International Journal of Engineering & Technology, 7 (4.38) (2018) 1416-1419

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Knowledge-Based Requirements Engineering in Detecting Ambiguity

Jacline Sudah Sinpang, Shahida Sulaiman, Norsham Idris

Department of Software Engineering, Faculty of Computing,
Universiti Teknologi Malaysia, 81310, Johor Bahru, Johor, Malaysia

Abstract

Requirements engineering is a phase where the elicitation and analysis of requirements are crucial. Low quality requirements could cost a software project to experience failure, as it does not meet users' needs. One of the causes of poor quality requirements is ambiguity. Software engineers or more specifically system analysts normally gather requirements in natural language where it is easy for both developer and stakeholders to understand each other. However, natural language in requirements gathering is prone to ambiguity. Ambiguous requirements will cause developers to understand differently from what the stakeholders actually require. Thus, it is important to avoid ambiguous requirements. As there are many types of ambiguity, this study focuses on the vagueness of functional requirements. The study implements rule-based reasoning in knowledge-based requirements engineering to detect vagueness. Rules for the proposed prototype tool are obtained from journals and experts in requirements engineering. The evaluation shows that diverse categories of vagueness can be detected earlier using the proposed technique and its tool.

Keywords: *Requirements engineering, Knowledge-based requirements engineering, Requirements ambiguity, Requirements vagueness.*

1. Introduction

Requirements engineering is the process of finding, recording and maintaining requirements for software solutions [1]. It is the most important phase in Software Development Life Cycle (SDLC) as it determines the success and failure of software development [2]. This phase includes four main activities that are requirements elicitation, analysis, specification, and verification [3].
In most cases, gathering of requirements are in a natural language such as English and then the requirements are analyzed to form a formal software design or model [4]. However, despite its advantage of being universal, flexible and simple; natural language are intrinsically ambiguous [5], [6]. The understanding of anything described in natural language can be affected potentially by geographical, psychological and sociological factors [7]. Thus, if stakeholders interpret requirements in a different way, this can result in an incorrect implementation [8]. Ambiguity is characteristically of poor quality requirements, and poor quality requirements are characteristic of challenged projects [9]. Therefore, it is important to tackle the issue of ambiguity in the early phase as soon as possible.
In order to tackle the issue of ambiguity in requirements engineering, this research focuses on the rule-based technique in knowledge-based system. A knowledge-based system comprises computer programs that use artificial intelligence to solve problems within a specialized domain that ordinarily requires human expertise. Artificial intelligence as defined by IEEE ISO, 8402:1995 [10] are: (i) Investigation of a planned computer system by showing the attributes related with the insight in human conduct: understanding language, learning, thinking from fragmented or dubious data, and taking care of issues. (ii) The order for creating a computer system that is able to do breezing through the Turing test, in which the conduct of the computer system is no different from human conduct. (iii) Investigation of critical thinking that is achieved by utilizing computational models.
The tools of artificial intelligence can be divided into two broad types: knowledge-based system (KBS) and computational intelligence (CI). KBS includes techniques such as rule-based, model-based and case-based reasoning. The main distinction between a knowledge-based system and a conventional program lies in the structure where, in a KBS, there are two separate modules which are (i) the knowledge module, called knowledge base and (ii) control module, called the inference engine [11].
Modern KBS have their roots in early computer systems known as Expert Systems [12]. The purpose of the knowledge-based is to store information regarding the application with which the expert system is concerned; this information is typically stored in the form of facts that are known about the application and its environment, and relationships between these facts [13]. The inference engine is designed to use the information stored in the knowledge-based to reason and deduce new knowledge that can be used to make decisions.
The following Section II outlines the related work for this study, Section III is the proposed work, Section IV is the experiment, and Section V is the result and discussion. The final section concludes the paper and its future work.

2. Related Work

Knowledge is the information about a particular area needed by a computer program to enable it to exhibit intelligent behavior with regard to a specific problem. Knowledge includes information about both real-world entities and the relationships between them [14]. Cauvin et al. [15] created a tool called Facilitator in bridging the gap between a software engineer and knowledge engineer. The tool is able to determine the most suitable domain and locate an

ontology automatically. By connecting software design and domain knowledge, it shows the potential for the increase of productivity of programmers by detecting mistakes at the early stage. Abke et al. [16] use project-based learning approach in software engineering by implementing collaborative knowledge transfer. Knowledge is obtained from a real-world situation and the knowledge is multiplied via wiki. Batanov [17] proposed a new definition of software engineering by exploring different types of knowledge to support knowledge-based software engineering.
Some researches adopt knowledge-based to improve requirements activity. Table 1 shows existing works in requirements engineering that apply knowledge-based.

Table 1. Knowledge-based in requirements engineering

Proposed work	Strength	Weakness
Knowledge distribution in requirements engineering by Kirikova and Grundspenkis [18]	Use of knowledge distribution contributes to completeness and transparency of requirements, and provides an opportunity to bridge the natural and artificial knowledge being used in requirements engineering	It is hard to maintain the knowledge distribution
Framework for knowledge-based by Nguyen et al. [1]	It offers stakeholders an automated means to elicit, structure, and manage requirements	Not able to detect the conflicts associated with the requirements that are not expressible in description logic
Ontology-based requirements engineering by Yang et al. [19]	Able to check ontology consistently	Not able to define the causes of inconsistency in ontology
Modelling framework by Prasetyo and Bandung [20]	Modeling framework can be utilized to design a knowledge-management-based requirement engineering framework	The modeling framework is not detailed enough for requirements engineering
Automation system by Runde et al. [21]	defines the possible requirements with all their attributes and possible values as well as their causal dependencies on other requirements	Query language needed to be described

Generally, in KBS, the interpretations may be classified into two categories: vagueness and ambiguity. Vagueness is associated with difficulty in making sharp or precise distinctions in the world that is a domain of interest is vague if it cannot be delimited by sharp boundaries. Ambiguity relates to one-to-many relationships, that is, situations in which the choice between two or more alternatives is left unspecified [22].
The works in Table 1 imply that researchers aim to improve certain activity in requirements engineering and by using knowledge-based techniques, improvements are made. However, researchers are yet to use knowledge-based in solving ambiguity in requirements. Thus, this gap motivates our research to explore knowledge-based in improving the quality of requirements by eliminating ambiguity.
The focus of this study is to use knowledge-based technique in detecting ambiguity especially vagueness of requirements that are elicited using natural language. The analysis of the requirements is conducted while the requirements are yet to be refined into a formal requirements language or models.

3. Rule-Based Technique in Detecting Ambiguity

KBS are normal database management systems extended with some kind of knowledge [18]. Knowledge usually means some kind of declarative language and takes the form of rules [23].

3.1 Knowledge-based System

Generally, KBS architecture is as shown in Fig. 1, which comprises user interface, inference engine, and knowledge base. Knowledge base and inference engine are separated because the knowledge base must be able to grow and change as knowledge is added.

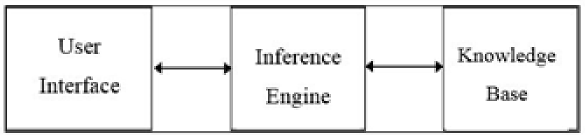


Fig. 1. Simple KBS architecture

A more elaborated description of KBS architecture is shown in Fig. 2 where the components are more likely to be found in any real-world system. The system holds a collection of general principles, which can potentially be applied to any problem; these are stored in the knowledge base. The system also holds a collection of specific details that apply to the current problem including details of how the current reasoning process is progressing; these are held in working memory. The inference engine processes this information.

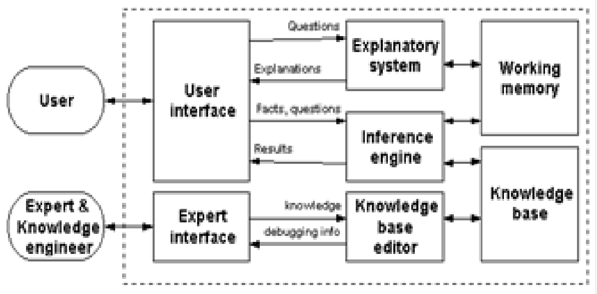


Fig. 2. Elaborated KBS architecture

3.2 Rule-based Reasoning

Rules can often represent the expertise that someone uses to do an expert task. A rule means a structure, which has “if” component, and a “then” component. The statement or sets of statements after the word “if” represent some observed patterns. The statement or set of statements after the word “then” represent some conclusions or some actions that are drawn from the observed patterns that should be taken into consideration. Thus, rule-based technique either (i) identifies a pattern and draws conclusions about what it means, (ii) identifies a pattern and advises what should be done about it and (iii) identifies a pattern and take appropriate action.
The essence of a rule-based reasoning system is that it goes through a series of cycles. In each cycle, it attempts to pick an appropriate rule from its collection of rules, depending on the present circumstances. As using a rule produces new information, it is possible for each new cycle to take the reasoning process further than the cycle before. This is rather like a human following a chain of ideas in order to conclude. This study will incorporate rule-based reasoning in a knowledge base that comprises rules and facts where forward chaining is applied that is the “if” clause is known to be true. Fig. 3 shows the architecture of the proposed tool.

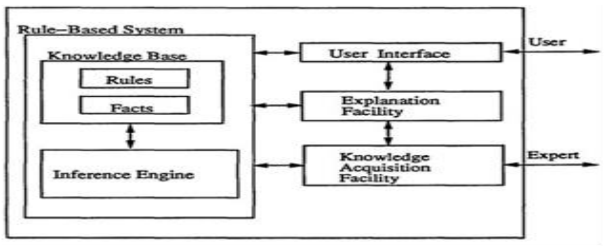


Fig. 3. Rule-based reasoning architecture

3.3 Rule Syntax in the Prototype Tool

Two main rules in the tool that can help in detecting ambiguity include: (i) Check – go through every existing rule to detect vague words and (ii) Insert – insert new rules to improve the current knowledge. Users for this tool shall be system analyst who analyzes the requirements before revising them into formal requirements language. Expert for this tool is the one who provided the knowledge or rules for the system. When system analysts find out that there are words that are not included in the rules but suspected to be ambiguous or vague, they can refer to the expert. The expert has the privilege to add-on new rules based on the new obtained knowledge.

The initial rules for the proposed tool are mainly obtained from journals. Firesmith [24] provides a detail classification of vague words. These vague words are branch out into five categories: (i) Vague subjects, (ii) Vague adjectives, (iii) Vague prepositions, (iv) Vague verbs and (v) Subjective phrases. These five categories are further broken down into more details attributes. If a new category is discovered, the rules should be added accordingly.

The proposed tool named as RbREAD (Rule-based Requirements Engineering for Ambiguity Detection) is written in JAVA language. It contains three main parts that include: (i) main, (ii) rules and (iii) menu. The main part of this tool is the rules part. Whenever requirements engineer enters a requirement, the tool will check, split the strings in the requirement sentence and check the elements of the string or words thoroughly. Thus, it is important that each requirement should be entered as a sentence and a single requirement not combination of requirements. The tool will display the results of the ambiguity detection of each requirement and prompt requirements engineer the detected vague words that exist in the parsed requirements. A more detail explanation of the steps in detecting ambiguity is shown in Algorithm 1.

Algorithm 1

```
Input: Requirements, $R = \{r_1, \ldots, r_n\}$
Vague Words, $V = \{v_1, \ldots, v_n\}$
Word Type, T = {subject, adjective, preposition, verb, subjective phrase}
Output: Vagueness category, $C = \{c_1, \ldots, c_5\}$

Begin
1.  Initialize Words, $W = \{w_1, \ldots, w_n\}$
2.  For each member of requirement in set R
    2.1  Split strings into set of words, W;
    2.2  For i=1 to n and each member $w_i$ of W
         2.2.1  If $w_i$ is a subject && $w_i$ is a member of set V then
                vagueness category is $c_1$;
         2.2.2  Else if $w_i$ is an adjective && $w_i$ is a member of
                set V then vagueness category is $c_2$;
         2.2.3  Else if $w_i$ is a preposition && $w_i$ is a member of
                set V then vagueness category is $c_3$;
         2.2.4  Else if $w_i$ is a verb && $w_i$ is a member of set V
                then vagueness category is $c_4$;
         2.2.5  Else if $w_i$ is a subjective phrase && $w_i$ is a member
                of set V then vagueness category is $c_5$;
    2.3  Repeat step 2.2 until every word in set W is checked;
3.  Display vague word found and its category;
4.  Repeat step 1 until every requirement in set R is checked;
End
```

4. Result and Discussion

RALIC (Replacement Access, Library and ID Card) dataset is used to evaluate the rule-based technique in RbREAD tool. It is an open source data for a case study project of University College London (UCL) [25]. Requirements gathered for RALIC system are with the aim to substitute the outdated access control systems, combining various existing access control mechanisms, and at the minimal, combine the photo ID card, access card, and library card. Fig 4 shows the example of requirements elicited for the RALIC system.

Table. 2. Elicited requirements for RALIC

No.	Requirements	Category
1	Security Systems are aware how they store info and who can access	Vague subjects
2	The card can be extended for future requirements, such as a digital certificate	Vague adjectives
3	Appointment system for students as above	Vague prepositions
4	Save time in processing memberships and renewals, increase accuracy of transfer of information as some of the members details are on the card	Vague verbs
5	All this should be quick and easy to do in the RALIC software	Vague subjects and vague adjectives

The proposed RbREAD tool interface allows users to choose either to enter requirements to be analyzed or to enter new rules to be added onto the current knowledge base. Fig. 4 shows the interface of the tool.

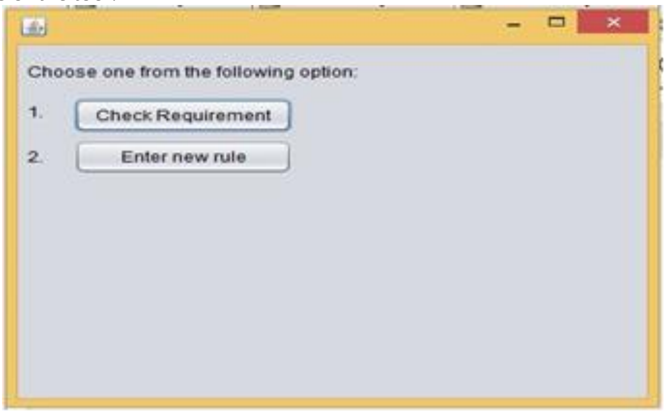
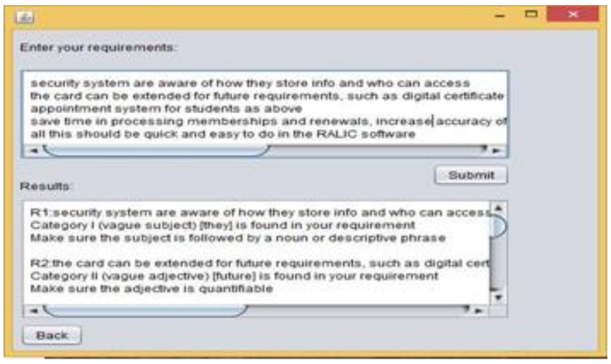


Fig 4. Main interface of the tool

If user chooses option 1, the tool will prompt them to enter the number of requirements that they wish to analyze. Then it will analyze every single word from each requirement and provide the type of vagueness that the word contains. Fig 5 shows the example of how the tool analyzes the requirements and displays the results.



```
Enter your requirements:

security system are aware of how they store info and who can access
the card can be extended for future requirements, such as digital certificate
appointment system for students as above
save time in processing memberships and renewals, increase accuracy of
all this should be quick and easy to do in the RALIC software

Results:

R1:security system are aware of how they store info and who can access
Category I (vague subject) [they] is found in your requirement
Make sure the subject is followed by a noun or descriptive phrase

R2:the card can be extended for future requirements, such as digital cert
Category II (vague adjective) [future] is found in your requirement
Make sure the adjective is quantifiable
```

Fig. 5. Examples of the analyzed requirements

The tool will analyze the requirements word by word until there are no more vague words detected in the requirements. However, the tool will not be able to detect the word if it is written differently from how it is entered by the expert. For example, if the user enters “user-friendly” instead of “user friendly”, the tool will assume that it is not a vague word. Thus, it is important for the experts to ensure the words are spelled correctly in either American or British English. This must be informed to system analysts or requirements engineers to ensure they enter the requirements with correct spelling to ensure high detection of words with vagueness.

5. Conclusion and Future Work

Vagueness in requirements causes ambiguity. If the requirements contain vague words, it is probable that the requirements will turn into vague requirements. Thus, it is important to analyze every single word in the requirements to make sure that the requirements are free from vague words before they are specified formally in requirements engineering phase.
Further work includes finding more categories of vagueness and their respective words to ensure the richness of the knowledge-based in the tool along with the improvement of the tool.

Acknowledgments

The authors express gratitude to the Research University Grant (RUG) of Universiti Teknologi Malaysia, Cost Centre 14H09 that supports this work.

References

[1] Nguyen, T. H., Vo, B. Q., Lumpe, M., Grundy, J,. “REInDetector: a framework for knowledge-based requirements engineering,” In Proceedings of the 27th IEEE/ACM international conference on automated software engineering (pp. 386-389). ACM, September 2012.
[2] Aoyama, M., & Saito, S., “Requirements Engineering Based on REBOK (Requirements Engineering Body Of Knowledge) and Its Practical Guide,” In Software Engineering Conference (APSEC), 2012 19th Asia-Pacific (Vol. 2, pp. 146-147), IEEE, December 2012.
[3] Sommerville, I., “Software Engineering. (9th ed.),” Addison-Wesley: Pearson, 2011.
[4] Umber, A., Bajwa, I. S., “Minimizing ambiguity in natural language software requirements specification,” In Digital Information Management (ICDIM), 2011 Sixth International Conference on (pp. 102-107). IEEE, September 2011.
[5] Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F., “Automated checking of conformance to requirements templates using natural language processing,” IEEE transactions on Software Engineering, 41(10), 944-968, 2015.
[6] Fatwanto, A.. “Software requirements specification analysis using natural language processing technique,” In QiR (Quality in Research), 2013 International Conference on (pp. 105-110). IEEE, June 2013.
[7] Liu, L., Li, T., Kou, X., “Eliciting relations from natural language requirements documents based on linguistic and statistical analysis,” In Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual (pp. 191-200). IEEE, July 2014.
[8] Chantree, F., Nuseibeh, B., De Roeck, A., Willis, A., “Identifying nocuous ambiguities in natural language requirements,” In Requirements Engineering, 14th IEEE International Conference (pp. 59-68). IEEE, 2006.
[9] Boyd, S., Zowghi, D., Farroukh, A., “Measuring the expressiveness of a constrained natural language: An empirical study,” In Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on (pp. 339-349). IEEE, 2005.
[10] IEEE ISO, 8402:1995. Quality Management and Quality Assurance.
[11] Hopgood, A. A., “Intelligent systems for engineers and scientists,” Boca Raton, FL: CRC Press, 2012.
[12] Gleich, B., Creighton, O., Kof, L., “Ambiguity detection: Towards a tool explaining ambiguity sources,” Requirements Engineering: Foundation for Software Quality, 218-232, 2010.
[13] Odisho, R., Xu, W., Bronlund, J., Peyron, M. A., “Knowledge-based system for autonomous control of intelligent mastication robots,” In Industrial Electronics (ISIE), 2017 IEEE 26th International Symposium on (pp. 1105-1110). IEEE, 2017.
[14] Bassiliades, N., Vuvhavas, I., “Active knowledge based system,” in Knowledge based system, vol. 1, no. 1, pp. 1-36,
[15] Cauvin, S. R., Sleeman, D., Vasconcelos, W. W., “Towards knowledge-intensive software engineering,” In Software Technologies (ICSOFT), 2015 10th International Joint Conference on (Vol. 1, pp. 1-8). IEEE, 2015.
[16] Abke, J., Gold, C., Kälberer, N., Kuhn, M., “Collaborative knowledge transfer via Wiki: A project based learning approach in software engineering,” In Interactive Collaborative Learning (ICL), 2014 International Conference on (pp. 283-288). IEEE, 2014.
[17] Batanov, D., “Knowledge-Based Support for Software Engineering,” In IFIP International Conference on Artificial Intelligence Applications and Innovations (pp. 219-229). Springer, Berlin, Heidelberg, 2010.
[18] Kirikova, M., Grundspenkis, J., “Using Knowledge Distribution in Requirements Engineering-6,” 2000.
[19] Ying-ying, Y., Zong-yong, L., Zhi-xue, W, “Domain knowledge consistency checking for ontology-based requirement engineering,” In Computer Science and Software Engineering, 2008 International Conference on (Vol. 2, pp. 302-305). IEEE, 2008.
[20] Prasetyo, N. A., Bandung, Y., “A design of software requirement engineering framework based on knowledge management and Service-Oriented Architecture Decision (SOAD) modeling framework,” In Information Technology Systems and Innovation (ICITSI), 2015 International Conference on (pp. 1-6). IEEE, 2015.
[21] Runde, S., Fay, A., & Wutzke, W. O., “Knowledge-based Requirement-Engineering of building automation systems by means of Semantic Web technologies,” In Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on (pp. 267-272). IEEE, 2009.
[22] Yang, J. D., Lee-Kwang, H., “Treating Uncertain Knowledge-Based Databases-11,” 2000.
[23] Jeffrey, D. U., “Principles of database and knowledge-base systems”, 1989.
[24] Firesmith, D., “Specifying good requirements,” Journal of Object Technology, 2(4), 77-87, 2003.
[25] Lim, S. L., “Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation,” Doctoral dissertation, University of New South Wales, 2010.