Measurement and Removal of Inconsistencies from Software Requirement Specification

Ashima Rani, Gaurav Aggarwal

Abstract:

Inconsistency is a major challenge in software requirements. Conventionally, SRS(Software Requirement Specifications) document are projected to be consistent[1]. In Software Requirements, measurement, classification and removal of inconsistencies is a complex task which is the interest of many authors and group of researchers [2]. We can assist requirement engineers and company analysts to validate their specifications which are written or gathered in natural language is in line with other study and representation of nature. By enhancing we can eliminate the inconsistency approach of inconsistency and because of this approach we will boost the reliability of the inconsistency. The software specifications as well as the consistency of software specification requirements will be increased.

Keywords: Inconsistency, Ontology, Natural Language Processing, Software Requirement Specification, First Order Logic.

1. Introduction:

Ontology is a philosophy branch, regarding the essential characteristics of all truth in the universe. Currently it has come to be in various cultures, including artificial intelligence, information engineering, and natural language processing. For each culture, different meanings are adopted. The commonly accepted definition is "ontology is a formal, explicit specification of a shared conceptualization". There is a great deal of well-known ontology established by various groups, For eg, WorldNet, Cyc, etc. With regard to various parameters, they are classified into distinct categories and implemented in various types of various versions [3].
There are mainly four types in an Ontology:
Entity: Represents an objects or thing, for example: person, man, woman.
Relation: Represents relationships between things, for example, a parent-child relationship between two person entities.
Role: Describes the participation of *entities* in a *relation*. For example, in a *marriage* relation, there are roles of *husband* and *wife*, respectively.
Resource: Represents the properties associated with an *entity* or a *relation*, for example, a name or date. Resources consist of primitive types and values, such as strings or integers.
The main objective of this paper to measurement the inconsistencies from software requirement specification document (SRS) and removal of inconsistencies from the SRS ontology.

2. Inconsistency:

2.1 What is Inconsistency in Ontology?

To denote any situation in which a set of definitions does not follow any relationship, we use the term inconsistency should keep between them. The correlation between descriptions can be expressed as a rule of coherence against which the description can be checked. Some laws can be captured in current practice in explanations of the development process; others In development tools, they can be embedded. Nevertheless, the bulk of these kinds of laws are not registered anywhere [4].

2.2 First-Order Logic

First-order logic is a formal logical system which, through additional logical operators of quantifiers [3], extends propositional logic. It is designed around objects and relationships and provides a well-defined representation of information that is ideal for automated representation.
Reasoning and comparatively quick to grasp for analysts. For ontology modelling, FOL was applied. For example, in ontology modelling, the Information Interchange Format (KIF), based on FOL, was proposed. Logic representations of recent times of restricted expressiveness, such as the logic of explanation, has been commonly used, but extra logical assumptions are important to catch the intended semantics of the words in ontology.

2.3 Semantic Networks

The graph of the structure of meaning is a semantic network. Specifically, "it is a graphical representation notation" Information of interconnected nodes and arc patterns. The definitions are defined by the nodes and the arcs are the interrelationships between each two nodes. This offers a convenient method for visualizing a knowledge base. The Semantic Network was used for several ventures in the creation of ontology. Semantic networks are believed to be the most suitable representation form for the vast quantities of semantic information in an intelligent system are recorded and encapsulated [3].

2.4 Causes of Inconsistencies

An inevitable aspect of software development processes is inconsistency. Even in the most well-defined, controlled and operated environment, optimized process of development, often unclear or conflicting system specifications, alternative design solutions exist, and implementation errors occur. The engineering stage of development requirements are especially illustrative of such

Inconsistencies. During requirements acquisition, customer requirements are often sketchy and uncertain. For large projects in particular, a number of “client authorities” may exist who have conflicting, even contradictory requirements [5].

2.5 Checking Consistency Rules

Here are three examples of consistency rules expressed in english:

1. In a dataflow diagram, if a mechanism is decomposed into separate diagram, the input flows to the parenting process must be similar to the feedback flows to diagram of child data flow.
2. For a limited library system, the idea of the contract for the operations notes that the recipient and the creditor are synonyms. Hence, the consumer list the actions listed in the assist manuals must suit the list of borrower actions in the help manual specified criteria.
3. Coding will not start until the systems specification of the specifications has been signed off by Review Board of the enterprise. And the programme code repository should be empty before status is defined changes SRS to "approved."

2.6 Detecting & Identifying Inconsistencies

Once consistency in terms of explicit rules has been established, inconsistency can be automatically detected by checking these rules. A type checker, for instance, will check whether an instance or variable conforms to its type specification or not. Likewise, a parser can check whether a sentence complies with the syntactic rules defined by its grammar or not. Simple Classical inferences to detect logical anomalies resulting from too much or too little knowledge, logic may also be used. For instance, this may detect a contradiction (the simultaneous statement, or deduction, of both X and X). Other forms of inconsistency are more difficult to spot or detect [5].

2.7 Acting in the presence of inconsistencies

The methods mentioned above manage inconsistencies in various ways. However, what they have in common is the objective of allowing continuing growth in the presence of inconsistency. Such interventions can include:

Ignoring the inconsistency completely and continuing development regardless. This may be appropriate in certain circumstances where the inconsistency is isolated and does not affect further development, or prevent it from taking place.

2.8 Handling Inconsistency

The choice of a strategy for inconsistency-handling depends on the meaning and effect it has on other aspects of the development process. It can be as easy to overcome the inconsistency as adding or removing information from a software description. But it also depends on basic disputes to be resolved or essential design decisions to be made. In such examples, instantaneous settlement is not the only choice. You can miss, postpone, circumvent, or minimize the difference. Sometimes the efforts to fix an inconsistency is substantially greater than the probability that there will be any detrimental effects of the inconsistency. In such examples, you will decide to ignore the difference. Good practice needs such decisions to be revisited as a project advances. Or as a structure evolves. Deferring the decision until later can give you more time to obtain additional information in order to facilitate resolution or to make the difference unimportant.

2.9 Measuring Inconsistency

Measurement is central to effective management of inconsistencies for several reasons. Developers use it most to know the amount and extent of anomalies in their definitions, and how they alter differently.

Developers may also use these measures to compare descriptions and assess, given a choice, which is preferred.

Sometimes developers need to prioritize different ways of identifying inconsistencies that desperately needs care. Also, they would need to assess their success by assessing how well they adhere to some predefined standard or process of development design.

Measures often taken to address inconsistencies rely on an impact evaluation of those acts have on project development. Measure the consequently, the effect of inconsistency care behavior is a key to effective action when inconsistency arises [9].

3. Proposed Work:

In this Section, we have to improve the inconsistency which is occurred in software requirement specification document. By enhancing inconsistency will remain, but the specification of the software requirement requires completeness and accuracy may decrease. By this approach, this limitation can strengthen and completeness can also be retained. We have proposed a algorithm that can remove inconsistency from software requirement specification ontology.
Algorithm:

Step 1: Prepare the ontology of software requirement specification (SRS) document.
Step 2: Checking rules with full document.
Step 3: If any inconsistent is detected
Step 4: Save it in temporary file.
Step 5: Then retrieve the link from the SRS document and remove it from document.

4. Evaluation:

In this section, we have applied the algorithm on Library management system document. Firstly we have developed the ontology of library management system and then read the software requirement specification (SRS) document. Checking the rules with whole document if any rule is broken down then we can say that contradictions arise in SRS document or inconsistencies occurs in the SRS document. After that we have to remove the inconsistencies from SRS document. Figure 1-4 shows the steps.

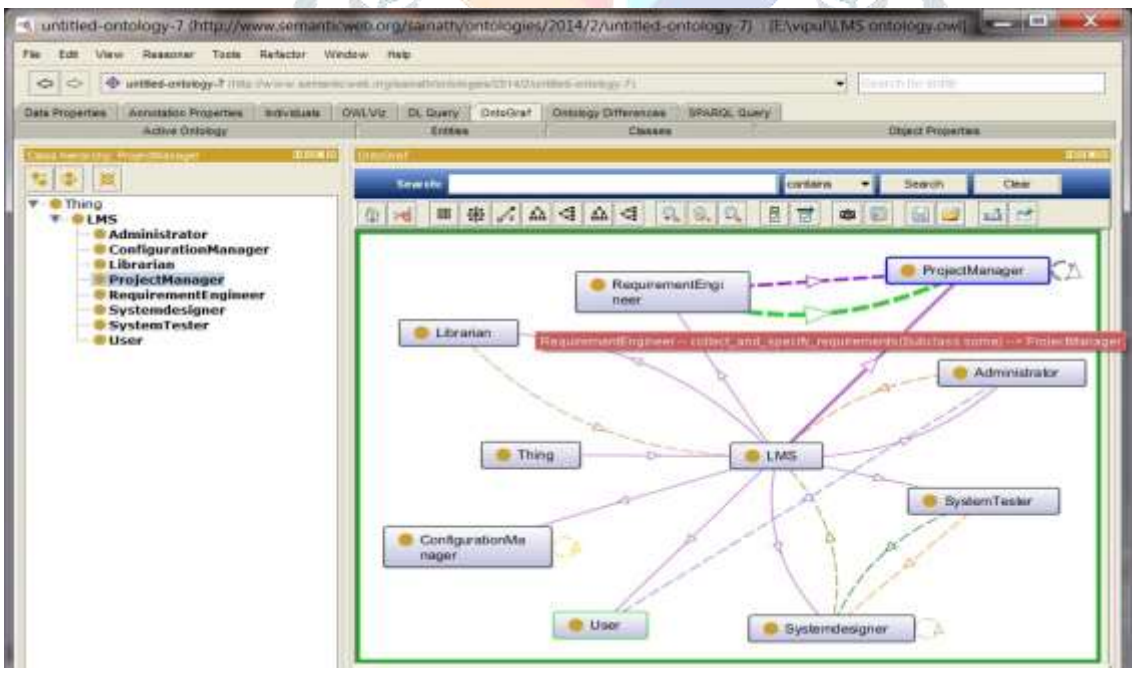

Fig 1: Ontology of Library Management System

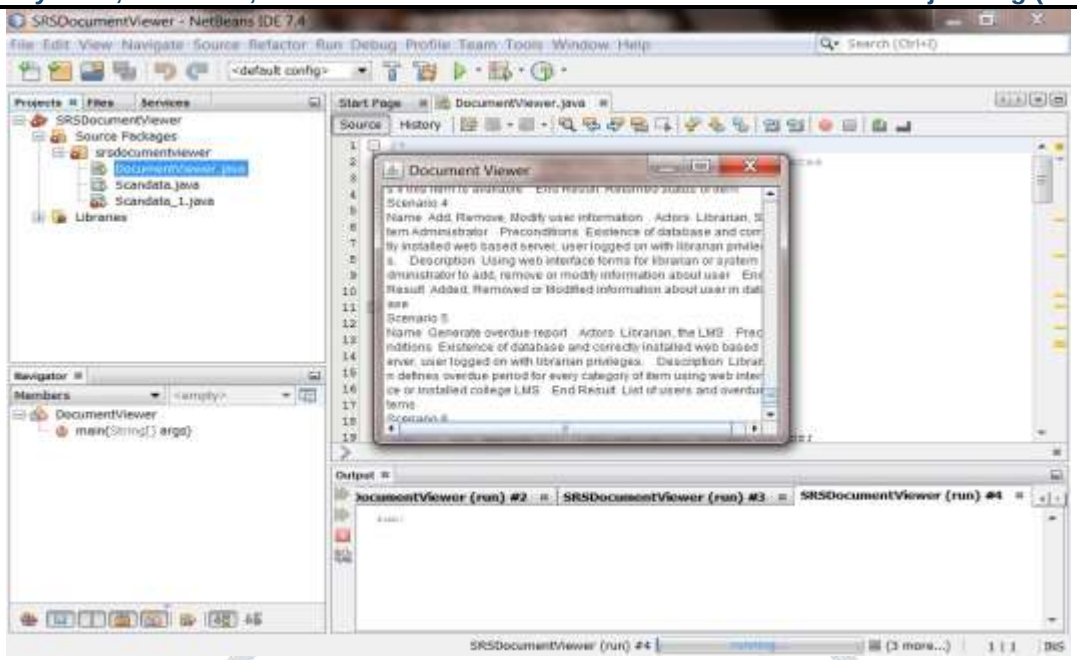

Fig 2: Read the Software Requirement Specification (SRS) document

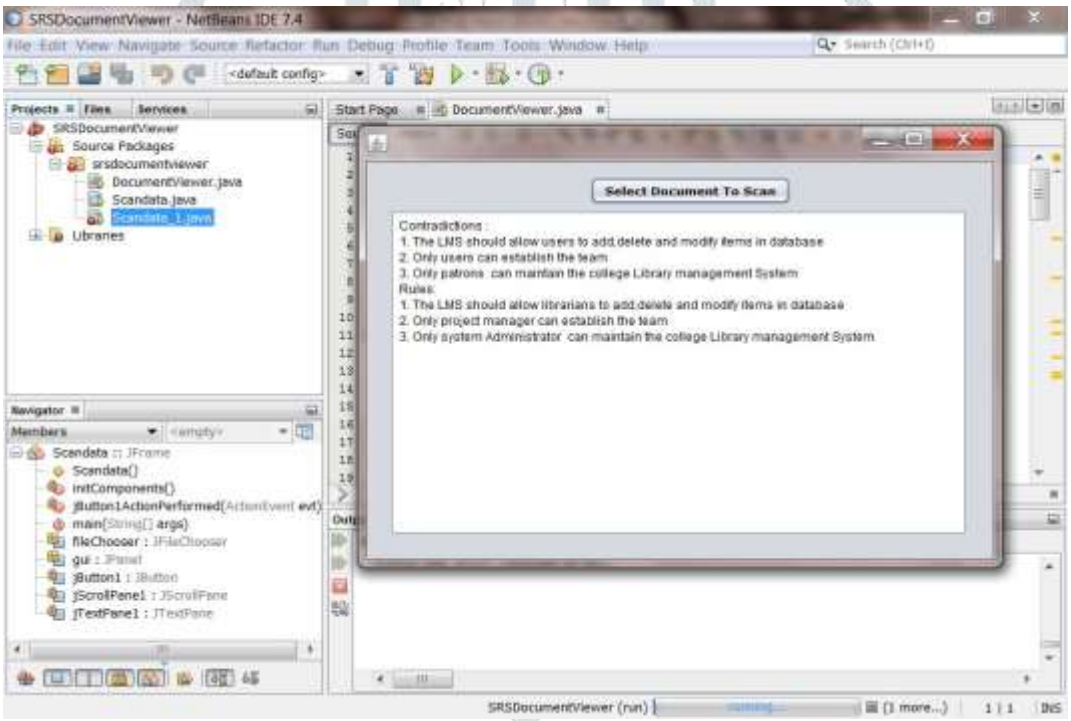

Fig 3: Detecting Inconsistencies in Software Requirement Specification (SRS) document

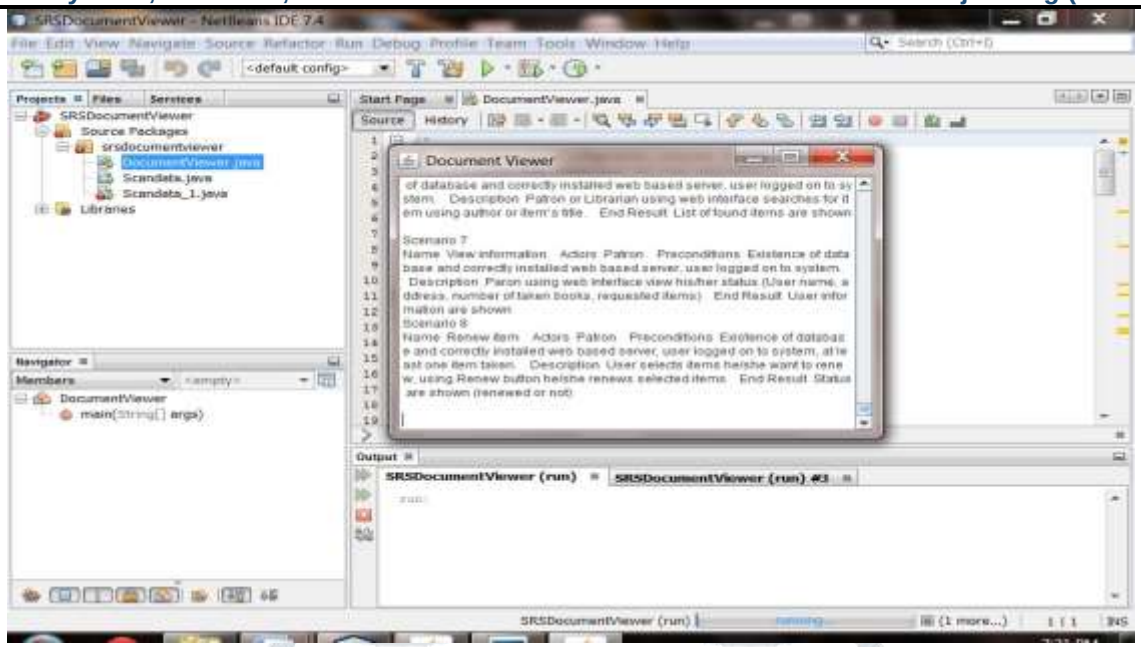

Fig 4: Removal of inconsistencies from SRS document

5. Conclusion:

This paper defines the algorithm for measurement and removal of inconsistencies from Software Requirement Specification (SRS) document. In software requirement specification there are various rules are defined and when the rule is broken down inconsistency arise in srs (software requirement specification) because of some contradictions. Here we have tried to remove the contradictions from software requirement specification and we can achieve the correctness of software requirement specification. This way we can improve the quality of the software requirement specification.

References:

[1] T´ulio Brand˜ao Soares Martins, Julio Cesar dos Reis. Addressing Inconsistencies in the DBPedia Evolution. Instituto de Computa¸c˜ao, Universidade Estadual de Campinas, 13081-970 Campinas, SP, December 2019.

[2] Harald Sack Gerald Topper, Magnus Knuth. Dbpedia ontology enrichment for inconsistency detection. In Proceedings of the 8th International Conference on Semantic Systems (I-SEMANTICS' 12), pages 33–40. I-SEMANTICS, 2012.

[3] J. M. Park, J. H. Nam, Q. P. Hu, H. W. Suh, "Product Ontology Construction from Engineering Documents", Yusongku, Taejon, 305-701 Republic of Korea, International Conference on Smart Manufacturing Application,2008.

[4] Bashar Nuseibeh, Steve Easterbrook, Alessandra Russ "Leveraging Inconsistency in Software Development"IEEE, 2000.

[5] Bashar Nuseibeh,"To Be and Not to Be: On Managing Inconsistency in Software Development", London, IEEE, 2001.

[6] Denger, C., D.M. Berry, and E. Kamsties, "Higher Quality Requirements Specifications through Natural Language Patterns", in Proceedings of the IEEE International Conference on Software- Science, Technology \& Engineering, IEEE,2003.

[7] V.Lamsweerde, R.Darimont, E.Letier. "Managing Conflict in Goal-Driven Requirements Engineering",Transactions on Software Engineering, Vol.24,IEEE, 1998.

[8] Egyed, A., "Scalable Consistency Checking Between Diagrams-The ViewIntegra Approach", in Proceedings of the 16th IEEE international conference on Automated software engineering, IEEE, 2001.

[9] Massila Kamalrudin,"Automated Software Tool Support for Checking the Inconsistency of Requirements" University of Auckland, Private Bag International Conference on Automated Software Engineering, IEEE/ACM, 2009.

[10] Ashima Rani, Gaurav Aggarwal, " Algorithm for automatic detection of ambiguities from software requirements", International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278-3075, vol 8, issue-9s,July 2019.