explain ambiguity sources. This tool reads the input text line by line and checks for matching regular expressions for ambiguity detection. Although the tool was able to perform lexical and syntactic analysis only, it is able to detect ambiguities on all levels from lexical and pragmatic.

Another study done by [13] has introduced a prototype tool named SREE (Systemized Requirements Engineering Environment) to detect the occurrence of instances of ambiguity in requirements specifications. The authors also have embedded a lexical analyzer in their tool to find ambiguities in natural language requirement specifications. However, the tool still has weakness especially in finding ambiguity matching indicators in the Plural corpus.

Allan et al [14] in their study has presented a software prototype that combined natural language processing (NLP) techniques and specialized dictionaries to examine software requirements written in English to identify if it satisfies the three properties (accuracy, non-ambiguity, and verifiability). This tool incorporated the WordNet and VerbNet dictionaries to help in analyzing and validating the words and verbs in the requirements specification. This research shows that the combination of specialized dictionaries such as WordNet and VerbNet, stand-alone tools such as parsers and a general purpose scripting language (Perl) to create a prototype tool that can to help in analyzing natural language requirements specifications.

Rong Li et al [15] presents an approach to the representation of requirements based on the requirement ontology. They proposed ontology to represent both sentence and word level semantics. They applied the Generalized Upper Model (GUM) to identify the requirement ontology and WordNet to explain keywords in order to capture the semantics of natural language requirements for further processing.

Lami et al [16] have proposed a methodology and a tool named QuARS for systematically and automatically analyze natural language requirements. This tool has proven effective in analyzing natural language requirement but limited to address linguistic defects and syntax-related issues. In addition, the effectiveness of this tool also strongly depends on the completeness and accuracy of the dictionaries it uses i.e. V-dictionary.

Our study is more focusing on preventing a lexical ambiguity in the requirements specification. Lexical ambiguity occurs when one word has several meanings, or two words of different origin come to the same spelling and pronunciations. It is found from the literature review that embedding a lexical database may help in finding potential ambiguity in natural language requirements specification.

The most English word has plural meaning that may give different meaning in the different context of usage. This lexical database can serve a thesaurus that will find the synonym of words.

III. Our Proposed Approach

To overcome the issue, we proposed to embed an English lexicon database, i.e. WordNet in the tool which will play it role as a thesaurus and lexical analysis will be performed. Figure 2 illustrates the tool framework proposed by [1], while Figure 3 highlights our contribution in this study. Our lexical analysis approach consists of three main steps:

a. Essential interaction extraction and POS tagging
b. Clustering and classification
c. Finding synonym of extracted terms/words

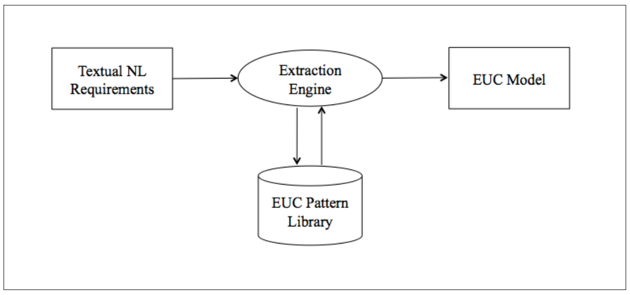

Figure 2. Previously developed interaction extraction approach.

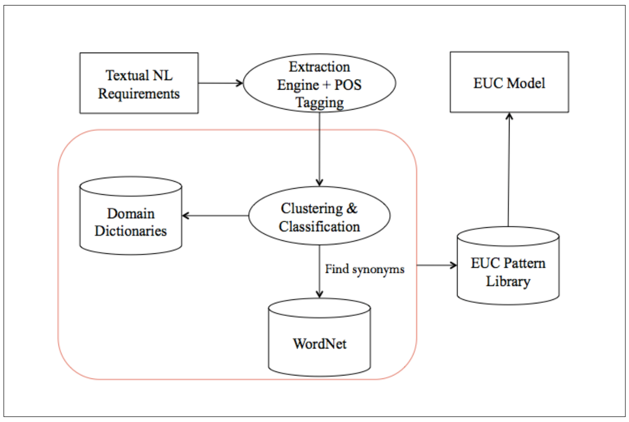

Figure 3. Our proposed approach

A. Essential Interaction Extraction and POS Tagging

Currently, there are three defined sentences structured as described below:

a. Verb (V) + Noun (N) (only) e.g. request (V) amount (N)
b. Verb (V) + Articles (ART)+ Noun (N) e.g. issue (V) a (ART) receipt (N)
c. Verb (V) + Adjective (ADJ)+ Noun (N) e.g. ask (V) which (ADJ) operation (N)

The extraction engine extracted the selected phrases ("key textual structures") from the natural language text based on their sentence structure. Any phrases that follow these defined structures will be accepted as essential interaction pattern in the interaction library.

To realize our proposed approach, we will improve this extraction algorithm for preparing the input for lexical analysis. Here, we will mark the textual natural language requirements with Part-of-Speech (POS) tagging. POS is a very important component for detecting ambiguity. Instead of only extracting selected phrases, our proposed approach will extract every word and marks them as an adjective, verb, nouns, pronouns, etc.

B. Clustering and Classification

External domain dictionaries will be developed to support this process. Here, we will define a glossary and classified all the domain- and system-specific terms used in the requirements. They contain sets of terms that are necessary to perform syntactical and lexical analysis and may vary according to the application domain and user need. The extracted words from the previous step will be matched and verified against these domain dictionaries.

C. Clustering and Classification

For this process, we will implement an algorithm that will automatically associate the appropriate meaning of words/phrases with our embedded thesaurus, i.e. WordNet. WordNet served as a passive component in this tool.

This new enhancement is expected to be able to identify and avoid the potential lexical ambiguity in the textual natural language requirements provided by the user. Further, it will also enhance the efficacy and scalability of the tool in order to extract essential interaction and abstract interaction from natural language requirements.

IV. Conclusion

This study is to improve the automated tracing tool to allow both technical and non-technical users to write the system requirements in natural language as possible. This study also aimed to overcome the issues of vague requirements statement, which can be interpreted differently. Some English word can have two or more possible meanings or synonym, which may lead to ambiguity. Our proposed approach is to embed our tool with an English Lexicon Database, i.e. WordNet in order to overcome the issue of vague requirements. This new enhancement is expected to be able to extract the abstract interaction automatically and then generate the EUC model from textual natural language requirements.

This study is still open for further future work. Currently, we are looking at the next level of requirement engineering process, i.e. requirements validation. We would like to enhance this tool. Therefore, it also will help in validating the extracted EUC model with the original requirements from the user. Another area of improvement is to look into the semantic in the requirements to allow non-technical users to write the requirements in natural language as possible. The current tool only accepts the requirements written in specific sentence structures. Finally, we will also look deeper into the linguistic ambiguity issues such as in lexical, syntactic, semantic, pragmatic, vagueness and generality.

Acknowledgment

This research is funded by Ministry of Higher Education Malaysia (MOHE), Universiti Teknologi Mara (UiTM), Fulgent Corporation, FRGS grant: FRGS/2/2013/ICT01/FTMK/02/2/F00185.

References

[1] Alberto Rodrigues da Silva, "Quality of Requirements Specifications: A Preliminary Overview of An Automatic Validation Approach," in 29th Annual ACM Symposium on Applied Computing, 2014, pp. 1021–1022.

[2] M. Kamalrudin, J. Grundy, and J. Hosking, "MaramaAI: tool support for capturing and managing consistency of multi-lingual requirements," Proc. 27th IEEE/ACM Int. Conf. Autom. Softw. Eng. - ASE 2012, p. 326, 2012.

[3] M. Kamalrudin, J. Grundy, and J. Hosking, "Tool support for essential use cases to better capture software requirements," in Proceedings of the IEEE/ACM international conference on Automated software engineering - ASE '10, 2010, p. 255.

[4] M. Kamalrudin and J. Grundy, "Generating essential user interface prototypes to validate requirements," in 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), 2011, pp. 564–567.

[5] M. Kamalrudin, J. Grundy, and J. Hosking, "Managing Consistency between Textual Requirements, Abstract Interactions and Essential Use Cases," in 2010 IEEE International Conference on Computer Software and Applications Conference (COMPSAC), 2010, no. July, pp. 327–336.

[6] M. Kamalrudin, J. Hosking, and J. Grundy, "Improving requirements quality using essential use case interaction patterns," in 2011 33rd International Conference on Software Engineering (ICSE), 2011, pp. 531–540.

[7] M. Kamalrudin, N. A. Moketar, J. Grundy, and J. Hosking, "Automatic Acceptance Test Case Generation From Essential Use Cases," in 13th International Conference on Intelligent Software Methodologies, Tools and Techniques, 2014, pp. 246–255.

[8] N. A. Moketar, M. Kamalrudin, S. Sidek, and M. Robinson, "TestMEReq : Automated Acceptance Testing Tool For Requirements Validation," in International Symposium on Research in Innovation and Sustainability, 2014, pp. 15–16.

[9] N. A. Moketar, M. Kamalrudin, S. Sidek, M. Robinson, and J. Grundy, "TestMEReq: Generating Abstract Tests for Requirements Validation," in Proceedings of the 3rd International Workshop on Software Engineering Research and Industrial Practice - SER&IP '16, 2016, pp. 39–45.

[10] N. A. Moketar, M. Kamalrudin, S. Sidek, M. Robinson, and J. Grundy, "An Automated Collaborative Requirements Engineering Tool for Better Validation of Requirements," in 31st IEEE/ACM International Conference on Automated Software Engineering (ASE 2016), 2016, pp. 864–869.

[11] N. A. Moketar, M. Kamalrudin, S. Sidek, M. Robinson, and J. Grundy, "An automated collaborative requirements engineering tool for better validation of requirements," in Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering - ASE 2016, 2016, pp. 864–869.

[12] B. Gleich, O. Creighton, and L. Kof, "Ambiguity Detection : Towards a Tool Explaining Ambiguity Sources," in 16th International Working Conference, REFSQ, 2010, vol. 6182, pp. 218–232.

[13] S. F. Tjong and D. M. Berry, "The design of SREE - A prototype potential ambiguity finder for requirements specifications and lessons learned," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7830 LNCS, pp. 80–95, 2013.

[14] A. B. Rojas, G. B. Sliesarieva, U. D. C. Rica, S. Jos, and C. Rica, "Automated Detection of Language Issues Affecting Accuracy , Ambiguity and Verifiability in Software Requirements Written in Natural Language," in Proceeding of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas, 2010, no. June, pp. 100–108.

[15] R. Li, K. He, and H. Chen, "From natural language requirements to requirement ontologies," 2010 2nd Int. Conf. Futur. Comput. Commun., pp. V3-755-V3-758, 2010.

[16] G. Lami, S. Gnesi, F. Fabbrini, M. Fusani, and G. Trentanni, "An Automatic Tool for the Analysis of Natural Language Requirements.".