ISSN: 0976-3031

Available Online at http://www.recentscientific.com

CODEN: IJRSFP (USA)

International Journal of Recent Scientific Research
Vol. 8, Issue, 8, pp. 19461-19466, August, 2017

International Journal of Recent Scientific Research

DOI: 10.24327/IJRSR

Research Article

INCONSISTENCY CHECKING IN REQUIREMENTS BY IMPROVED SEMANTIC REASONING (ICRISR)

Yesudoss J* and Ramani A.V

Department of Computer Science Sri Ramakrishna Mission Vidyalaya College of Arts and Science Coimbatore-641 020, India

DOI: http://dx.doi.org/10.24327/ijrsr.2017.0808.0698

ARTICLE INFO

Article History:

Received 06th April, 2017
Received in revised form 14th June, 2017
Accepted 23rd July, 2017
Published online 28th August, 2017

Key Words:

Inconsistency, Semantic Reasoning, Natural Language Processing.

ABSTRACT

Requirements Engineering (RE) is the process of understanding the customer expectations about the system to be developed, and to document them in a easily readable and understandable format, which will serve as reference for the subsequent design, implementation and verification of the system. RE is the first segment in the software development life cycle, concerning about the requirements of stakeholders within the software system being developed. In the field of requirements engineering, measuring inconsistency is crucial to effective inconsistency management. A practical measure must consider both the degree and significance of inconsistency in specification. In existing sematic reasoning approach for checking inconsistencies in requirement documents, an antonym dictionary is generated for available terms in documents to check semantically contrasting terms exist. However matching antonyms alone failed to increase the maturity level of implementation. In this paper, an improved semantic resoning approach is proposed by capturing the semantics of natural language for better understanding the meaning of sentences like Synonyms, Hypernyms, Hyponyms and Acronyms dictionaries of words to check semantically similar terms exist.

INTRODUCTION

It is miles extensively recognized that inconsistency is unavoidable for the duration of the requirements stage, although maximum existing software development techniques or equipment assume consistency [1-3]. A practical way of dealing with inconsistency is getting to know to stay with inconsistency as opposed to parry it [3]. Moreover, in many instances, it can be perfect to take the initiative in dealing with inconsistency to facilitate the requirement's improvement and its control [2]. Inconsistencies could be viewed as signals of problematical statistics about necessities.

Measuring inconsistency is crucial for powerful inconsistency management [2, 1]. In general, clients and developers want to realize the wide variety and severity of inconsistencies in their necessities specs. Regularly, developers want to use these measures to prioritize inconsistencies on the way to discover inconsistencies that require pressing attentions, and to assess the development after inconsistency handling. In other phrases, the builders want to realize if a hard and fast of necessity statements come to be more or less "consistent" after a specific inconsistency handling movement has been taken.

It can not always surprise that strategies for measuring inconsistent specifications in classical logic are attractive [4]. Abstract identification is the manner of analyzing and extracting the important key terms, which is meant to indicate the concept of the particular document. Abstract terms of particular documents play an important role in assisting the software developer for an efficient software development in a particular period and without errors. The automated abstraction identification to extract abstract terms referred to as Ontological based relevance abstraction identification [5]. In practical inconsistency-handling, clients and developers need to recognize each the importance and severity of inconsistency. The relative importance of a requirement's statement constantly affects the evaluation of importance of an inconsistent specification. Consequently, valuable of measuring inconsistent specifications is the need to take the relative significance of requirements statements into consideration.

Semantic Reasoning approach is used to maintain semantic consistency and attain brief relation among words, extra to natural lexical parsing and additionally adopt over in the natural language. The semantic reasoning mentioned right here

Corresponding author:* **Yesudoss J
Department of Computer Science Sri Ramakrishna Mission Vidyalaya College of Arts and Science
Coimbatore-641 020, India

is to extract the relation between adjectives and adverbs. It checks consistency, abstraction, and terms of the documents are matched. Then the semantic reasoning approach only considers antonym of terms in documents with abstraction terms [6]. But in this paper, we proposed that Synonym, Hypernym/ Hyponym and Acronym of terms in documents are considered. Then this method is called improved semantic reasoning.

Related Work

Measuring inconsistent specification in terms of the priority based on scoring vector, which integrates the measure of the degree of inconsistency with the measure of the significance of inconsistency [7]. To apply model composition to cope with this problem in a staged approach. First, heterogeneous necessities are translated in model fragments that are instances of a common meta model. Then, those fragments are merged in one specific model. On this kind of version inconsistencies together with underneath-specifications can be incrementally detected and formal analysis is made feasible. It is fully supported by way of model composition framework [8]. ConsVIS or was tool for consistency checking of ontologies. This tool is a consistency checker for formal ontologies, consisting of both traditional data modeling languages and the latest ontology languages. ConsVISor checks consistency with the aid of verifying axioms [9].

A characterization of inconsistency in software program improvement and a framework for managing in this context. It attracts upon realistic reports of dealing with inconsistency in big-scale software program development projects and relates a few lessons learned from these experience [2] and then try to bridge the gap between early requirements specification and formal techniques. A new specification language, called Formal Tropos was endorsed that is based on the primitive standards of early requirements frameworks (actor, aim, strategic dependency). However, supplements them with a rich temporal specification language [10]. In standard, each tool concentrates on one unique type of description and defines consistency narrowly in terms of integrity policies for that description type. Such technique-particular consistency checking is extraordinarily beneficial, but covers best a fragment of the variety of consistency relationships which can have an effect on software development [11].

A systematic approach for identifying those identification of syntactic aliasing involves automated generation of patterns for identifying syntactic variances of terms, including abbreviations and introduced-aliases [12]. A measure-driven logic framework for handling non-canonical necessities. The framework includes five principal parts, figuring out non-canonical requirements, measuring them, generating candidate proposals for managing them, selecting typically suited proposals, and revising them in line with the chosen proposals. This generalization may be taken into consideration as an attempt to cope with non-canonical requirements together with logic-based inconsistency dealing with in requirements engineering [13]. Develop two evaluation mechanisms to detect two forms of modelling errors. The first mechanism worries the detection of inconsistent specification of contexts in a goal model. The second one worries the detection of conflicting context modifications that get up resulting from the moves accomplished through the machine to fulfill one of a kind requirements simultaneously [14]. A distance-based totally paraconsistent semantics for DL-Lite wherein meaningful conclusions may be rationally drawn even from an inconsistent knowledge base and a distance-based inconsistency measurement was increased for DL-Lite to provide greater informative metrics that can tell the variations among axioms causing inconsistency and amongst inconsistent knowledge [15].

Improved Semantic Reasoning

A specification here is a set of sentences from the requirement documents. For every sentence, first it is parsed by a natural language parser to extract all grammatical ingredients. Then in keeping with the dependency relation extracted by means of the parser, the translator decomposes the sentence into clauses recursively to isolate the impartial temporal units. After the decomposition, a syntax tree was constructed from the elements of the sentence, to extract atomic propositions, and to infer the temporal relationship of individual temporal elements according to the subordinators and modifiers. Typically an atomic proposition comes from a subject and its predicate extracted by using the dependency relation from the parser, i.e., within the shape of predicate challenge, to mix a variable and its valuation. For multiple subjects related with conjunctions, they may be decomposed to generate specific atomic propositions and then linked via the corresponding logic operators. [6]

To maintain semantic consistency and achieve clear temporal formulas, and more to natural lexical parsing, semantic reasoning is additionally adopted over in the natural language. The semantic reasoning mentioned right here is to extract the relation between adjectives and adverbs respectively according to their meaning in a specification, instead of semantic roles labeled by way of a parser. [6]

The proposed approach is the improved semantic reasoning mentioned to extract the words (such as synonym, hypernym/ hyponym and acronym) according to meaning in a specification. More exactly, after extracting the words used in a specification, then it is applied in pairs of semantically contrasting phrases by looking up a corresponding dictionary specified by means of users. With these semantically associated phrases, the variety of atomic propositions is decreased, to use inside the generated formulas, and avoid including the assumptions on the mutual unique propositions.

The antonyms, synonyms, hyponyms and acronyms are in online for each of the given word, which is straightforward to be carried out but time consuming. There are two-steps extraction procedure to compute the pairs of antonyms, synonyms, hyponyms and acronyms in a specification. Pairs of antonyms usually come from the sentences with equal topics. Therefore, at first the antonym, synonym, hyponym, and acronym candidates are organized related to the same subjects according to their dependency relation (subject, dependent) extracted by the natural language parser, where every word in the dependent set is initialized with color green. If the number of words in the dependent set for a subject is larger than one, an antonym, synonym, hypernym, hyponym and acronym dictionary is used to check whether semantically contrasting

words exist in the same set, otherwise continue to deal with other groups. The reason is that it cannot be used the derived antonyms, synonym, hypernym, hyponym and acronyms for the corresponding proposition reduction.

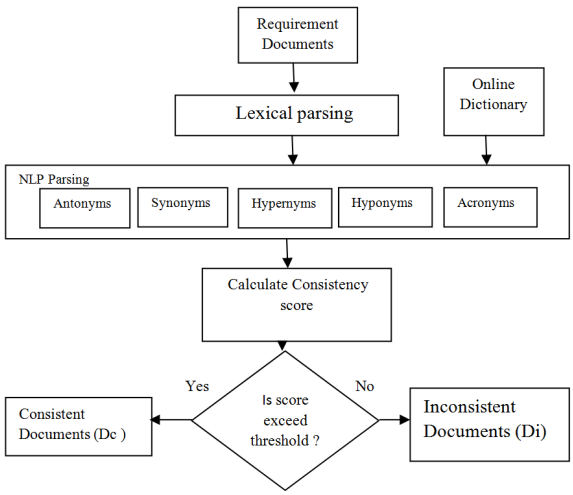

Figure 1 Overall Work Flow of Improved consistency Checking In Requirement Documents

The overall work flow of improved semantic reasoning structure is represented in Fig.1. The collected set of specific Requirement documents are parsed into lexical parser. The collected Requirement documents are the output of identified documents through "Ontological based relevance abstraction identification" technique [5]. This method was used to extract the specific set of domain related documents with the help of among the huge number of existing requirement documents for a particular requirement document of stack holder. This selected set of documents or any other type of set of relevant documents can be given as an input to this work. The lexical parser used here is tree tagger which convert a sentence into a sequence of tokens as subject, verb, noun, adverb, adjectives and prepositions for all documents. The antonym using online dictionary is extracted for verb, adverb and adjectives of each subject to find inconsistence sentences available in the documents. In the same way find synonym, hypernym, hyponym and acronym terms of the sentences using online dictionary to check semantically similar sentences available in documents. The documents are ranked based on number of consistence sentences availability.

Algorithm: Improved Semantic reasoning

input: Requirement documents (Rd), Specification S for subjects

Output: pairs of antonyms in the specification, pairs of synonyms in the specification, pairs of hyponyms in the specification, pairs of acronyms in the specification.

begin

initialize wordset = Ø ; s.dep= Ø
set color (s.dep) as green
for ∀ d ∈ Rd
subject = extract(S, wordset)
for ∀ s ∈ subject do
{
if |s.dep| ≥ 1 then
{
for ∀ w ∈ s.dep do
if w.color = = green then
{
if wordset(w).antonym == Ø then
{
wordset(w).antonym ← online(w)
antonmy = s.dep ⋂ wordset(w).antonym
}
if wordset(w). synonym == Ø then
{
wordset(w). synonym ← online(w)
synonmy = s.dep ⋂ wordset(w).synonym
}
if wordset(w). hypernym == Ø then
{
wordset(w). hypernym ← online(w)
hypernymy = s.dep ⋂ wordset(w).hypernym
}
if wordset(w). hyponym == Ø then
{
wordset(w). hyponym ← online(w)
hyponmy = s.dep ⋂ wordset(w).hyponym
}
if wordset(w). acronym == Ø then
{
wordset(w). acronym ← online(w)
acronmy = s.dep ⋂ wordset(w). acronym
}
if antonmy ≠ Ø then
{
w.color = blue
for w′ ∈ antonym do
w′.color = blue
wordset(w′).antonym = wordset(w′).antonym ∪ {w}
}
if synonmy ≠ Ø then
{
w.color = blue
for w′ ∈ synonym do
w′.color = blue
wordset(w′).synonym = wordset(w′).synonym ∪ {w}
}
if hypernym ≠ Ø then
{
w.color = blue
for w′ ∈ hypernym do
w′.color = blue
wordset(w′).hypernym = wordset(w′).hypernym ∪ {w}
}
if hyponymy ≠ Ø then
{
w.color = blue
for w′ ∈ hyponym do
w′.color = blue
wordset(w′).hyponym =
wordset(w′).hyponym ∪ {w}
}
if acronmy ≠ Ø then
{
w.color = blue

```
            for w′ ∈ acronym do
            w′.color = blue
          wordset(w′).acronym =wordset(w′).acronym ∪ {w}
            }
      }
    }
 for each d  ϵ documents
 for each subject
 for each w′.term  ϵ w′
 if  wordset(w′).antonym ϵ d
 d.score =d.score-1
 else if  wordset(w′). synonym ϵ d
 d.score =d.score+1
 else if  wordset(w′). hypernym ϵ d
 d.score =d.score+1
else if  wordset(w′). hyponym ϵ d
 d.score =d.score+1
 else if  wordset(w′).acronym ϵ d
 d.score =d.score+1
 }
  return (documents, score)
```

The "subject" is grouped elements depending on same subjects, and "wordset" is the stored set of antonym, synonym, hyponym, and acronym candidates with the extracted antonyms, synonyms, hyernyms, hyponyms and acronyms.
Mark the status of dependent words(s.dep) in a subject during improved semantic reasoning, with use of two colors, "green" color for all the words in the dependent set of the subject with comparing the dictionary, and "blue" color stands for the existence of antonyms, synonyms, hyponyms, and acronyms in the set.

These colors are indicators for our proposition reduction in the transformation process. That is, a word marked with green will be directly converted into atomic propositions with the corresponding subject. The items in the dependent sets of subjects (s.dep) are initialized with green. Then, it is assumed that every given word candidate (antonmy, acronmy, hyponmy and synonomy) can find its words from the corresponding dictionary. At first wordset(w) is initialized and the dependency relation from the specification is extracted. In the "extract" function, the extracted words candidates are saved in wordset, wherein first of all the sets of their words are empty.
Then, for every extracted word "w" in subject "s", if it weren't analyzed before, these words are looked from the corresponding dictionary through online and the end result is uploaded to the word set of w in wordset.

Further, the word (w) is checked for antonym, and if not automatically goes to another part to synonym, hypernym, hyponym and acronym, it will be processed. If antonyms, acronyms hyponyms, hypernym and synonyms are found for wordset (w′) in s.dep, it is marked in blue color. The subject and extracted wordset are returned to check consistency of document. The semantic reasoning is improved with the use of acronym, hyponym, hypernym and synonym.

After collecting wordset for each subject is cheked in requirement documents. If terms available in document, the document score is incremented. Finally top k documents are selected as consistent documents, remaining are considered as inconsistent documents.

RESULT AND DISCUSSION

The comparison is made in terms of the performance metrics referred to as the precision accuracy and recall that are defined certain within the following subsections.

Precision

Precision is defined as the Percentage of correct predicted results from the set of input terms. The precision value should be more on the proposed methodology than the existing approach for the better system performance.

Precision is calculated by using the following equation

$$precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|}$$

In semantic reasoning method, 20 documents are taken and consider 12 relevant documents and 7 retrieved documents. Then 6 documents are matched. Hence the precision value is

$$precision = \frac{\{5,7,8,9,10,11,13,14,17,18,19,20\} \cap \{3,7,13,14,18,19,20\}}{7} = \frac{6}{7} = 0.85$$

Similarly, in improved semantic reasoning method, 20 documents are taken and considered 12 relevant documents and 11 retrieved documents. Then 10 documents are matched. Hence the precision value is

$$precision = \frac{\{5,7,8,9,10,11,13,14,17,18,19,20\} \cap \{6,7,8,9,10,11,14,17,18,19,20\}}{11} = \frac{10}{11} = 0.91$$

Figure 2 shows that the precision comparison. The precision increases for the proposed improved semantic reasoning method is compared to the existing semantic reasoning method.

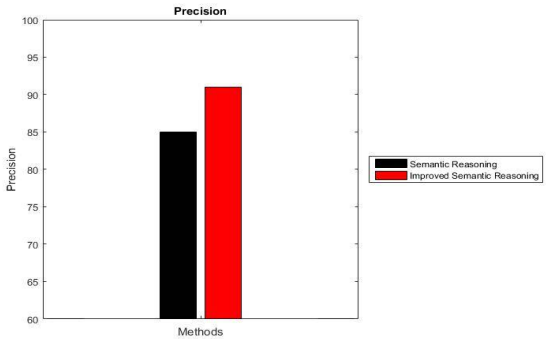

Fig 2 Precision Comparison

Table 1 Precision Comparison

	Semantic Reasoning	Improved Semantic Reasoning
Precision	85	91

Based on the table the result shows that the precision is increased for proposed improved semantic reasoning method compared to the existing semantic reasoning method. It can be proved that the proposed methodology provides better results than the existing.

Recall

The recall or true positive rate (TP) is the proportion of positive cases that were correctly identified, as calculated using the equation:

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrived documents}\}|}{|\{\text{relevant documents}\}|}$$

In semantic reasoning method, 20 documents are taken and considered 12 relevant documents and 7 retrieved documents. Then 6 documents are matched. Hence the recall value is,

$$recall = \frac{\{5,7,8,9,10,11,13,14,17,18,19,20\} \cap \{3,7,13,14,18,19,20\}}{12} = \frac{6}{12} = 0.50$$

Similarly, in improved semantic reasoning method, 20 documents are taken and consider 12 relevant documents and 11 retrieved documents. Then 10 documents are matched. Hence the recall value is

$$recall = \frac{\{5,7,8,9,10,11,13,14,17,18,19,20\} \cap \{6,7,8,9,10,11,14,17,18,19,20\}}{12} = \frac{10}{12} = 0.83$$

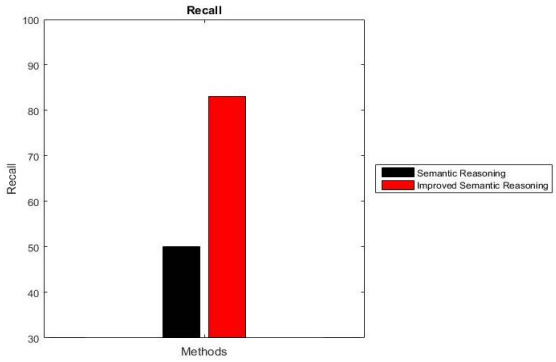

Fig 3 Recall Comparison

Figure 3 shows that the recall comparison. The recall increases for proposed improved semantic reasoning method is compared to the existing semantic reasoning method.

Table 2 Recall Comparison

	Semantic Reasoning	Improved Semantic Reasoning
Recall	50	83

Based on the table, the result shows that the recall is increased for proposed improved semantic reasoning method compare to existing semantic reasoning method. It can be proved that the proposed methodology provides better results than the existing.

F-Measure

The F-Measure is described the average of the information retrieval precision and recall metrics

$$F - \text{measure} = 2.\frac{\text{precision. recall}}{\text{precision} + \text{recall}}$$

In semantic reasoning method produced the precision value 0.85 and recall value 0.50, then the f-measure value is,

$$F - measure = 2.\frac{(0.85 * 0.50)}{0.85 + 0.50} = 0.62$$

In proposed method produced the precision value 0.91 and recall value 0.83, then the f-measure value is,

$$F - measure = 2.\frac{(0.91 * 0.83)}{0.91 + 0.83} = 0.86$$

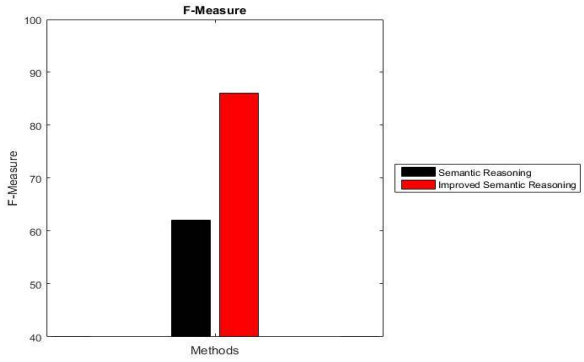

Figure 4 Recall Comparison

Figure 4 shows that the F-measure comparison. The F-measure increases for proposed improved semantic reasoning method are compared to the existing semantic reasoning method.

Table 3 F-Measure Comparison

	Semantic Reasoning	Improved Semantic Reasoning
F-Measure	62	86

Based on the table the result shows that the F-measure increased for proposed improved semantic reasoning method compare to existing semantic reasoning method. It can be proved that the proposed methodology provides better results than the existing.

CONCLUSION

A new approach is proposed for measuring semantic consistency to requirement documents with natural language lexical parsing. The improved semantic reasoning method mentioned here is to extract the pairs of antonyms between adjectives and adverbs. In this work, additionay the synonym, hypernyms/ hyponyms, and acronym extraction process is added to compute the of pairs of antonym in the specification. This process of extraction in the requirement documents has improved the semantic reasoning. Hence, the experimental assessments were carried out and proved that this improved semantic consistency approach proposed here provides a better result than the existing approach in terms of precision, recall and F-Measure.

References

Nuseibeh, Bashar, Steve Easterbrook, and Alessandra Russo (2000). Leveraging inconsistency in software development. *Computer* 33.4: 24-29.

Nuseibeh, Bashar, Steve Easterbrook, and Alessandra Russo (2001). Making inconsistency respectable in software development. *Journal of Systems and Software* 58.2: 171-180.

Easterbrook, Steve, and Marsha Chechik. 2nd international workshop on living with inconsistency (IWLWI01) (2001). ACM SIGSOFT Software Engineering Notes 26.6: 76-78.

Hunter, Anthony, and Bashar Nuseibeh. Managing inconsistent specifications: reasoning, analysis, and action (1998). ACM Transactions on Software Engineering and Methodology (TOSEM) 7.4: 335-367.

Yesudoss, J., and A. V. Ramani. Ontological based Relevance Abstraction Identification Technique and Evaluation (2016). *Indian Journal of Science and Technology* 9.32.

Yan, Rongjie, Chih-Hong Cheng, and Yesheng Chai. Formal consistency checking over specifications in natural languages (2015). Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. EDA Consortium.

Mu, Kedian, *et al*. Measuring Inconsistency in Requirements Specifications (2005).ECSQARU.

Perrouin, Gilles, *et al*. Composing Models for Detecting Inconsistencies: A Requirements Engineering Perspective (2009).REFSQ. Vol. 5512.

Baclawski, Kenneth, *et al*. Consistency checking of semantic web ontologies The semantic web-ISWC (2002): 454-459.

Fuxman, Ariel, *et al*. Model checking early requirements specifications in Tropos (2001). Requirements Engineering. Proceedings. Fifth IEEE International Symposium on IEEE.

Nuseibeh, Bashar, Steve Easterbrook, and Alessandra Russo (2000). Leveraging inconsistency in software development. *Computer* 33.4: 24-29.

Misra, Janardan. Terminological inconsistency analysis of natural language requirements (2016). Information and Software Technology 74: 183-193.

Mu, Kedian, *et al*. From inconsistency handling to non-canonical requirements management: A logical perspective (2013). *International Journal of Approximate Reasoning* 54.1: 109-131.

Ali, Raian, Fabiano Dalpiaz, and Paolo Giorgini. Reasoning with contextual requirements: Detecting inconsistency and conflicts (2013). Information and Software Technology 55.1: 35-57.

Zhang, Xiaowang, *et al.* A distance-based framework for inconsistency-tolerant reasoning and inconsistency measurement in DL-Lite (2016). *International Journal of Approximate Reasoning.*

How to cite this article:

Yesudoss J and Ramani A.V.2017, Inconsistency Checking In Requirements by Improved Semantic Reasoning (ICRISR). *Int J Recent Sci Res.* 8(8), pp. 19461-19466. DOI: http://dx.doi.org/10.24327/ijrsr.2017.0808.0698

****