International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 04, 2020
ISSN: 1475-7192

RAUMM: A Model for Requirement Ambiguity and Uncertainty Management in Software Requirement Specification

[1]Halima Sadia, [2]Syed Qamar Abbas

Abstract

The Requirement specification document serves as a contract between the developer and the various stakeholders of the system under development. For better quality control, every requirement needs to be verified before being based line. According to the report of Systems Sciences Institute at IBM, it cost 6 times more when fixing a bug in implementation phase than fixing the same bug during the design phase. Furthermore, it costs 15 times more to fix bugs in testing phase if left during the design phase.

This paper attempts to propose a model Requirement Ambiguity and Uncertainty Management Model (RAUMM) to deal with requirement ambiguity and uncertainty at early stage of software development. A tool has been developed to detect four types of ambiguities i.e. lexical, syntax, syntactic and referential ambiguity in requirements specification document. A compilation of potential ambiguous words is done from the taxonomy present in Ambiguity Handbook. This proposed tool will help the analysts to detect potential ambiguities. Several requirements writing rules have been taken from literature. Ambiguity Assessment has been performed by implementation of Adaptive Fuzzy Neural Network and ambiguities detected by tool have been taken as input. Relationship between Uncertainty and ambiguity has been analyzed finally.

Keywords: *Software Engineering, Software Requirement Specification, Requirement ambiguity, Lexical ambiguity, Syntax ambiguity, Syntactic ambiguity, Referential ambiguity, Adaptive Neuro Fuzzy System.*

I. Introduction:

Software Industry has emerged as a major component in the IT sector. Software development is a high risk activity and involves high failure rates. Making software involves requirement collection. Users, Developers and various stakeholders are involved in the requirement collection process. According to the Standish Group report, Feb, 2019, 83.9% of IT projects fail completely or partially (Chaos Report, 2019). The top 3 reasons for failure being *Lack of user input, *Incomplete Requirements & Specifications and *Changing Requirements & Specifications. Requirement collection is perceived as one of the most important activities resulting in successful

[1] Department of Computer Science & Engineering, Integral University, Lucknow-226026
[2] Ambalika Institute of Technology & Management, Dr. A.P.J. Abdul Kalam Technical University, Lucknow-226026

software production. Standish Report (Chaos Report, 2019) indicates that requirement “is a primary source of software project risk and software defects”. What the stakeholders of the system want are Requirements. The degree of uncertainty between the developer’s view and what the stakeholders expect is what we call Requirement Risk. The Risk involved at the requirements phase always lowers the performance of the product under development.

The pressure of releasing the software product within the given budget and timeline, often forms the basis of ignorance of verifying the requirements and removing the VUCA risks. Software projects are characterized to have a very high failure rate, so to deliver a successful software project it requires a methodology that deals with the potential requirement related risks determined at an early stage. Literature shows several types of risks that are involved. Potential requirement related risks are (Halima Sadia, Syed Qamar Abbas, 2019):

- Requirement Volatility & Uncertainty
- Requirement Complexity
- Requirement Ambiguity

Literature reveals that Requirement Volatility, Requirement Uncertainty, Requirement Complexity and Requirement Ambiguity i.e. VUCA are the basic sources of risks for other risks too. The literature review reveals that research work have been carried out on these factors (VUCA) independently, but no formal model using these VUCA risks is available (Halima Sadia, Syed Qamar Abbas, 2019). The causes and respective effects are summarized in ‘VUCA’. VUCA is a trendy managerial acronym that has its root from a different (military) background (Stiehm, J. H., Townsend, N.W., 2002). To reduce requirement related risk, it is important to rigorously work on the problem definition before hitting the solution space. The Design team carries the process of requirement collection and fulfillment in parallel. This leads to indeterminate requirement specifications.

Due to this gap formed by users understanding of the system and the technical limitations of the actual implementation of the system, Ambiguity creeps in. Ambiguity is a result of establishing vague/unclear/incomplete interpretations of the requirements. If ignored in the requirements phase, this spirals into defects and delays and further defects and further delays in the software product development. In order to detect these requirements related risks in the requirement collection phase, avoid developmental delays and create valid test cases, ambiguity detection and review is an important activity to be carried out.

This paper is organized as follows:

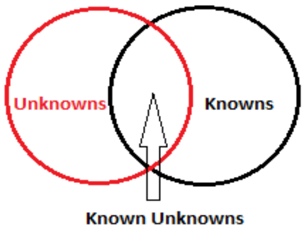
i. Definitions and Background knowledge of the terminology used.
ii. Existing Literature on how to resolve Requirement Ambiguity.
iii. Formation of Ambiguity Types and Attributes.
iv. Development of an automated tool to detect different ambiguity types in NL SRS.
v. Validation

1.1 Definitions and Background:

This section discusses the background understandings of the terminologies that this paper aims to address.

1.1.1 Requirement Uncertainty

Uncertainty is "the state of knowledge in which each alternative leads to a set of results but the probability of occurrence of each outcome is unknown to the decision maker" (Jauch, L., & Kraft, K., 1986). Requirement Uncertainty can be a result of lack of knowledge in a range of areas in requirements collection. It is a situation where the present scenario of knowledge is unpredictable in terms of the circumstances, conditions, consequences and events in which the software will be developed. It can be identified as a gap between the users need and the information that the developers have (Nidumolu, S., 1996). Uncertainty imposes a threat, until it shapes into a critical problem.



Uncertainty Spectrum

Unknown Unknowns	**Known Unknowns**	**Knowns**
Complete lack of knowledge results in uncertainty.	*When partial information exists, probability exists that a event may occur and the impact on objectives can also be evaluated.*	*When complete information exists about what is to happen and the possible impact on project objectives*

Figure 1: Uncertainty Spectrum

Requirement Uncertainty can be an important factor (Wideman, R. M. 1992, E. Lutters, F. van Houten. 2013) on which the success of the software product under development depends. The Uncertainties should be identified in the requirements phase itself and effective management should be carried out.

1.1.2 Requirement Ambiguity

The Software Specification document is written after collecting requirements from different stakeholders. While writing these requirements ambiguities can occur. A Requirement is ambiguous if it has more than one interpretation. Studies reveal that 71.8% of the requirement specification document is written in natural language, 15.9% uses structured natural language, and 5.3% of the requirement specification is written in formal language (Daniel M. Berry, 2004). Taking these ambiguous requirements as a base line for software development often results in project failure. Due to this gap formed by users understanding of the system and the technical limitations of the actual implementation of the system, Ambiguity creeps in. Ambiguity is a result of establishing vague/unclear/incomplete interpretations of the requirements. If ignored in the requirements phase, this spirals into defects and delays and further defects and further delays in the software product development. In order to detect these requirements related risks in the requirement collection phase, avoid developmental delays and create valid test cases, ambiguity detection and review is an important activity to be carried out.

1.1.3 Software Requirement Specification (SRS)

The Software Requirement Specification is defined as a contract between the user and the developer. Requirements Specification provides middling between developers and users of the system. According to IEEE "An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation" (IEEE, 1993). However IEEE has not stated any definition on unambiguous requirement specification. In fact, from the requirement specification perspective, someone is always there who interprets differently from the others. Therefore it is said that there are no unambiguous specifications, rather all specifications are useful specifications.

II. Related Work

Requirement Ambiguity is a very complex concept and can have multi-dimensional effects (Stiehm, J. H., Townsend, N.W., 2002). Research reveals that around 87.7 % of software requirements are documented using natural language (NL) (Beg, R et al., 2008). Due to the potential qualities of ease of understanding and flexibility of NL, Developers use it as a means to write the specifications document. However, despite its advantages the use of Natural language is also prone to ambiguities and vagueness that results in ambiguous and uncertain requirements. Ambiguity becomes an inherent characteristic of documents written in natural language. The possibility of having more than two ways of interpreting an expression is termed as Ambiguity (Gill, K.D. et al., 2014). More than one interpretation of the user requirements may lead to incorrect implementation and thus result in a product deviated from the actual user requirements. A number of taxonomies of ambiguity and its types are present in literature. The following table (Table 1) summarizes the types of ambiguities defined in literature (Sandhu G, 2015; Alessio Ferrari et al. 2014; Yannick Versley, 2008).

Ambiguity Types	Definition	Sub-Types	Example
Lexical Ambiguity [16]	Single word having multiple meanings.	Homonymy Polysemy	e.g. return : A Key on a typewriter/Computer Keyboard, An income tax return A Coming back Getting something back

			again
Syntactic or Structural Ambiguity [16]	When a sentence or phrase can be interpreted to have more than one meaning.	Attachment Ambiguity Analytical Ambiguity Coordination Ambiguity Elliptical Ambiguity	e.g. The chicken is ready to eat I saw the man with the binoculars Look at that dog with one eye
Semantic [16]	When sentences have different interpretations with respect to the scope and context of the word in which it has been used. It is with respect to the logical form of sentence.	Can be caused by Scope Ambiguity Coordination Ambiguity Referential Ambiguity	e.g. Amy's car Amy's husband Amy's greatest fear
Pragmatic Ambiguity [17]	Arises when the same statement has different interpretations due to the difference in the domain knowledge of the readers of the statement.		e.g. two readers can interpret the same requirement based on their domain knowledge or perception
Referential / Anaphoric Ambiguity [18]	When the entities are well defined but it is not clear which noun is being referred to.		e.g. The brick fell on the computer but it is not broken, John met Mark before he went to the market, Bob said to Joe that he must leave.

Table 1: Types of Linguistic Ambiguities

For getting a more clear understanding of ambiguity, its nature, sources and techniques for detection, a quick literature review needs to be made. The Requirement Ambiguity detection approaches are broadly classified into Manual, Semi-Automatic using NLP and Semi-Automatic using Machine Learning Techniques. In addition a number of tools have been developed to detect potential ambiguities (Table 2). The Manual approach of ambiguity detection is based on Inspection techniques and Review techniques. In Inspection each requirement is manually searched for any potential ambiguities by the stakeholders of the system under development. It includes checklist and scenario-based reading techniques (Denger C, Berry DM,. 2003; Hill E et al., 2008; Kamsties E et al., 2001; Popescu D et al., 2008; Anda B, Sjberg DIK. 2002) . In Review (Havasi C et al., 2007) the reviewers first search for potential ambiguities. Secondly they rate the ambiguities according to their severity and then based on this the requirements are analyzed and corrected by respective stakeholders. The Semi-Automatic using NLP uses two approaches Ontology and Natural Language Patterns (Hussain I et al., 2007; Polpinij J. 2009). In Ontology the less significant words are found and removed in the SRS. Example tools are Concept-Net3, WordNet, Brandeis Semantic Ontology, RESI, ResearchCyc, YAGA, Ontology Based Text Classification Method (Wijewickrema CM., 2014; G. A. Miller et al., 2006; Cycorp Inc., 2015; C. Havasi et al., 2007; F. M. Suchanek et al., 2007). Finally, the document is rewritten by removing potential ambiguities (Wang X-Zet al., 2012). Implementation names newspeak, ANLT, GSA.

Under the Semi-Automatic using Machine Learning Techniques category four techniques are commonly used. Decision-Tree works by adding features in a tree. The highest useful features are added first. Searching in the tree continues unless all training examples are extensively exhausted. Implementations include Decision-Tree Text Classification Technique, Test-Based Risk Identification Methodology (TBRIM), fuzzy decision (J. J. Romano, J. D. Palmer, 1998; Polpinij J, Ghose A. 2008.) Another technique under the same category, Support Vector Machine (SVM) uses Parts Of Speech (POS) tagging. SVM is implemented as SVM-light-TK software, Poly-SVM, combined Poly-SVM and PAMCT (Seijas L, Segura E., 2009). In Naïve Bays (NB), word probability and word count is used. Both are applied to the training data to built Bag of Words (BOW). Implementations include NB classifier to detect ambiguity in requirement, NB classifier to classify the antecedents and anaphoric ambiguity, NB text classifier to detect coordinating ambiguity in natural language (Clark A, et al., 2013; Brown PF et al., 1992). The last technique under this category is N-Gram Modeling (Sharma R et al., 2014). The working principle is based on Probabilistic Language Model and Prediction Using Words.

Author	Tool Name	Functionality
(Gleich, B., et al., 2010)	**POS Tag Based Tool**	Use of regular expressions, part of speech tagging (POS) and list of ambiguous words Lack of calculating the percentage of the detected ambiguity.
(Popescu, D., et al., 2007)	**DOWSER TOOL**	Dowser parses the requirements using constraining grammar. It does not detect ambiguity automatically.
(Femmer, H. et al., 2014)	**QUALICEN**	Use of POS tagging, morphological analysis and dictionaries.
(Korner, S.J., T. Brumm, 2009)	**RESI**	Tags with part of speech (POS). Application of Ontologies WordNet, ResearchCyc, ConceptNet and YAGO.
(Umber A. et al., 2011)	**SR-ELICITOR**	Use of Semantic of Business Vocabulary (SBVR).

		Use of Rules to capture NL SRS document.
(Bajwa I. et al., 2012)	**NL2OCL**	It solves syntactic ambiguity only. It translates NL SRS document to formal constraints. Stanford POS tagger and the Stanford Parser for syntactic analysis of English specification.

Table 2: Requirement Ambiguity Detection Tools

It has been observed that various ambiguity detection techniques and tools are present in literature (Khin Hayman Oo et al., 2018.) However it has been observed that no tool focuses on the combined detection of the ambiguity types present. This paper proposes a methodology to detect Lexical, Semantic, Syntactic and Referential Ambiguity in the Requirements Document and further relate these ambiguous requirements to uncertainties present.

III. Proposed Methodology

Requirement Ambiguity and Uncertainty Management Model (RAUMM) :

This research work proposes Requirement Ambiguity and Uncertainty Management Model (RAUMM**).** This model performs the task into the following three phases:

1. Requirement Ambiguity Detection
2. Requirement Ambiguity Assessment
3. Analysis of Ambiguity and Uncertainty

3.1 Requirement Ambiguity Detection: The Workflow for detection of ambiguous requirements is illustrated in figure 2. The Implementations and details are discussed in the subsequent sections.

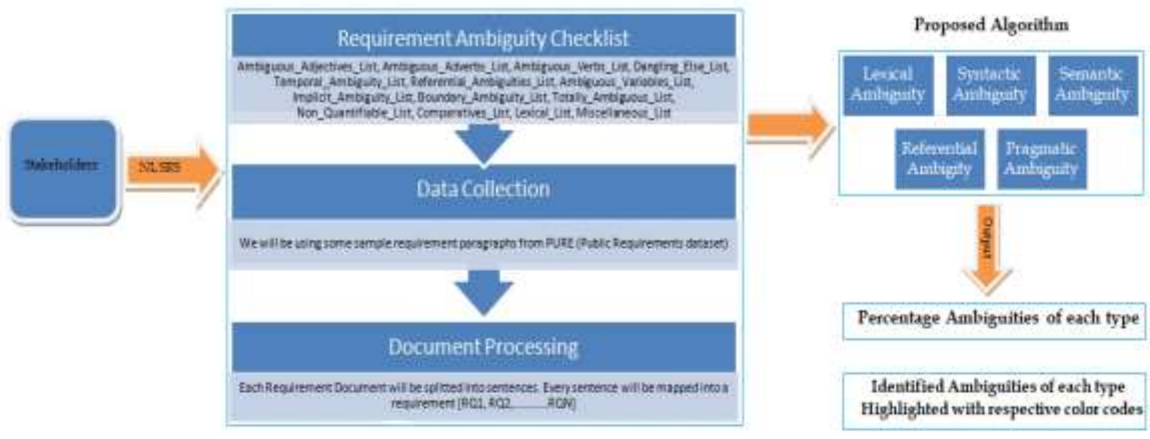

Stakeholders
NL SRS
Requirement Ambiguity Checklist
Ambiguous_Adjectives_List, Ambiguous_Adverbs_List, Ambiguous_Verbs_List, Dangling_Else_List, Temporal_Ambiguity_List, Referential_Ambiguities_List, Ambiguous_Variables_List, Implicit_Ambiguity_List, Boundary_Ambiguity_List, Totally_Ambiguous_List, Non_Quantifiable_List, Comparatives_List, Lexical_List, Miscellaneous_List
Data Collection
We will be using some sample requirement paragraphs from PURE (Public Requirements dataset)
Document Processing
Each Requirement Document will be splitted into sentences. Every sentence will be mapped into a requirement [RQ1, RQ2.........RQN]
Proposed Algorithm
Lexical Ambiguity
Syntactic Ambiguity
Semantic Ambiguity
Referential Ambigity
Pragmatic Ambiguity
Output
Percentage Ambiguities of each type
Identified Ambiguities of each type Highlighted with respective color codes

Figure 2: Workflow for Detecting Ambiguities in NL SRS

Step1: Requirement Ambiguity Checklist

To find and manage potential ambiguities in Requirements Specification Document, the root causes needs to be identified. To start with, a checklist of potential ambiguous words is built. This checklist will form the dataset for the proposed algorithm. We have taken the previous literatures as a guideline and a checklist of potentially ambiguous words is prepared (Table 3).

Potential Ambiguous Words	**Ambiguity Checklists**	**Description**
ALL, EACH, ANY, EASY, APPROPRIATE, ACCEPTABLE, CUSTOM, ACCURATE, EFFICIENT, EVERY, FEW, FREQUENT, IMPROVED, ESSENTIAL, INFREQUENT, INTUITIVE, INVALID, MANY, MOST, NORMAL, ORDINARY, RARE , PERIODICALLY, SAME, SEAMLESS, SEVERAL, SIMILAR, SOME, STANDARD, IMMEDIATELY, THE TYPICAL, THE COMPLETE, THE AVERAGE, THE ENTIRE, TRANSPARENT, USER-FRIENDLY, TYPICAL, USUAL, VALID, THE RARE, SUFFICIENT, THE OCCASIONAL	Ambiguous Adjectives	Word belonging to one of the major form classes in any of numerous languages and typically serving as a modifier of a noun to denote a quality of the thing named, to indicate its quantity or extent, or to specify a thing as distinct from something else
ONLY, BY AND LARGE, ACCORDINGLY, ALMOST,APPROXIMATELY, COMMONLY, EFFICIENTLY, CUSTOMARILY, HARDLY EVER, FREQUENTLY GENERALLY, IN GENERAL INFREQUENTLY, JUST ABOUT, INTUITIVELY, MORE OFTEN THAN NOT, NEARLY, MORE OR LESS, MOSTLY, NOT QUITE, ON THE ODD OCCASION, NORMALLY, OFTEN, ORDINARILY, ROUGHLY, RARELY, SELDOM, SEAMLESSLY, SIMILARLY, SOMEWHAT, SOMETIME, USUALLY, TRANSPARENTLY, VIRTUALLY,	Ambiguous Adverbs	Word belonging to one of the major form classes in any of the numerous languages, typically serving as a modifier of a verb, an adjective, another adverb, a preposition, a phrase, a clause, or a sentence, expressing some relation of manner or quality, place,

TYPICALLY, ALSO, OTHERS		
VALIDATE, VERIFY, ADJUST, INDICATE, ALTER, CALCULATE, MANIPULATE, MAY, SUPPORT, UPDATE, PROVIDE, ENABLE, MINIMIZE, MAXIMIZE, CHANGE, COMPUTE, EDIT, OPTIMIZE, CUSTOMIZE, COMPARE, CREATE, MODIFY, MATCH, AMEND, PROCESS DERIVE, CONVERT, MIGHT, DETERMINE, PRODUCE, PERFORM, IMPROVE	Ambiguous Verbs	Word that characteristically is the grammatical centre of a predicate and expresses an act, occurrence, or mode of being, that in various languages is inflected for agreement with the subject, for tense, for voice, for mood, or for aspect, and that typically has rather full descriptive meaning and characterizing quality but is sometimes nearly devoid of these especially when used as an auxiliary or linking verb
MUST BE, WILL BE, IS ONE OF, SHOULD BE, COULD BE, MAY BE, CAN BE, SHALL, PROBABLY, USUALLY, POSSIBLY	Dangling Else	The requirement has no other exit when one case is not met
ANNUALLY, AT THE APPROPRIATE TIME, TWICE A YEAR, BIMONTHLY, QUARTERLY, MONTHLY, BIWEEKLY, AFTER, , AT A GIVEN TIME, DAILY, EVERY OTHER WEEK, EVERY OTHER MONTH, YEARLY, WEEKLY, TWICE A MONTH, SOON, QUICKLY, FAST, LATER, IN A WHILE, UNTIL, BY, WHEN	Temporal Ambiguity	Words that has time/duration type that invites multiple interpretation. Un-boundary timing or duration
ABOVE, SUCH, THESE, THOSE, AFTER, BEFORE, NEXT, PREVIOUS, BELOW, THEY, IT, THEM, THIS	Referential Ambiguities	Sentence that contains more than one requirement in a sentence. Sentence contains explicit references to (not numbered sentences, not defined, not described, no glossary)
THE WINDOW, THE STATUS, THE INFORMATION, THE APPLICATION, THE DATABASE, THE FRAME, THE MESSAGE, THE SCREEN, THE SYSTEM, THE COMPONENT, THE FILE, THE DATA, THE MODULE, THE RULE, THE FIELD, THE TABLE, THE VALUE, THE PAGE	Ambiguous Variables	Common word that invites vague interpretation and understanding. Too generic.

AS NECESSARY, ALSO, ALTHOUGH, BESIDES, EVEN THOUGH, AS WELL, FURTHERMORE, FOR ALL OTHER, BUT, MOREOVER, WHEREAS, HOWEVER, NOTWITHSTANDING, IN ADDITION TO, LIKEWISE, OTHERWISE, AS REQUIRED, ON THE OTHER HAND, THOUGH, YET, STILL, UNLESS	Implicit Cases of Ambiguity	Apart from the cases given there are some implicit cases of ambiguities. If neglected there is a chance that they result in ambiguous requirements.
AMONG, UPTO, INCLUDING	Boundary Ambiguity	It has no definite boundary of true or false (or between yes and no).
*Incomplete sentences ending with “?”, “ TBD”, “etc.”, “/”, brackets *AND and OR in the same sentence *ANY, INCLUDE, MINIMUM, MAXIMUM, BOTH	Totally Ambiguous	If we find these conditions in the requirements then there are maximum chances that the respective requirement is an ambiguous one. The Requirement needs to be verified further.
FLEXIBLE, EFFICIENT, MODULAR, ACHIEVABLE, ADEQUATE, ACCOMPLISH, POSSIBLE (OR POSSIBLY), CORRECT (OR CORRECTLY), MINIMUM REQUIRED, BETTER, HIGHER, MINIMUM ACCEPTABLE, FASTER, LESS, SLOWER, INFREQUENT, TO THE EXTENT SPECIFIED, TO THE EXTENT REQUIRED, TO BE COMPATIBLE, TO BE ASSOCIATED WITH	Non-Quantifiable	Words and phrases that cannot be quantified, such as
"EARLIEST," "LATEST," "HIGHEST	Comparatives	“est" should always be a suspect.
INSTANTANEOUS, SIMULTANEOUS, ACHIEVABLE, COMPLOTS, FINISH, DEGRADED, A MINIMUM NUMBER OF, NOMINAL/NORMAL/AVERAGE, MINIMUM, STEADY-STATE, COINCIDENT, ADJACENT, SYNCHRONOUS	Miscellaneous	Words and phrases whose meaning is of dispute between the developer and the stakeholders.
UNTIL, DURING, THROUGH, AFTER, AT,	Lexical Ambiguity	Words that potentially cause lexical

COULD, SHOULD, MIGHT, USUALLY, NORMALLY, ACTUALLY, 100%, ALL ERRORS, HE, SHE, IT, FAST, THIS		ambiguities.

Table 3: Requirement Ambiguity Checklist

Step 2: Data Collection:

The Input for the Ambiguity Detection Algorithm is SRS Document. We will be using some sample requirement paragraphs from PURE (Public Requirements dataset). The PURE dataset is a collection of 79 requirements document that are available on the Web (Alessio Ferrari, et al., 2017).

Step 3: Document Processing:

For checking for potential requirement ambiguities it is required that the SRS document is processed line by line. We will use a sentence splitting function that will break the SRS paragraphs into sentences. Every Document is given a Unique Doc Name, defined as alphanumeric ID identifying the individual documents. For each document, All Requirements are structured and tagged with a Unique ID {RQ1, RQ2,……..RQN}. So each pair [Doc Name, RQ1,RQ2…….RQN] defines the structured requirements belonging to the respective SRS Document identified by the Unique Doc Name.

*However, due to the inherent limitation of unstructured nature of the SRS, we will implement the Algorithm using sample requirements paragraph from the requirements document present in the PURE Dataset. After that the implementation can be extended to any of the SRS Documents. After completing the document processing phase we get the following two lists:

- The Split Sentence function maps each sentence into a Req_Set {R1, R2…….RN}.
- Tot_Req [denotes the total number of requirements in the input SRS].

Step 4: Identifying Potential Ambiguities

For every Document and Requirements pair [Doc Name, RQ1,RQ2…….RQN] we will compare every Requirement Sentence to the Training Dataset that has been formed by extensively collecting a checklist of potential ambiguous words called the **Requirement Ambiguity Checklist**. The Training Dataset constitutes of the following Ambiguous Requirement checklists based on the Ambiguity Attributes obtained from the Ambiguity Handbook literature represented in Table 3.

Ambiguous_Adjectives_List, Ambiguous_Adverbs_List, Ambiguous_Verbs_List, Dangling_Else_List, Temporal_Ambiguity_List, Referential_Ambiguities_List, Ambiguous_Variables_List, Implicit_Ambiguity_List, Boundary_Ambiguity_List, Totally_Ambiguous_List, Non_Quantifiable_List, Comparatives_List, Lexical_List, Miscellaneous_List

Step 5: Detect Lexical, Syntactic, Semantic, Referential Ambiguities

Condition 1: Detecting Lexical Ambiguity

- For every Requirement in the Req_Set{R1, R2……RN}compare each word with

- the Lexical_List, Totally_Amiguous_List, Boundary_Ambiguity_List, Comparatives_List,
- Non_Quantifiable_List, Miscellaneous_List, Implicit_Ambiguity_List. If a match is found with
- any of the lists then the word will be measured as Lexical Ambiguity and it will be stored in the
- Lexical-Req List and a variable Lexical_Count is incremented.

Condition 2: Detecting Syntactic Ambiguity

- For every Requirement in the Req_Set{R1, R2……RN}compare each word with
- the Ambiguous_Adjectives_List, Ambiguous_Adverbs_List. If a match is found in any of the lists
- then the word will be measured as Syntactic Ambiguity and will be stored in the Syntactic-Req List
- and a variable Syntactic_Count is incremented.

Condition 3: Detecting Semantic Ambiguity

- For every Requirement in the Req_Set{R1, R2……RN}check for the following matches:
- Coordination Ambiguity: when more than one AND and OR are present in a sentence.
- Scope Ambiguity : match Scope_List{ every, each, all, some, several, a, not}in a sentence
- If a match is found in any of the lists then the word will be measured as Semantic Ambiguity and will
- be stored in the Semantic-Req List and a variable Semantic_Count is incremented.

Condition 4: Detecting Referntial Ambiguity

- For every Requirement in the Req_Set{R1, R2……RN}compare each word with the
- Referential_Ambiguities_List. If a match is found within the list then the word will be measured as
- Referential Ambiguity and it will be stored in the Referential-Req List and a variable
- Referential_Count is incremented.

Step 6: Calculate Percentage Ambiguities of each type in the NL SRS

Once the whole NL SRS has been processed for identifying the potential ambiguities, compute the average ambiguities of all types of ambiguities as:

Lexical Ambiguity = Lexical_Count / Tot_Req *100

Syntactic Ambiguity = Syntactic_Count / Tot_Req *100

Semantic Ambiguity = Semantic_Count / Tot_Req *100

Referntial Ambiguity = Referential_Count / Tot_Req *100

Step7: Identified Ambiguities of each type highlighted with respective color codes

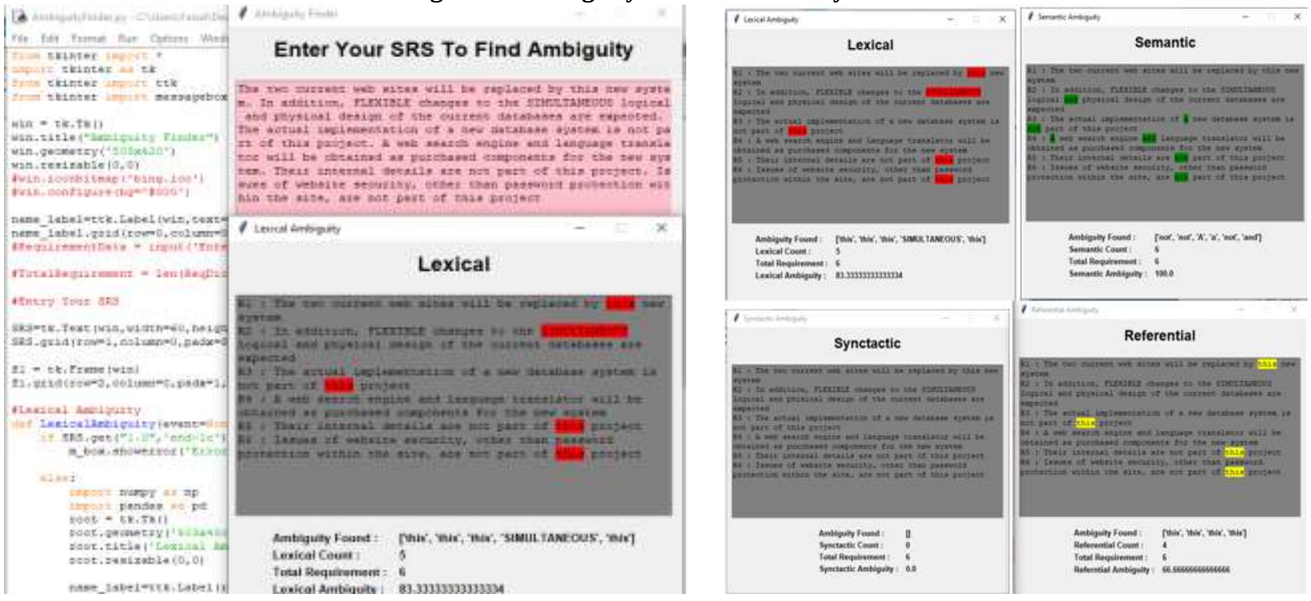
After processing all the Requirements and calculating the percentage ambiguities of each type the following output will be generated. For every requirement [RQ1, RQ2…….RQN] obtained from the SRS the corresponding match in the Requirement Ambiguity Checklist will be highlighted with respective color codes (Table 4). The System Analyst can easily find potential ambiguities by seeing the highlighted words. The color codes will highlight respective ambiguities. So after the ambiguities are detected this output can be used by the system analyst to verify the requirements and remove the potential ambiguities from the SRS.

Color Code	Ambiguity Type	Color Code	Ambiguity Type	Color Code	Ambiguity Type	Color Code	Ambiguity Type
🟥	Lexical Ambiguity	🟪	Syntactic Ambiguity	🟩	Semantic Ambiguity	🟨	Referential Ambiguity

Table 4: Color Code for highlighting Ambiguity Types

Tool for Ambiguity Detection: A tool for detection of ambiguities in requirement **AmbiguityFinder** has been developed on the basis of above mentioned algorithm. It has been developed in ***Python.*** Potential ambiguous words as mentioned in checklist are used baseline words to detect ambiguities in SRS. The tool (Figure 3) is based on the algorithm mentioned in previous subsection.

Figure 3: Ambiguity Finder Tool in Python



```
from tkinter import *
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

win = tk.Tk()
win.title("Ambiguity Finder")
win.geometry('505x420')
win.resizable(0,0)
#win.iconbitmap('bing.ico')
#win.configure(bg='#000')

name_label=ttk.Label(win,text=
name_label.grid(row=0,column=
#RequirementData = input('Ente

#TotalRequirement = len(ReqDi

#Entry Your SRS

SRS=tk.Text(win,width=40,heig
SRS.grid(row=1,column=0,padx=

f1 = tk.Frame(win)
f1.grid(row=2,column=0,padx=1,

#Lexical Ambiguity
def LexicalAmbiguity(event=
    if SRS.get("1.0",'end-1c')
        m_box.showerror('Error

    else:
        import numpy as np
        import pandas as pd
        root = tk.Tk()
        root.geometry('
        root.title('Lexical Am
        root.resizable(0,0)

        name_label=ttk.Label(
```

Enter Your SRS To Find Ambiguity

The two current web sites will be replaced by this new system. In addition, FLEXIBLE changes to the SIMULTANEOUS logical and physical design of the current databases are expected. The actual implementation of a new database system is not part of this project. A web search engine and language translator will be obtained as purchased components for the new system. Their internal details are not part of this project. Issues of website security, other than password protection within the site, are not part of this project

Lexical

R1 : The two current web sites will be replaced by this new system
R2 : In addition, FLEXIBLE changes to the SIMULTANEOUS logical and physical design of the current databases are expected
R3 : The actual implementation of a new database system is not part of this project
R4 : A web search engine and language translator will be obtained as purchased components for the new system
R5 : Their internal details are not part of this project
R6 : Issues of website security, other than password protection within the site, are not part of this project

Ambiguity Found : ['this', 'this', 'this', 'SIMULTANEOUS', 'this']
Lexical Count : 5
Total Requirement : 6
Lexical Ambiguity : 83.33333333333334

Semantic

Ambiguity Found : ['not', 'not', 'A', 'a', 'not', 'and']
Semantic Count : 6
Total Requirement : 6
Semantic Ambiguity : 100.0

Synctactic

Ambiguity Found : []
Synctactic Count : 0
Total Requirement : 6
Synctactic Ambiguity : 0.0

Referential

Ambiguity Found : ['this', 'this', 'this', 'this']
Referential Count : 4
Total Requirement : 6
Referential Ambiguity : 66.66666666666666

3.2 Requirement Ambiguity Assessment

The discussion in the above sections gives explanation for the importance of measuring requirement ambiguity at an early stage of software development. To measure the ambiguity well, **Requirement Ambiguity Assessment** is required. In this work various aspects of requirement ambiguity are studied and analyzed. After critical analysis of various factors, few attributes have been taken for analysis and a Requirement Ambiguity Assessment is proposed herewith using Adaptive Neuro-Fuzzy Approach with following four key attributes:

1. Lexical Ambiguity; 2. Syntactic or Structural Ambiguity; 3. Semantic Ambiguity; 4. Referential / Anaphoric Ambiguity

Ambiguity in requirements at the later stage can cause uncertainty in the development; it can also affect the quality of the software. The challenging issue is to deal with the ambiguity and to measure it. **Requirement Ambiguity Assessment** solves this problem by providing a quantitative result using the above mention attributes and implementing a fuzzy based system.

3.2.1 Adaptive Neuro Fuzzy Inference System (ANFIS)

It is a hybrid scheme which amalgamates the cloaked advantages of Artificial Neural Network and Fuzzy Logic (Adel Mellit et al., 2011).The scheme is already employed in a variety of forecasting and modeling problems efficiently (Melek Acar Boyacioglu, Derya Avci., 2010; B. Somayeh Mousavi, Alireza Hirad., 2011; Soumadip Ghosh, *2016*). ANFIS instigates with the fuzzification of input parameters. Definition of membership function and fuzzy rules are designed by effective use of the learning capability of Artificial Neural Network. The ANFIS system is trained through a dataset of many inputs to one output. After the proper training, the system operates perfectly as a fuzzy expert system (Vibha Gaur et al., 2014; Soo Ling Lim, Anthony Finkelstein, 2012).

ANFIS is a layered network with 5 layers that performs training in two different passes with some epochs. The node outputs, during each epoch, are calculated up to layer 4. Lastly at layer 5, single consolidated output value is aggregated through resulted outputs of various nodes and in order to get the assertion parameter the errors are propagated back throughout the layers. The Sugeno Fuzzy Inference System (SFIS) is widely employed in establishing a relationship between a series of input and output sets. The relationship is characterized by a linear equation known as **First-order Sugeno FIS**. The structure of the consequent parts of the first-order Sugeno FIS can be articulated by following linear function (C. D. Doan et al., 2005; A. M. Elwakdy, B. E. Elsehely., 2008).

y = fm(X1, X2 … … … . . Xk) where **m:** no of output membership functions; **k:** no of inputs

3.2.2 Proposed Algorithm: In this work an Adaptive Neuro Fuzzy Inference System (ANFIS) tool in MATLAB is used. The network is trained by Hybrid Approach for learning algorithm which is acombination of back propagation and least mean square algorithmThis sheme consists of four steps:

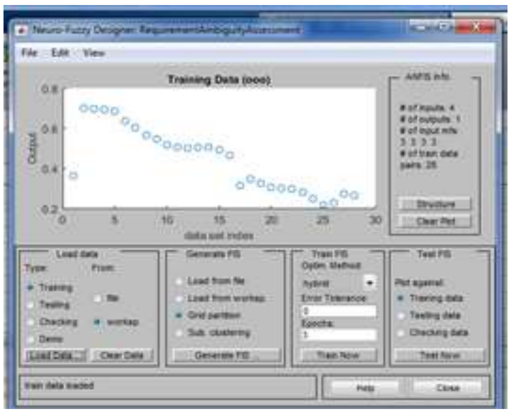
1) Loading of Training Data , 2) Generating ANFIS Model Structure , 3) Training of ANFIS

4) Requirement Ambiguity Assessment: View FIS Structure, Generated Rules, Output Surface of FIS

Step 1: Loading of Training Data: In this work PURE dataset is used with 42 modules. The dataset is divided into two parts: Training dataset and testing dataset. The training data is employed to train the system and the test data is used for its validation. The dataset used here for training includes 28 modules and testing dataset includes 14 modules. Training dataset was chosen randomly from 42 modules. The membership functions are trained using the same 28 modules (Figure 3).

Step 2: Generating Fuzzy Inference System

The ANFIS System proposed in this work is a first-order Sugeno FIS with one output and four inputs. Each input variable has a generalized bell type membership function whereas the output variable uses constant membership function. The Sugeno type FIS has been developed and for the training of this FIS, an Adaptive Neuro Fuzzy system (ANFIS) is designed that makes use of the Sugeno FIS Structure. (Figure 4)



ANFIS Info.
# of inputs: 4
# of outputs: 1
# of input mfs: 3 3 3 3
# of train data pairs: 28

Load data — Type: [✓ Training] Testing Checking Demo; From: file [✓ worksp.]
Generate FIS — Load from file; Load from worksp.; [✓ Grid partition]; Sub. clustering
Train FIS — Optim. Method: hybrid; Error Tolerance: 0; Epochs: 3
Test FIS — Plot against: [✓ Training data] Testing data Checking data

train data loaded

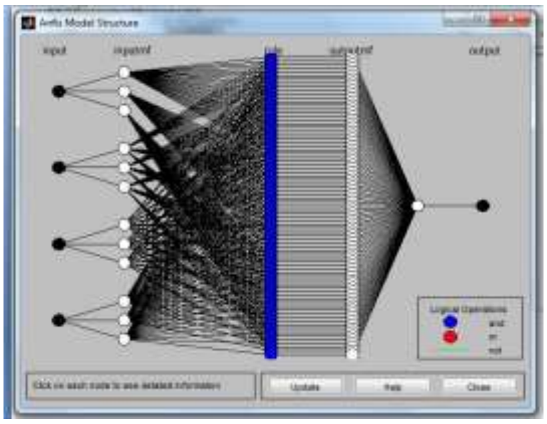
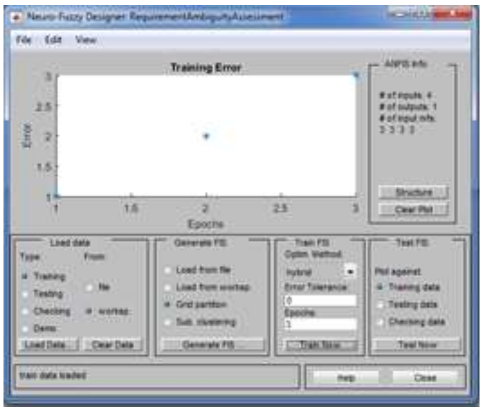


Figure 4: Loading of Training Data

Figure 5: Generating Sugeno based Fuzzy Inference system

Step 3: Training of ANFIS

The training of ANFIS is being performed by using Hybrid Algorithm. ANFIS applies subtractive clustering to cut training time which gets increased exponentially with the number of input variable. It scales back the input dimensions by accumulating extremely densed data into a number of small data clusters (T.L. Saaty, 1987; Carlson, 1996). This technique is advantageous when a huge data set is present and provides relatively good speed.



ANFIS Info.
# of inputs: 4
# of outputs: 1
# of input mfs: 3 3 3 3

Load data — Type: [✓ Training] Testing Checking Demo; From: file [✓ worksp.]
Generate FIS — Load from file; Load from worksp.; [✓ Grid partition]; Sub. clustering
Train FIS — Optim. Method: hybrid; Error Tolerance: 0; Epochs: 3
Test FIS — Plot against: [✓ Training data] Testing data Checking data

train data loaded

Figure 6: ANFIS Traing

Step 4: Ambiguity Assessment

- View FIS Structure

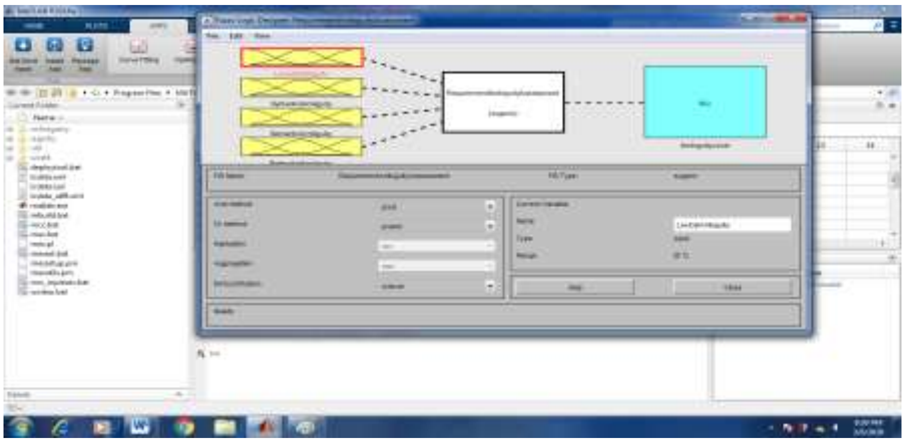
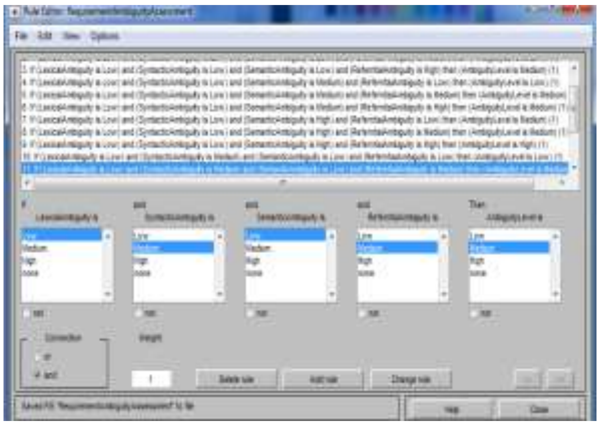


Figure 7: Generating FIS Structure

Generated Rules: The general structure of the rules can be expressed as following:

IF (X1 is M1) AND (X2 is N1) AND (X3 is O1) AND (X4 is P1) AND (X5 is Q1) THEN Y=f1(X1,X2.....X5) Whereas values of input membership functions M1,N1,O1,P1,Q1 are Guassian membership function.



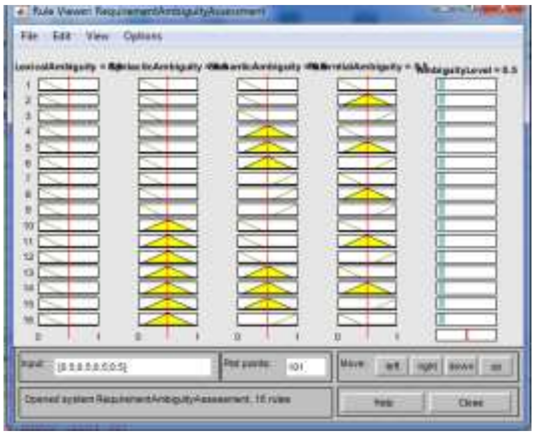


Figure 8: Rule Editor **Figure 9: Production Rule Generation**

- Output Surface of FIS

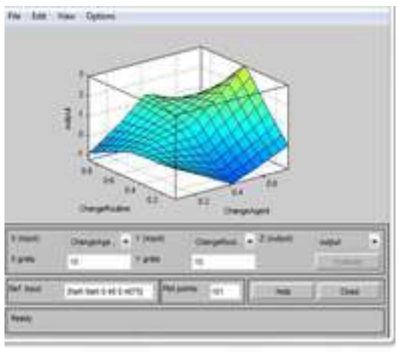


Figure 10: Surface View of ANFIS

3.2.3 Empirical Analysis

With the purpose of cramming the performance of ANFIS Model for **Requirement Ambiguity Assessment** proposed herewith, an experiment has been performed. The MATLAB software (Version 8.0.0.783) is used for the simulation of the proposed framework. The list of parameters used for simulation in MATLAB is shown in the Table 5 below.

Parameter Variables	**Associated Values**
Simulation Tool	MATLAB 8.0.0.783 (R2012b)
Dataset used for experimentation	PURE dataset
Total No. of projects	42
No. of projects used for training	28
No. of projects used for testing	14
FIS method	Grid Partitioning
Optimization method	Hybrid
No. of membership functions	4
Type of membership functions	Trimf, Trapmf, gbellmf, gaussmf, gauss2mf, pimf, dsigmf, psigmf.
No. of epochs	3

Table 5: List of Parameter Variables and Their Values

Obtaining Training and Test Data

The training and test data using domain knowledge was obtained as a preprocessing step of the frameworks. The decision rules were outlined in the following way:

"If (LexicalAmbigiuty is Low) and (SyntacticAmbiguity is Low) and (SemanticAmbiguity is Low) and (ReferentialAmbiguity is Low) then (Ambiguity is Low)".

Further rules were framed in analogous manner. Generated SFIS is simulated many times and the results produced are documented in a data set. The Surface plots have been attained to characterize the association among various input and output variables. Figure 10 reveals the correlation among the input variables:

Requirement Ambiguity Attributes				**Output**
Lexical Ambiguity	***Syntactic Ambiguity***	***Semantic Ambiguity***	***Referential Ambiguity***	***Ambiguity Level***
0.092	0.092	0.868	0.868	0.361
0.220	0.252	0.692	0.774	0.698
0.228	0.260	0.676	0.766	0.697
0.260	0.356	0.628	0.755	0.690
0.276	0.372	0.620	0.750	0.682
0.300	0.313	0.604	0.734	0.636
0.324	0.404	0.540	0.726	0.603
0.332	0.420	0.564	0.710	0.565
0.302	0.436	0.568	0.694	0.546
0.355	0.460	0.532	0.671	0.515
0.372	0.468	0.516	0.655	0.506
0.388	0.484	0.508	0.647	0.501
0.395	0.492	0.484	0.632	0.504
0.400	0.568	0.460	0.607	0.506
0.462	0.636	0.340	0.337	0.492

0.500	0.660	0.332	0.313	0.463
0.516	0.684	0.324	0.274	0.313
0.532	0.492	0.300	0.265	0.346
0.520	0.706	0.270	0.242	0.326
0.556	0.332	0.260	0.234	0.304
0.503	0.748	0.244	0.218	0.302
0.612	0.706	0.220	0.210	0.295
0.635	0.812	0.204	0.292	0.278
0.620	0.828	0.180	0.187	0.248
0.668	0.844	0.148	0.155	0.215
0.700	0.860	0.124	0.147	0.225
0.724	0.876	0.108	0.131	0.273
0.748	0.884	0.092	0.107	0.262

Table 6: PURE dataset

Comparative Analysis: Different membership functions available in ANFIS have been used to analyze Ambiguity and a comparative result has been shown in Table 7.

Lexical	Syntatic	Semantic	Referential	Tri mf	Trap mf	Gbell mf	Gauss mf	Gauss 2mf	Pi mf	Dsig mf	Psig mf
0.092	0.092	0.868	0.868	0.361	0.369	0.365	0.367	0.360	0.367	0.366	0.364
0.220	0.252	0.692	0.774	0.698	0.692	0.699	0.698	0.693	0.692	0.691	0.699
0.228	0.260	0.676	0.766	0.697	0.699	0.699	0.695	0.694	0.696	0.699	0.695
0.260	0.356	0.628	0.755	0.693	0.698	0.696	0.698	0.690	0.695	0.698	0.695
0.276	0.372	0.620	0.750	0.682	0.688	0.687	0.686	0.681	0.687	0.686	0.689
0.300	0.313	0.604	0.734	0.636	0.635	0.636	0.637	0.633	0.636	0.637	0.639

						8					
0.324	0.404	0.540	0.726	0.603	0.605	0.605	0.605	0.601	0.605	0.608	0.607
0.332	0.420	0.564	0.710	0.565	0.566	0.567	0.565	0.561	0.565	0.568	0.567
0.302	0.436	0.568	0.694	0.546	0.547	0.548	0.548	0.540	0.549	0.545	0.546
0.355	0.460	0.532	0.671	0.515	0.515	0.515	0.515	0.515	0.515	0.515	0.515
0.372	0.468	0.516	0.655	0.506	0.506	0.506	0.506	0.506	0.506	0.506	0.506
0.388	0.484	0.508	0.647	0.501	0.501	0.501	0.501	0.501	0.501	0.501	0.501
0.395	0.492	0.484	0.632	0.504	0.504	0.504	0.504	0.504	0.504	0.504	0.504
0.400	0.568	0.460	0.607	0.506	0.506	0.506	0.506	0.506	0.506	0.506	0.506
0.462	0.636	0.340	0.337	0.492	0.492	0.492	0.492	0.492	0.492	0.492	0.492
0.500	0.660	0.332	0.313	0.463	0.463	0.463	0.463	0.463	0.463	0.463	0.463
0.516	0.684	0.324	0.274	0.313	0.313	0.313	0.313	0.313	0.313	0.313	0.313
0.532	0.492	0.300	0.265	0.346	0.346	0.346	0.346	0.346	0.346	0.346	0.346
0.520	0.706	0.270	0.242	0.326	0.326	0.326	0.326	0.326	0.326	0.326	0.326
0.556	0.332	0.260	0.234	0.304	0.304	0.304	0.304	0.304	0.304	0.304	0.304
0.503	0.748	0.244	0.218	0.302	0.302	0.302	0.302	0.302	0.302	0.302	0.302
0.612	0.706	0.220	0.210	0.295	0.295	0.295	0.295	0.295	0.295	0.295	0.295
0.635	0.812	0.204	0.292	0.278	0.278	0.278	0.278	0.278	0.278	0.278	0.278
0.620	0.828	0.180	0.187	0.248	0.248	0.248	0.248	0.248	0.248	0.248	0.248
0.668	0.844	0.148	0.155	0.215	0.215	0.215	0.215	0.215	0.215	0.215	0.215
0.700	0.860	0.124	0.147	0.225	0.225	0.225	0.225	0.225	0.225	0.225	0.225
0.724	0.876	0.108	0.131	0.273	0.273	0.273	0.273	0.273	0.273	0.273	0.273
0.748	0.884	0.092	0.107	0.262	0.262	0.262	0.262	0.262	0.262	0.262	0.262

Table 7 : Computed Ambiguity Level For PURE Projects-Training Case – ANFIS Functions

This work provides an approach for assessing requirement ambiguity at an early stage of software development. It is a hybrid scheme which combines the potential benefits of FIS and ANN. It is equipped with the potential to produce a Fuzzy Inference System along with linear relationship between input- output data that helps in analytical inference of fuzzy data. It has been observed by the empirical analysis that the proposed approach is capable of analyzing the ambiguous requirements at an earlier stage of software development.

3.3 Relation between Uncertainty and Ambiguity

When compiling the literal meaning of Ambiguity in several dictionaries (Oxford English Dictionary, Cambridge, Webster, Collins, the following clusters of meanings appeared :

1. the possibility of interpreting an expression in two or more distinct ways
2. vagueness or uncertainty of meaning
3. uncertainty
4. a situation or statement that is unclear because it can be understood in more than one way
5. the state of being uncertain
6. uncertainty or dubiousness

Ambiguity is sometimes described as ‘second order uncertainty’, where there is uncertainty even about the definitions of uncertain states or outcomes. The difference here is that this ‘second order uncertainty’ is about the human definitions and concepts, not an objective fact of nature. Mathematically, ambiguity is defined as the state of being skeptical of the type of statistical distributions related to an uncertainty.

At times uncertainties emerge, no matter how much information is present. These are situations where it is not possible to predict what roles are being played by the participating entities. This type of uncertainty results in ambiguity (Wideman, R. M., 1992).

One approach of resolving requirement ambiguity and requirement uncertainty in the requirements phase is to capture the ambiguities and then reduce it into uncertainty. In this paper an attempt has been made to first identify the potential ambiguities present in the requirements document and then further give guidelines to remove these uncertainties resulting out of the ambiguous or vague requirements.

Accomplishment of any software project is directly proportional to the quality of its requirements in terms of its functionality and monetarily. Ambiguities in requirements during development life cycle extensively contribute to the quality of requirement specification. The requirements cannot be treated independently as they are related and influence others in compound manners. Ambiguous requirements may be interpreted differently; the remaining requirements may be effected as a result of this interpretation. This may certainly cause Requirement Uncertainty. Hence, Uncertainty in requirements due to ambiguous requirements must be examined by analyzing interdependencies among the requirements. Dependencies among requirements are very vital to determine as it can be very useful to understand the behavior of requirements in presence of ambiguities.

3.3.1 Requirement Uncertainty by analyzing Requirement Dependency

The Requirements detected as ambiguous through the implementation of above mentioned process are collected into a Requirement Repository R. The Requirement Repository R can be defined as a set of requirements such that R=$\sum_i^n$ (R_i, $R_{j...}$ R_n). It contains the requirements along with its definition and attributes, including dependency information. The dependency information describes the relationship among requirements that is useful to determine how a requirement influence and probably impacts other requirements. A requirement R_i can be described by the following characteristic set:

{Requirement_ID, Dependency_Queue, Ambiguity_Level, Dependency_Degree, Uncertainty_Index}
Where

Requirement_ID	It is the unique identifier for each requirement.
Dependency_Queue	It determines the requirements R_i is depending upon.
Ambiguity_Level	It determines the level upto which a requirement is ambiguous as identified by the Requirement Ambiguity Detection Tool.
Dependency_Degree	It describes the degree up to which a requirement is dependent on others.
Uncertainty_Index	The no of elements in Dependency_Queue and Dependency_Degree determines uncertainty in the requirements at any time instance.

Table 8: Requirement Attributes

To begin with, a directed graph will be embedded for the requirements stored in requirement repository considering requirements as vertices and their relationship as edges. Then Adjacency matrices X, Y will be populated at time t, t+d for the graph (d is a constant) respectively. Comparison of Matrices will provide the change in dependencies due to presence of ambiguous requirements and degree of uncertainty.

Step1: Embedding Requirement Dependency Graph

It is a directed graph in which requirements are symbolized as vertices and their relationship as edges. If requirement (R_i) is depending on requirement (R_j), corresponding edge $\mathbf{Ri \rightarrow R_j}$ must be drawn in the graph (Figure 11).

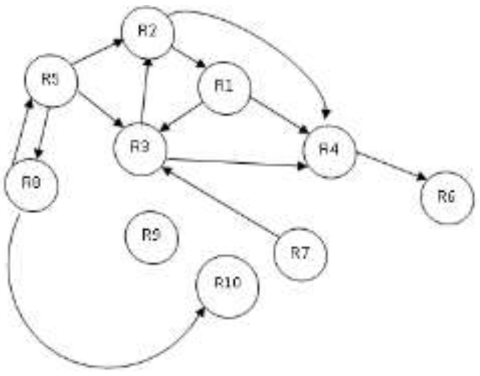


Figure 11: Requirement Dependency Graph1 (RDG1)

Step 2: Generate Requirement Dependency Matrix

It is an adjacency matrix of N*N where requirements are infiltrated as input to rows and columns. It has three values {0, 1 and -1}.This matrix provides Dependency Degree as output (Figure 12).

$$
\begin{array}{c|cccccccccc}
 & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & \mathbf{9} & \mathbf{10} \\
\mathbf{1} & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{2} & 1 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{3} & -1 & 1 & 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 \\
\mathbf{4} & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{5} & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\
\mathbf{6} & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{7} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

$$
\begin{array}{c|cccccccccc|}
\mathbf{8} & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
\mathbf{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
\end{array}
$$

Figure 12: Requirement Dependency Matrix (RDM1) at time t

It is an iterative process. Due to presence of ambiguous requirement, the requirements may keep on changing with their graph and matrix. Generation of graphs and matrices at regular time (Figure 13) makes it adaptive in nature.

$$
\begin{array}{c|cccccccccc|}
 & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & \mathbf{9} & \mathbf{10} \\
\mathbf{1} & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\mathbf{3} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
\mathbf{4} & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
\mathbf{5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\mathbf{7} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{8} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
$$

$$\begin{array}{c|cccccccccc|}
9 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{array}$$

Figure 13: Requirement Dependency Matrix (DM1) at time t+d

Comparison of matrices generated at different times is providing the relationship among various requirements. Also the uncertainty of requirements is being determined by calculating Uncertainty Index. Triangular Distribution function is used to compute **Uncertainty_Index.** It is a Continuous probability distribution with a probability density function formed like a triangle.

Uncertainty_Index(Uncertainty_Index|Ambiguity_Level|Depdency_Degree) =

TriangularDist (Uncertainty_Index, Ambiguity_Level, Dependency_Degree).

This index can be used as Uncertainty Threshold which is very useful to deal with the requirement ambiguity and uncertainty.

IV. Conclusion and Future Work

Ambiguity and Uncertainty in requirement specification always results in software failure, high development cost, delays in schedule, due to rework on requirements. This paper attempts to build a Management Model for Requirement Ambiguity and Uncertainty in Software Requirement Specification (RAUMM). The work is carried out in three stages, Requirement Ambiguity Detection, Requirement Ambiguity Assessment and Analysis of Ambiguity and Uncertainty. The tool identifies potential ambiguous requirements (Lexical, Syntactic, Semantic and Referential). The Future Work involves training the dataset to keep on adding potential ambiguities arising out of SRS from the PURE and few more datasets. The frequencies of mostly used ambiguous words can also be used in the model to completely avoid them. Also the tool should be modified to incorporate pragmatic and other ambiguity types.

References:

1. M. Elwakdy, B. E. Elsehely, 2008. Speech Recognition using A Wavelet Transform to Establish Fuzzy Inference System through Subtractive Clustering and Neural Network (ANFIS). International Journal of Circuits, Systems and Signal Processing, 2(4), 381-386.
2. Additional Standish Group report, Retrieved from https://www.opendoorerp.com/the-standish-group-report-83-9-of-it-projects-partially-or-completely-fail/ last updated on Feb 20, 2019.
3. Adel Mellit, Soteris A. Kalogirou., 2011. ANFIS-based Modelling for Photovoltaic Power Supply System: A Case Study. Renewable Energy, 36(1), 250–258, Doi: 10.1016/j.renene.2010.06.028.

4. Alessio Ferrari, Giorgio Oronzo Spagnolo, Stefania Gnesi, 2017. PURE: a Dataset of Public Requirements Documents, IEEE 25th International Requirements Engineering Conference, Doi: 10.1109/RE.2017.29.

5. Alessio Ferrari, Giuseppe Lipari, Stefania Gnesi, Giorgio O. Spagnolo, 2014. Pragmatic Ambiguity Detection in Natural Language Requirements, 978-1-4799-6355-3/14, IEEE.

6. Anda B, Sjøberg DIK. Towards an Inspection Technique for Use Cases Models, 2002. Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (1325):127-34.

7. Somayeh Mousavi, Alireza Hirad, 2011. Automatic Gender Classification using Neuro Fuzzy System. Indian Journal of Science and Technology, 4(10). Doi: 10.17485/ ijst/2011/v4i10/30158.

8. Bajwa I.S., Lee M., Bordbar B., 2012. Resolving Syntactic Ambiguities in Natural Language Specification of Constraints. In: Gelbukh A. (eds) Computational Linguistics and Intelligent Text Processing. CICLing 2012. Lecture Notes in Computer Science, Vol 7181. Springer, Berlin, Heidelberg.

9. Beg, R., Q. Abbas, and A. Joshi, 2008. A Method to Deal with the Type of Lexical Ambiguity in a Software Requirement Specification Document. First International Conference on Emerging Trends in Engineering and Technology,ICETET'08.

10. Berry D.M., Kamsties E. , 2004. Ambiguity in Requirements Specification. In: do Prado Leite J.C.S., Doorn J.H. (eds) Perspectives on Software Requirements. The Springer International Series in Engineering and Computer Science, vol 753. Springer, Boston, MA.

11. Berry, D.M., Kamsties, E., Krieger, M.M., 2003. From Contract Drafting To Software Specification: Linguistic Sources of Ambiguity. Univ. of Waterloo Technical Report, http://se.uwaterloo.ca/˜dberry/handbook/ambiguityHandbook.pdf (accessed 27.12.2009).

12. Brown PF, DeSouza PV, Mercer RL, Della Pietra VJ, Lai JC, 1992. Class-Based n-gram Models of Natural Language. Computational Linguistics.; 18(1950) pp:467-79.

13. D. Doan, S. Y. Liong, Dulakshi S. K. Karunasinghe, 2005. Derivation of Effective and Efficient Data Set with Subtractive Clustering Method and Genetic Algorithm, Journal of Hydroinformatics, 7 (4), 219-233. Retrieved from http://jh.iwaponline.com/content/7/4/219.

14. Havasi, R. Speer, and J. Alonso, 2007. ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge, Recent Advances in Natural Language Processing, Borovets, Bulgaria, [Online]. Available: http://web.media.mit.edu/jalonso/cnet3.pdf

15. Ark A, Giorgolo G, Lappin S., 2013. Statistical Representation of Grammaticality Judgments: the Limits of N-Gram Models. Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, pp: 28-36.

16. Cycorp Inc., “ResearchCyc.” [Online]. Available: http://research.cyc.com/ accessed 2015.

17. Denger C, Berry DM, 2003. Higher Quality Requirements Specifications through Natural Language Patterns. IEEE International Conference on Software-Science, Technology & Engineering (SwSTE’03) 0-7695-2047-2/03, pp :1-11.

18. Lutters, F. Van Houten., 2013. Ambiguity and Uncertainty of Requirements in Product Development. International Conference on Competitive Manufacturing COMA' 13.

19. M. Suchanek, G. Kasneci, and G. Weikum, 2007. Yago: A Core of Semantic Knowledge. [Online] Available: http://suchanek.name/work/publications/www2007.pdf.

20. Fabian de Bruijn, Hans L. Dekkers, 2010. Ambiguity in Natural Language Software Requirements: A Case Study pp. 233–247, Springer-Verlag Berlin Heidelberg 2010.

21. Femmer, H., Fernandez, D.M., Juergens, E., 2014. Rapid requirements checks with requirements smells: two case studies. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering.. ACM.

22. A. Miller, C. Fellbaum, R. Tengi, P. Wakefield, H. Langone, B. R. Haskell, 2006. WordNet: A Lexical Database for English Language. [Online] Available: http://wordnet.princeton.edu/.

23. Gill, K.D., Raza, A., Zaidi, A.M., Kiani, M.M., 2014. Semi-Automation for Ambiguity Resolution in Open Source Software requirements. IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE), pp: 1-6.

24. Gleich B., Creighton O., Kof L., 2010. Ambiguity Detection: Towards a Tool Explaining Ambiguity Sources. In: Wieringa R., Persson A. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2010. Lecture Notes in Computer Science, vol 6182. Springer, Berlin, Heidelberg.

25. Halima Sadia, Syed Qamar Abbas, 2019. A Systematic Literature Review Of Multi-Criteria Risk Factors (VUCA) In Requirement Engineering. International Journal Of Scientific & Technology Research Volume 8, Issue 11, November 2019 ISSN 2277-8616.

26. Havasi C, Speer R, Alonso JB, 2007. ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge. Proceedings of Recent Advances in Natural Languages Processing, pp: 1-7.

27. Emily Hill, Zachary P. Fry, Haley Boyd, Giriprasad Sridhara, Yana Novikova, Lori Pollock, K. Vijay-Shanker, 2008. AMAP: Automatically mining abbreviation expansions in programs to enhance software maintenance tools. Proceedings 5th Working Conference on Mining Software Repositories, pp 79-88.

28. Hussain I, Ormandjieva O, Kosseim L., 2007. Automatic Quality Assessment of SRS Text By Means of A Decision-Tree-Based Text Classifier. International Conference on Quality Software. (Qsic): 209-18. Doi: 10.14419/ijet.v7i2.29.13808.

29. IEEE, 1993. IEEE Recommended Practice for Software Requirements Specifications. ANSI /IEEE Standard 830-1993. New York, NY: Institute of Electrical and Electronics Engineering.

30. J. J. Romano, J. D. Palmer, 1998. TBRIM: Decision Support for Validation/Verification of Requirements, SMC'98 Conference Proceedings. IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218), San Diego, CA, USA, 1998, pp. 2489-2494 vol.3.

31. Lawrence R., Jauch Kenneth L. Kraft, 1986. Strategic Management of Uncertainty. Academy of Management Review, pp 777–790. https://doi.org/10.5465/amr.1986.4283934.

32. Berry D.M., Kamsties E., 2004. Ambiguity in Requirements Specification. In: do Prado Leite J.C.S., Doorn J.H. (eds) Perspectives on Software Requirements. The Springer International Series in Engineering and Computer Science, vol 753. Springer, Boston, MA, https://doi.org/10.1007/978-1-4615-0465-8_2.

33. Karlsson, 1996. Software Requirements Prioritizing. Proceedings of the 2nd International Conference on Requirements Engineering (ICRE '96), IEEE Computer Society, pp. 110 –116. ISBN: 0-8186-7252-8.

34. Khin Hayman Oo, Azlin Nordin, Amelia Ritahani Ismail, Suriani Sulaiman, 2018. An Analysis of Ambiguity Detection Techniques for Software Requirements Specification (SRS). International Journal of Engineering & Technology, 7 (2.29) pp: 501-505.

35. Korner, S.J. and T. Brumm, 2009. RESI-A Natural Language Specification Improver. International Conference on Semantic Computing. ICSC'09. IEEE, Doi: 10.1109/ICSC.2009.47

36. Melek Acar Boyacioglu, Derya Avci, 2010. An Adaptive Network-Based Fuzzy Inference System (ANFIS) for the Prediction of Stock Market Return: The Case of he Istanbul Stock Exchange. Expert Systems with Applications. 37(12), 7908–7912. Doi:10.1016/j.eswa.2010.04.045.

37. Nidumolu, S., 1996. Standardization, Requirements Uncertainty and Software Project Performance, Information and Management, 31, pp: 135-150.

38. Polpinij J, Ghose A., 2008. An Automatic Elaborate Requirement Specification by Using Hierarchical Text Classification. International Conference on Computer Science and Software Engineering, CSSE. pp: 706-9.

39. Polpinij J., 2009. An Ontology-Based Text Processing Approach for Simplifying Ambiguity of Requirement Specifications. IEEE Asia-Pacific Services Computing Conference, APSCC 2009. pp: 219-26.

40. Popescu D., Rugaber S., Medvidovic N., Berry D.M., 2008. Reducing Ambiguities in Requirements Specifications Via Automatically Created Object-Oriented Models. In: Paech B., Martell C. (eds) Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs. Monterey Workshop 2007. Lecture Notes in Computer Science, vol 5320. Springer, https://doi.org/10.1007/978-3-540-89778-1_10.

41. Popescu D., Rugaber S., Medvidovic N., Berry D.M., 2008. Reducing Ambiguities in Requirements Specifications Via Automatically Created Object-Oriented Models. In: Paech B., Martell C. (eds) Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs. Monterey Workshop 2007. Lecture Notes in Computer Science, vol 5320. Springer, https://doi.org/10.1007/978-3-540-89778-1_10.

42. Sandhu, G. and S. Sikka, 2015. State-of-Art Practices to Detect Inconsistencies and Ambiguities from Software Requirements. International Conference on Computing, Communication & Automation (ICCCA), IEEE.

43. Seijas L, Segura E., 2009. Detection of Ambiguous Patterns using SVMS: Application to Handwritten Numeral Recognition. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) LNCS, pp: 840-7.

44. Sharma R, Bhatia J, Biswas KK, 2014. Machine Learning for Constituency Test of Coordinating Conjunctions in Requirements Specifications. Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering - RAISE 2014. pp: 25-31.

45. Soo Ling Lim & Anthony Finkelstein, 2012. StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation. IEEE Transactions on Software Engineering. 38 (3),707–735. Doi: 10.1109/TSE.2011.36.

46. Soumadip Ghosh, Sushanta Biswas, Debasree Chanda and Partha Pratim Sarkar., 2016. Breast Cancer Detection using a Neuro-fuzzy based Classification Method, Indian Journal of Science and Technology, 9(14), Doi: 10.17485/ijst/2016/v9i14/76802.

47. Stiehm, J. H., Townsend, N.W., 2002, The U.S. Army War College: Military Education in a Democracy, Temple University Press.

48. T.L. Saaty. (1987).The Analytic Hierarchy Process: What it is and How it is Used. Mathematical Modelling, 9(3–5), pp: 161-176. Doi: 10.1016/0270-0255(87)90473-8.

49. The Ambiguity Review Process, 2003. Richard Bender. R Bender - Bender RBT Inc, 2003 - benderrbt.com.

50. The Chaos Report, 2009. Standish Group, Retrieved from http://www.standishgroup.com/sampleresearch/ choas2009.pdf accessed on 20 October 2014.

51. Umber A., Bajwa I.S., Asif Naeem M., 2011. NL-Based Automated Software Requirements Elicitation and Specification. In: Abraham A., Lloret Mauri J., Buford J.F., Suzuki J., Thampi S.M. (eds) Advances in Computing and Communications. ACC 2011. Communications in Computer and Information Science, vol 191. Springer, https://doi.org/10.1007/978-3-642-22714-1_4.

52. Vibha Gaur, Anuja Soni, Punam Bedi and S.K. Muttoo, 2014. Comparative Analysis of ANFIS and ANN for Evaluating Inter-Agent Dependency Requirements. International Journal of Computer Information Systems and Industrial Management Applications, 6, pp: 23–34. Retrieved from www.mirlabs.net/ijcisim/index.html.

53. Wang X-Z, Dong L-C, Yan J-H., 2012. Maximum Ambiguity-Based Sample Selection in Fuzzy Decision Tree Induction. IEEE Transactions on Knowledge and Data Engineering. 24(8), pp: 1491-505.

54. Wideman, R. M., 1992. Project and Program Risk Management: A Guide to Managing Project Risks and Opportunities. Project Management Institute.

55. Wijewickrema CM, 2014. Impact of an Ontology for Automatic Text Classification. Annals of Library and Information Studies; 61(4), pp: 263-72.

56. Versley, Y., 2008. Vagueness and Referential Ambiguity in a Large-Scale Annotated Corpus. Research on Language and Computation, 6, pp: 333–353. https://doi.org/10.1007/s11168-008-9059-1.