Detecting Ambiguity in Requirements Analysis Using Mamdani Fuzzy Inference

Jacline Sudah Sinpang, Shahida Sulaiman and Norsham Idris
Department of Software Engineering, Faculty of Computing,
Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia.
flynnyjac@gmail.com

***Abstract*—Natural language is the most common way to specify requirements during elicitation of requirements as stakeholders can better specify the services they want from a particular system. However, it is arguable that requirements gathered in natural language is free from error especially ambiguity. Ambiguity in requirements can cause requirement engineers or system analysts to perceive the requirements according to their understanding instead of stakeholders understanding. This study attempts to detect ambiguity mainly vagueness as early as possible using Mamdani fuzzy inference when analyzing requirements. Dataset used in this study comprises raw requirements that are still in natural language form. In order to create fuzzy rules, the analysis of the requirements in natural language involves the process of capturing the text patterns of the requirements. The results show that it is possible to use Mamdani fuzzy inference that can detect ambiguity in requirements analysis phase.**

***Index Terms*—Mamdani Fuzzy Inference; Natural Language; Requirements Analysis; Requirements Engineering.**

I. INTRODUCTION

Requirements engineering is a vital activity that can influence the whole phases in software development project [1]. Requirements provide the foundation to refine and elaborate the whole software development life cycle. The development of any software must be based on a high-quality requirement engineering process [2]. Thus, ambiguous requirements can contribute to low-quality requirements that can lead to the failure of a project.

Requirement according to the international standard of IEEE std., 29148-2011 [3], refers to a form of a report that represents a necessity and its related constraints and conditions. Requirements engineering, on the other hand, involves all lifecycle activities devoted to the identification of user requirements, analysis of the requirements to also derive additional requirements, documentation of the requirements as a specification, and validation of the documented requirements against user needs, as well as processes that support these activities [4].

Natural language (NL) is normally used to represent users' requirements that are to be met by the system or services [5]. The pervasive medium for this communication that is natural language is widely accepted to be tricky for high-accuracy communication because of its characteristic that leads to ambiguity and familiarity [6]. In addition, ambiguity in requirements can cause various issues that influence the system to be built, in light of the fact that ambiguity becomes a bug if not found and settled at early stages [7]. Ambiguous requirements may bring about misinterpretations among stakeholders, and prompt a few issues [8]. Thus, it is important that requirements engineers or system analysts handle the issues such as ambiguity at the early stage in requirement engineering process itself.

This research aims to solve the issue of ambiguity at the early stage by adopting the artificial intelligence (AI) technique that can help to detect the issue. Meziane and Vadera [9] state that some researchers adopt AI techniques to improve the software development activities and there is huge potential in utilizing AI for supporting and upgrading software engineering. Some existing works apply the techniques in AI for certain phases in requirements engineering and they have proven the significance in enhancing the software development activities.

AI as defined in the IEEE ISO, 8402:1995 [10] are summarized as below:

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 difference from human conduct.
iii. Investigation of critical thinking that is achieved by utilizing computational models

Fuzzy inference system is a part of AI where the involved process maps the given input variables to an output space via fuzzy logic based deducing mechanism. The system comprises If-Then rules, membership functions and fuzzy logical operations [11]. The If-Then rules in fuzzy logic approximate to people linguistic variable; this inference process is projecting crisp quantities onto human language and promptly yielding a precise value [11].

The three types of fuzzy inference system include (i) Mamdani fuzzy inference, (ii) Sugeno fuzzy inference and (iii) Tsukamoto fuzzy inference. Given the nature of this study is to detect ambiguity in natural language requirements, it focuses on Mamdani fuzzy inference where the whole process is weighting on the If-Then set and then the output of each rule will be reshaped by a matching number and the defuzzification will help to aggregate this output to the original fuzzy set.

The objectives of this study are:

i. To study and identify the characteristics of ambiguity (vagueness) to fit the fuzzy rules
ii. To enhance requirements analysis by implementing Mamdani fuzzy inference technique
iii. To evaluate the proposed technique in requirements analysis using an established requirements dataset

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.

II. Related Work

Arora et al. [12] suggest that NL is a standout amongst commonplace practices in eliciting requirements from the stakeholders, as it is less demanding to derive it. It is the fact that English is etymologically ambiguous and semantically conflicting [13]. Due to its nature, requirements engineers have accepted the fact that NL is inherently ambiguous.

In English literature, writers may intentionally utilize ambiguity in the sentences to give readers a chance to expand their creativity. In any case, if there is any ambiguity in requirements, it could bring about undesirable mistakes. Thus, it is necessary for requirements engineers to have basic ability to comprehend requirements and recognizing the ambiguity in requirements [14].

Ambiguity in texts is normally interpreted as sentences that have more than a single meaning. However, Massey et al. [14] define that there are six ambiguity types in line with the definitions that are used in requirements engineering, law and linguistic. The six ambiguity types are shown as below. This study focuses on vagueness.

i. Lexical ambiguity: A word or expression with numerous legitimate meaning.
ii. Syntactic ambiguity: Arrangement of word with different legitimate syntactic understandings regardless of context.
iii. Semantic ambiguity: Consist of more than one interpretations in the sentence.
iv. Vagueness: Statement that concede marginal case or relative interpretation.
v. Incompleteness: Provides too little detail in conveying the meaning in a grammatically correct sentence.
vi. Referential ambiguity: Confuses reader with it references on the provided context in a grammatically correct sentence.

In order to identify or eliminate ambiguity in requirements, there are works that attempt to solve these issues. Researchers use Natural Language Processing (NLP) to identify and solve ambiguity in requirements [6, 12]. Some works aim to tackle the issue of ambiguity in requirements. For example, software requirements specification is introduced to capture the complete description of the system. However, this does not necessarily ensure that the requirements are not ambiguous. Table 1 shows some of the works that attempt to solve ambiguity in requirements engineering.

There is noteworthy potential in utilizing AI to enhance the phases in the software development life cycle. Similar with other disciplines, software development quality enhances the experience, developers' knowledge, past activities and aptitude [8]. AI is a technique where a machine can learn from its experience and improve accordingly. Thus, it promotes automation in related problems. By applying Fuzzy modelling for product qualities in requirements, Davril et al. [15] discover that the technique can support the design of a product configurator by focusing on product qualities and enabling users to manipulate and perceive product regarding qualities.

Table 1
Solving Ambiguity in Requirements Engineering

Proposed work	Strength	Weakness
Framework by Arora et al. [12]	Provides a robust and accurate basis for checking conformance to templates	The framework is limited by the ontology editor features in which if the requirements captured are not according to the existing features, it might give a not really accurate result
Automated ambiguity detection tool by Gleich et al. [13]	Able to detect ambiguity and its sources	The tool is created only to focus on lexical and syntactic ambiguity
Ambiguity taxonomy by Massey et al. [14]	Taxonomy helps to identify ambiguity in legal texts	Participants in the case study did not really agree to the number and type of ambiguities in legal texts
Automated approach to generate semantic of business vocabulary and rules by Umber and Bajwa [16]	Provides a higher accuracy as compared to other natural language based tools	This approach only focuses on semantic ambiguity
Agent Oriented Framework by Bhardwaj and Goyal (2014) [17]	Framework is stable in a long-standing identification of the need.	No quantitative approach was made.
Hybrid Approach by Kumar et al. (2013) [18]	Approach is beneficial in handling requirements gathering in agile development.	This approach only effective for agile development.
Classification Methodology by Parra et al. (2015) [2]	Classifier is able to evaluate the quality of requirements.	Classifier needs more training to improve efficiency.
UML Integration by Siddique et al. (2014) [19]	Use cases are best applied to big projects or new developed system.	It is not confirmed how effective criteria in use cases at detecting faults.

Beritelli et al. [20] propose a simple approach to a small vocabulary word recognition by using fuzzy pattern matching. The finding of the approach stated that the use of fuzzy logic in the matching phase makes it easier to separate the class represented by the various words, thus simplifying the task of the final decision block. In addition, Baresi et al. [21] use fuzzy goal to specify requirements and adaptation capabilities in self-adaptive systems. It helps to transform a goal into live entities, the distinction between crisp and fuzzy goals, with which one can associate different satisfaction levels and the definition of adaptation strategies as if they were goals. All these elements help embed self-adaptability in software systems from the very beginning (requirements elicitation), and produce reasoning on possible consequences.

Although Table 1 shows the work in NL, the context of this NL is a requirement that has been written in a professional manner. It means that the requirements have been documented for software requirements specification (SRS). However, this research focuses on raw requirements that have been elicited using natural language that have not been documented into SRS in other words the requirements are not written in a proper requirement specification styles. Thus, this research anticipates discovering ambiguity in requirements as early as possible even before the requirements are professionally documented will lead to a better quality of

requirements. The If-Then rule in fuzzy logic can help when the input of the experiments is linguistic as the rules in fuzzy logic approximate to people linguistic variable.

III. THE PROPOSED WORK

A fuzzy inference system consists of three major steps, as shown in Figure 1. The first step is the fuzzification step. This first step involves the change of the numerical values into a different set of membership degrees in fuzzy. The second step is where the inference engine will analyze the fuzzy input using fuzzy rule base. The third step performs the defuzzification if necessary. It produces a crisp value from the rule aggregation result.

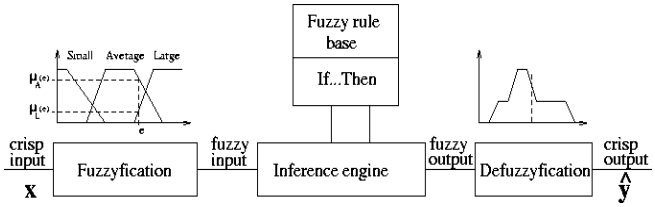

Figure 1: Fuzzy Inference System

A fuzzy rule is written as ***If*** *situation* ***Then*** *conclusion*. The situation, called rule premise or antecedent, is defined as a combination of relations such as *x is A* for each component of the input vector. The conclusion part is called consequence or conclusion.

Operators:
IS: the relation *x is A* quantified by the membership degree of *x* to the fuzzy set *A*
AND: conjunction operator, denoted ^ , the most common operators are minimum and product
OR: disjunction operator, the most common are maximum and sum

For this research, conjunctive rules which mainly use *AND* operator are applied to get the probability of requirements that contain certain combination words that might cause the requirements to be ambiguous requirements. Conjunctive rules represent positive knowledge; input (*A*) and output (*C*) represent pairs of combined possible values. Figure 2 shows an example of a general rule that is applied to the fuzzy logic.

If *a sentence contains word1* ***and*** *word2* ***then*** *the sentence is vague*

Figure 2: General rule

Certain combination of words that could contribute to vague requirements are specifically analyzed using conjunctive rules. The output from rule will give the probability on the vagueness of the requirements when certain combinations of words are detected in the sentence.

IV. THE EXPERIMENT

There are three stages of analysis in order to detect the ambiguity of requirements. The stages are: (i) manual analysis, (ii) natural language processing and (iii) fuzzy logic analysis. Figure 3 shows the flow of the analysis.

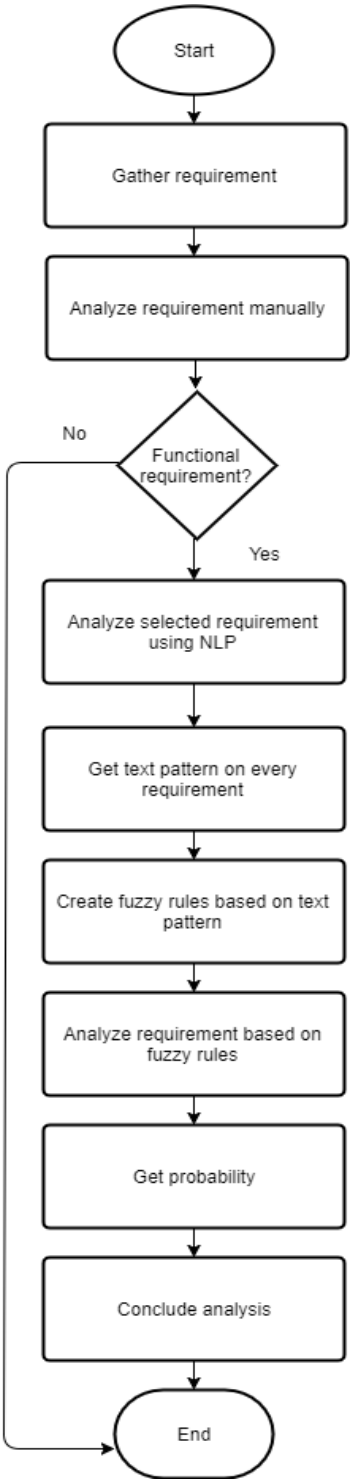

Figure 3: Flowchart of Analysis

Data selected for this research is an open source data, which is a case study project of University College London (UCL) [22]. The data gathered in the project is RALIC project (Replacement Access, Library and ID Card). It is a software project to enhance the existing access control system at UCL.

RALIC aims 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. RALIC is a combination of development and customization of an off-the-shelf system [22]. The scope of the project is shown in Table 2.

Table 2
RALIC Project Scope

Scope Item	Description
1	Replace swipe card readers with smart card readers
2	Source and install access card printers
3	Decide on card design and categories
4	Define user groups and default access rights
5	Provide a more accurate card holder database, save resources on manual data input, and facilitate automated provision and suspension of access and library borrowing rights
6	Issue new cards to staff, students, visitors and contractors
7	Replace the Library access control system
8	Use new cards at the Bloomsbury Fitness Centre

Due to the variety of stakeholders, requirements gathered for RALIC rather have conflicting requirements. For example, members of the UCL Development and Corporate Communications Office preferred the ID card to have UCL branding, but the security guards prefer otherwise for security reasons in case the cards are lost. Table 3 shows some of the requirements elicited for the project.

Table 3
Excerpt of RALIC Elicited Requirements

No.	Requirements
1	User friendly system which does not require a complex training programme
2	The card can be extended for future requirements, such as a digital certificate
3	To make the software interface easy to maintain
4	System is able to continue operation of up-to-date hardware and server operating systems
5	To improve the quality of the access systems database

A. *Manual Analysis*

The first stage of this research is to manually analyze the requirements and the first step is to separate the requirements into functional and non-functional requirements. In this analysis, 60 requirements are eliminated because they are non-functional requirements and only 328 functional requirements will be included in further analysis for the scope of this study.

The second step is to identify requirements that are either ambiguous or incomplete. There are 74 requirements that are considered ambiguous, 21 requirements are incomplete and 233 requirements are free from ambiguity and incompleteness. This manual analysis does not totally depend on the knowledge of the analyzer as some templates also guide on how to specify good requirements [23].

B. *Natural Language Processing*

This analysis aims to observe the pattern of the texts (requirements) to create the rules in the fuzzy logic. It involves the analysis of 233 texts using Stanford CoreNLP. The outputs from this analysis help to see the details of every single word from each requirement. From the output, each word is tagged with part-of-speech (POS) tag to understand the requirements further. The algorithm performs the analysis that allows every word from the text to be further broken down to their lemma, POS and the parsing of every word.

C. *Fuzzy Logic Analysis*

Let the universe of discourse X be the subset of real numbers R, $X = \{x_1, x_2, x_3, \ldots, x_n\}$. A fuzzy set $\tilde{A} = \{(x, \mu_A(x)) | x \in X\}$ in X is a set of ordered pairs, where $\mu_A(x)$ is called a membership function and $\mu_A(x) : X \rightarrow [0, 1]$. The membership function for fuzzy sets can take any values from the closed interval [0, 1]. The greater $\mu_A(x)$ is, the greater the truth of the statement that element x belongs to set $\tilde{A}$ is.

After observing the pattern of every text, the pattern is then used to create rules for the fuzzy set. Figure 4 shows the rules that are created from the observed text patterns. NN refers to noun, JJ is adjective, VB is verb, VBZ (verb, 3rd person singular present) and RB is adverb.

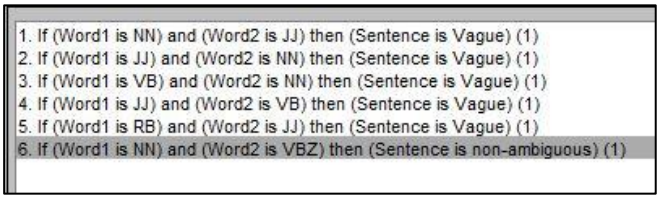

Figure 4: Fuzzy rules

V. Result and Discussion

From the experiment that had been conducted from text language analysis, the pattern of every text was identified and the pattern of words that cause a sentence to be vague and words that cause a sentence to be complete was recognized. Considering the first requirement from Table 3 as an example: "User friendly system which does not require a complex training programme" this requirement contains both words that is tagged with NN (User), JJ (friendly) and JJ (complex) making the requirements to be rather subjective instead of objective. Figure 5 shows the pattern of the vague requirement based on the chosen example.

(ROOT (FRAG (NP (NP (**NN User**) (**JJ friendly**) (NN system)) (SBAR (WHNP (WDT which)) (S (VP (VBZ does) (RB not) (VP (VB require) (NP (DT a) (**JJ complex**) (NN training) (NN programme))))))) (. .)))

Figure 5: Text pattern

By referring to the instances of probability values in Table 4, when the tagged words appear in the sentence, the chances for the sentence to be ambiguous is 50% and when the sentence contains more adjective (JJ) the probability shows that chances for it to be ambiguous is higher.

Table 4
Probability of Vague Requirements

	Word1	Word2	Probability
Membership Function Value	0.5	0.5	0.383
	1	1	0.5
	0.133	1	0.641
	0.225	0.15	0.396
	0.381	0.441	0.344
	0.821	0.177	0.375
	0.271	0.714	0.334

The example of text pattern output as shown in Figure 7 and probability values as in Table 4 are analyzed in accordance to Algorithm 1.

Algorithm 1: NLP and Fuzzy Logic Analysis

Input: Stakeholders requirements {$R_{1,}$…, Rn}
Output: Text pattern and probability
Begin
1. *Input requirements;*
2. *Enter command to tokenize every input;*
3. *Print text pattern output;*
4. *Check text pattern for every requirement;*
5. *Create rules on fuzzy based on text pattern identified;*
6. *Analyze probability with fuzzy;*
7. *Get probability output;*

End

Figure 6: Algorithm 1

From the obtained results, the probability of a requirement to be vague is most likely due to the existence of vague words such as *user friendly*, *simple*, *reduce*, and *quick*. When these vague words are found in requirements, it can cause the requirements to be vague since the words cannot be measured. However, if the vague word is accompanied by another condition that is measureable such as *ninety percent simpler*, the requirements will be complete requirements.

A combination of NN (noun) and JJ (adjective) are more likely to produce a noun phrase (NP) in which the word tends to be descriptive such as *very few*, *extremely large*, *small amount*. By conducting further analysis using fuzzy, the impact of the vague words can be seen clearly. From Table 4, the results show that the higher the presence of *word2* in the requirements, the likelihood of the requirements to appear vague is higher. In addition, *word2* are the words that are tagged as JJ (adjective), NN (noun), RB (adverb) and VBZ (verb, 3rd person singular present). Words that are normally tagged with these words cannot be measured such as *efficient*, *quick*, *easy*, and *reliable*.

The results also show that the combination of noun (NN) word and adjective (JJ) word is most likely to produce noun phrase (NP). Noun phrase is normally vague verb that seems qualitative rather than quantitative. Examples of noun phrase include *very few*, *extremely large* and *user friendly*. Thus, by repeating this experiment to more ambiguous requirements, more text patterns and probability values can be derived to identify vagueness that could also guide requirements engineers or system analysts on what text patterns to be avoided to gain high quality requirements.

VI. Conclusion and Future Work

Linguistically, vague is defined as something that are not clearly or explicitly stated or expressed. Although it is common to use vague words in daily conversation, in the case of requirements, it is best to be avoided. From the conducted experiment, we can conclude that a requirement comprising a vague word is most likely will turn out to be a vague requirement. However, if these vague words are followed by words that can measure the vagueness, the probability for such requirements to be vague is low.

In detecting ambiguous requirements especially in term of vagueness, it is very important to look at the word that is either noun, adjective or adverb. This word or word phrase is rather subjective than objective in which without a determiner, the probability for the requirements to be ambiguous is higher when such words present in a sentence. Thus, fuzzy inference technique could further analyze the combination of words that might produce ambiguous requirements. Through the probability, it helps to detect combination of words that most likely causes vagueness to the requirements.

Future work includes further analysis to capture more text patterns for other types of ambiguity. In-depth validation and verification of the results are necessary to ensure the accuracy of the obtained results using the Mamdani fuzzy inference.

Acknowledgment

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

References

[1] D. Pandey, U. Suman and A. K. Ramani, "An effective requirement engineering process model for software development and requirements management," *2010 International Conference on Advances in Recent Technologies in Communication and Computing*, Kottayam, 2010, pp. 287-291.

[2] E. Parra, C. Dimou, J. Llorens, V. Moreno and A. Fraga, "A methodology for the classification of quality of requirements using machine learning techniques," *Information and Software Technology*, vol. 67, Nov. 2015, pp. 180-195.

[3] ISO/IEC/IEEE International standard, "Systems and software engineering - Life cycle processes - Requirements engineering", in *ISO/IECIEEE 29148:2011(E)*, pp. 1-94, Dec. 1 2011.

[4] J. Kazmier, B. Berenbach, D. J. Paulish and A. Rudorfer, *Software and Systems Requirements Engineering: In Practise*. New York, NY: McGraw-Hill Education, 2009, pp. 39-72.

[5] E. Kamsties, "Understanding ambiguity in requirements engineering, Engineering and Managing Software Requirements," in *Engineering and Managing Software Requirements*, A. Aurum, and C. Wohlin, Eds. Berlin, Heidelberg: Springer, 2005, pp. 245-266.

[6] S. F. Tjong, *Avoiding Ambiguity in Requirements Specifications*. Doctoral dissertation, University of Waterloo, 2008.

[7] A. Nigam, N. Arya, B. Nigam, and D. Jain, "Tool for automatic discovery of ambiguity in requirements," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 5, pp. 350-356, Sep. 2012

[8] A. Ferrari, G. Lipari, S. Gnesi and G. O. Spagnolo, "Pragmatic ambiguity detection in natural language requirements," in *2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE*), Karlskrona, 2014, pp. 1-8.

[9] F. Meziane and S. Vadera, "Artificial intelligence in software engineering current developments and future prospects," in *Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects*, Hershey, New York, USA: IGI Global, 2010, pp. 273-294.

[10] IEEE/ISO International standard, "Quality management and quality assurance", in *IEEE ISO 8402:1995*, 1995.

[11] C. H. Wang, *A study of membership functions on Mamdani-type fuzzy inference system for industrial decision-making*. Thesis and dissertation, University of Lehigh, 2015.

[12] C. Arora, M. Sabetzadeh, L. Briand and F. Zimmer, "Automated checking of conformance to requirements templates using natural language processing," in *IEEE Transactions on Software Engineering*, vol. 41, no. 10, Oct. 1 2015, pp. 944-968,.

[13] B. Gleich, O. Creighton and L. Kof, "Ambiguity detection: towards a tool explaining ambiguity sources," *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2010, pp. 218-232.

[14] A. K. Massey, R. L. Rutledge, A. I. Antón and P. P. Swire, "Identifying and classifying ambiguity for regulatory requirements," *2014 IEEE 22nd International Requirements Engineering Conference (RE*), Karlskrona, 2014, pp. 83-92.

[15] J. M. Davril, M. Cordy, P. Heymans and M Acher, "Using fuzzy modeling for consistent definitions of product qualities in requirements," *2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, Ottawa, 2015, pp. 1-8.

[16] A. Umber and I. S. Bajwa, "Minimizing ambiguity in natural language software requirements specification," *2011 Sixth International Conference on Digital Information Management*, Melbourn, 2011, pp. 102-107.

[17] S. Bhardwaj and A. K. Goyal, "A comparative analysis of agent oriented requirements engineering frameworks," *International Journal of Computer Applications*, vol.87, no.8, Feb. 2014.

[18] M. Kumar, M. Shukla and S. Agarwal, "A hybrid approach of requirements engineering in agile software development," *2013 International Conference on Machine Intelligence on Research Advancement*, Katra, 2013, pp. 515-519.

[19] A. B Siddique, S. Qadri, S. Hussain, S. Ahmad, I. Maqbool, and A. K. N. Khan, "Integration of requirements engineering with UML in software engineering practices," *Sci. Int. (Lahore)*, vol. 26, no. 5. pp. 2157-2162, 2014.

[20] F. Beritelli, G. Cilia and A. Cucè, "Small vocabulary word recognition based on fuzzy pattern matching," in *Proc. of the European Symposium on Intelligent Techniques*, Crete (Greece), 1999.

[21] L. Baresi, L. Pasquale, and P. Spoletini, "Fuzzy goals for requirements-driven adaptation," *18th IEEE International Requirements Engineering Conference*, Sydney, 2010, pp. 125-134.

[22] S. L. Lim, *Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation*. Doctoral dissertation, University of New South Wales, 2010.

[23] D. Firesmith, "Specifying good requirements," *Journal of Object Technology*, vol. 2, no. 4, pp. 77-87, 2003.