CBASeRA: A Compiler Based Approach towards Semantic Requirements Analysis

Atifa Rafique[1], Kashif Ayub[2], Muhammad Ilyas[3] and Josef Kueng[4]

[1,2,3]Department of Computer Science and IT, University of Sargodha, Sargodha, Pakistan
[4]Institute for Application Oriented Knowledge Processing, Johannes Kepler University of Linz, Austria

Abstract

Requirements Engineering (RE) becomes one of an important aspect in Software Development Life Cycle (SDLC). To improve the overall process of requirements engineering, different techniques, models and practices are available. Semantic based requirements analysis and verification is one of the techniques to improve the overall quality of software. A new methodology is proposed for semantic based requirement analysis and verification which practices compiler based approach. Lexical Analyzer highlights ontologies from each requirement specification, described in natural language. During Ontologies Recognizer, the relationships and entities are extracted by acknowledging ontologies. Now by applying SQL commands on ontologies (entities and relationships), we form Requirements Knowledge Base. Tree Based Semantic Analyzer constructs a well-structured tree of entities and relationship. Semantic application on this tree presents the requirements in unambiguous form. In order to remove the ambiguities, the process of semantic based requirements analysis and verification is described with a case study.

Keywords: *Requirements, Requirements Analysis, Ontologies, Knowledge Base, Compiler, Knowledge Base*

1. Introduction

In Software Development Life Cycle (SDLC), requirements engineering plays a major role. RE is the process of identifying the stakeholder's needs. It is also used to document the requirements for the analysis, communication and development of software. RE consists of many activities like eliciting, modeling, analyzing, communicating, agreeing, and evolving requirements [20]. A lot of research work is performed in this area of software engineering. The American Standish Group's professional research indicates that's the 50 percent of software failure is almost depend on factors that are related to software requirements [1]. Many theories and practices are available for RE, but there is a big gap between these theories and practices which make a RE process weak [2]. So RE must be strong enough that they increase the overall quality of software.

In RE, the involvement and usage of semantic web and ontologies has given innovative dimensions. These areas facilitate the requirements elicitation and analysis process. They allow communication and understanding between stakeholders throughout the SDLC [3]. Semantics are meanings, which used for requirements analysis and verification. Requirements can be describe in many form like it may be written in natural language (English), in mathematical form, or in semi-formal form (use cases, scenarios) [22, 23]. The ambiguity can be arises in semi-formal or natural language representation of requirements because for most of the participants, it is difficult to understand the requirements which are written in these from.

RE can also be supported with Knowledge Base (KB) systems. They are helpful for the requirements analysis and verification because they can explore large amount of application specific knowledge [14]. Informal requirements are gathers from customer in KB, Requirements Analysis and Knowledge Elicitation System (RAKES) and then produce the formal specification. RAKES is a good example of KB implementation [13]. In this paper, we will use Requirement Knowledge Base (RKB), which is used to store the requirements.

For semantic and syntactic analysis of natural language statements, Compiler Based Approach (CBA) is used. Compiler consists of front end and back end interfaces. Front end have phases like lexical analyzer, syntax analyzer, semantic analyzer, and intermediate code generator. Back end interface consists of code optimizer and code generator [15]. Front end compiler is also used for natural language processing. Many techniques are available for processing the statements in natural language to check their syntax and semantics. For example, first they find out the keywords and then describe the meanings of sentence. One of available technique which is used to check the semantics of sentences in natural language is CBA. In this technique the input string is organized as noun, prepositional, verb, adverb, adjective phrases etc. Then these phrases are analyzed during semantic analysis process to remove the ambiguities in words. Knowledge

base is used to store all the possible meanings for words [16]. In order to extract information from such KB, we use Structured Query Language (SQL) for database manipulation. Here's SQL commands are used to manipulate ontologies in Knowledge base.

New requirements analysis and verification method is introduced in this paper which is CBA. Requirements are defined in the form of natural language statements. Now the first phase of proposed framework is Lexical Analysis through which the entities and relations are highlighted and then Ontologies Recognizer generates the ontologies from highlighted information. SQL commands are used to form a RKB, from which Tree Based Semantic Analyzer construct a tree. Finally the Requirements Generator produces the furnished requirements by parsing a tree.

In this paper we present the CBA for Semantic Based RE. For requirements analysis and verification, RKB and SQL commands are used. Section 2 describes context and background work that is related to RE, Semantic RE, RKB and Compiler. In section 3 the detailed proposed approach is provided. While Section 4 discusses the case study which support our proposed framework. Section 5 presents the prototype for proposed approach. The final section gives a conclusion followed by future work.

2. Related Work

In past, many research works have been performed to support the RE process. Bashar Nuseibeh et al. presented a comprehensive roadmap to RE. RE activities are re-examined and associated with only core activities. They include the need for requirements elicitation, analysis and modeling [20]. Opdahl et al presented a review of research papers published at ten conductive annual workshops on RE. They performed qualitative analysis for the evolution of RE activities, so the research interests are disclosed [21].

Semantic based RE and ontologies techniques are used to improve the RE process [18]. These assist in requirements elicitation and analysis process. FengdiShu, et al. provides the method of requirements elicitation, which based on users individualities and context. This method encourages the user involvement in requirements elicitation and also improves the domain knowledge reuse [17]. Ontologies play an important role in software applications development [4]. These applications include natural language processing [5], databases [6], multimedia [7], data mining [8], and information retrieval systems [9].

Haruhiko et al. proposed an approach in which they have applied the knowledge of domain ontology towards the requirement elicitation and analysis process. They map the software requirements description with the domain ontology. Their domain ontology system contains inference rules and thesaurus parts that are suitable for processing the semantics. It facilitates the requirements engineers for requirements specification analysis with respect to application domain semantics. They show three types of semantic processing with the help of case study. These semantic processing includes: identifying the inconsistent and incomplete requirements, quantifying the requirements specification through its meaning, and predicting the changes in requirements [18].

Requirements analysis and verification can be facilitated by the semantic based approaches. A new semantic approach based on domain methodology is presented for the analysis and the verification of requirement [10]. Semantic Wiki is one of the semantic based approaches which is used for RE. It is used explicitly to expose the relationship between requirements elements [11]. Yanwu Yang, et al. presented an integrated (two level) framework for semantic based RE (shown in fig. 1). This two level framework is basis for requirements understanding and management. It is also act as eliciting, analyzing, modeling, communication, and approving requirements. The lower level integrates the user ontology, enterprise ontology and domain ontology for the semantic representation of software requirements. User ontology is used for eliciting and modeling the requirements when user has no clear idea what they want. Enterprise ontology defines the rules, goals, resources and responsibilities with respect to business to hold high level requirements. Domain ontology plays key role, it assists the stakeholders to share background knowledge. The middle level of this framework consists of RE activities including modeling, analysis, communication and evolution. Requirements knowledge is acquired according to application domain. Requirements knowledge base is used to structured and store this knowledge. Requirements items can be analyzed in figure 1 [19].

Semantic based composition is another idea used in formal semantic studies, which benefits the reasoning to identify the conflicts between requirements and also assist the meaningful mapping to derived architecture [12]. Haibo Hu, et al., proposed an approach, a structural and formal semantic based approach on domain ontology. Moreover, it uses inference rule for analysis and verification of software requirements. They described requirements in natural languages.

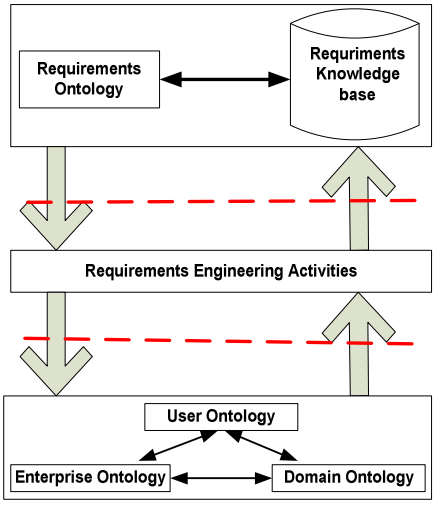

Fig.1 An integrated framework for semantic requirements engineering [19].

For requirements analysis, requirements descriptions are decomposed into atomic requirement item which represents with triplet *D (C, R, AR, X)* of semantics in domain ontology. Now the requirements are analyzed and verified according to provided framework (figure 2). The domain ontology which is described in this paper consists of semantic elements. These elements are represents as concepts and relationship between these concepts and rules of inference. They map the requirement elements to the domain ontology with the help of inference rules to analyze and verity the completeness, correctness, and consistency. In order to support their idea they used predicate calculus notations for requirements elements and domain ontology representations. They introduced notations for mapping functions *Fm(r)* and inference rules *D(c)/D(p)* which describe the instance concept ***c*** or binary relation ***p*** [10].

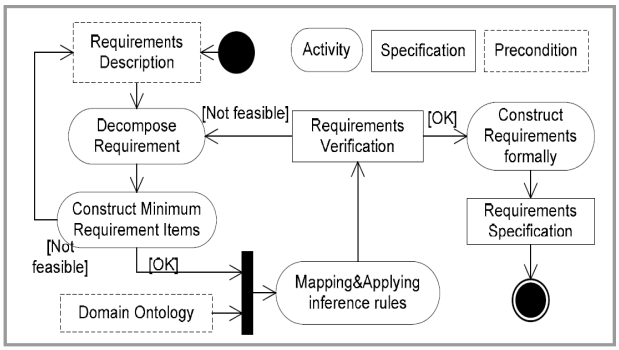

Fig. 2 Framework of requirement analysis based-on ontology [10].

The KB is another important aspect for analysis of requirements used in RE. In a Knowledge based approach, RAKES is implemented for requirements analysis. RAKES take informal requirements from user and produce formal specification. For input, system uses the requirements in natural language and produce output in Formal specification. It also produces another kind of output like as side notes that is stored in knowledge base. A formal analysis is performed to analyze the requirements. This approach is totally concerned with analysis phase; however the information that is stored in knowledge base can also be useful during the entire software life cycle [13]. Overview of RAKES is shown in figure 3.

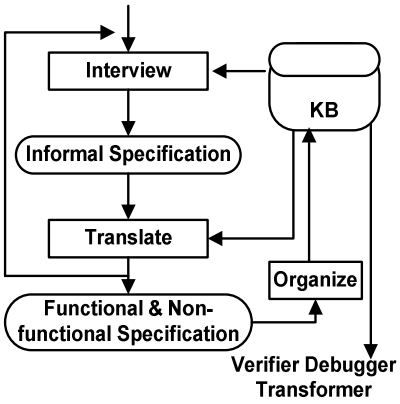

Fig. 3 Overview of RAKES [13].

For semantic and syntactic analysis, another good approach is the usage of compilers, which have many applications used in natural language processing (shown in figure 4). It extracts the words and phrases (verbs, nouns, adverbs, adjectives …) from natural language statements. These words and phrases are analyzed to remove the ambiguities from statements. Knowledge Base is used to keep the record of application specific knowledge and general knowledge. It contains all possible meanings of sentences. Lexical analyzer check the syntax and semantic of each word in a statement and then store it in Knowledge Base. In this method the input strings are converted to SQL statements and then computer run these statements [16].

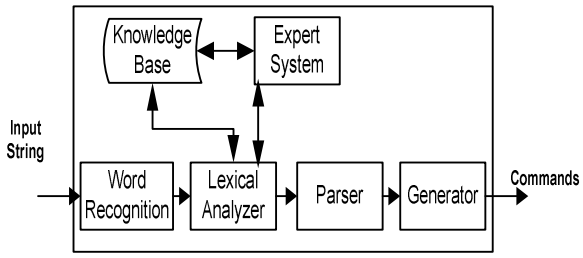

Fig. 4 Natural Language Processing [16].

3. Proposed Framework

This paper proposed a new requirements analysis and verification method which uses CBA. Here in this idea a string is an input, this string represents a requirement in simple natural language. Exactly one information string represents a requirement. The requirement string as input went to Lexical Analyzer. This analyzer reads the string and highlights the entities and relationships. Entities can be subjects or objects and relationships can be verbs. Now the highlighted information comes into Ontologies Recognizer, where Ontologies Jars are generated on the bases of highlighted information from input string. Now the actual string shrinks to core information related to some requirement. Requirement Knowledge Base (RKB) forms a Knowledge Base via SQL statement on the bases of relations between recognized ontologies. Tree Based Semantic Analyzer generates a well-structured tree of information from these ontologies, so the information gets some shape of representation. Now Requirement Generator generates requirements from constructed tree of ontologies and sends these requirements as its output.

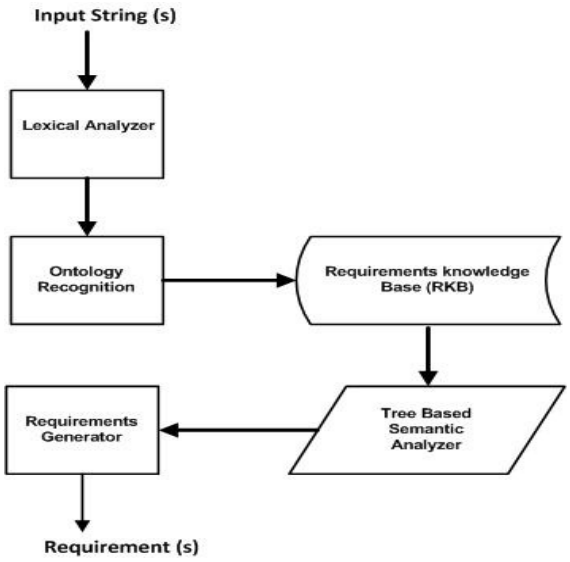

Fig.5 CBASeRA Framework

3.1 Lexical Analyzer

Lexical Analyzer gets string as an input. This string consists on a natural language sentence which completely describes a requirement. This requirement passes to Lexical Analyzer where process re-reads the string completely and highlights some very basic information to it like entities and relationship. Normally these relations are of general type in these natural language sentences.

3.2 Ontologies Recognizer

Ontologies Recognizer recognizes the highlighted information (Entity1, Entity2, Relation1…) from a string and generates ontologies on the basis of this highlighted information, embeds some extra information like relationships between them. Now actual string breaks into small parts of information and shrinks as well.

3.3 Requirements Knowledge Base (RKB)

In this process, Ontologies Jar (O_1, O_2, O_3…, O_n) form a RKB by using SQL commands. This KB can be of nested form, means multiple relationships can be describe in it. Like as:

Relation1 (Entity1, Relation2 (Entitiy2, Entity3)) etc.
Above example of KB is actually describing multiplicity of relationships between ontologies. RKB is shown in figure 6.

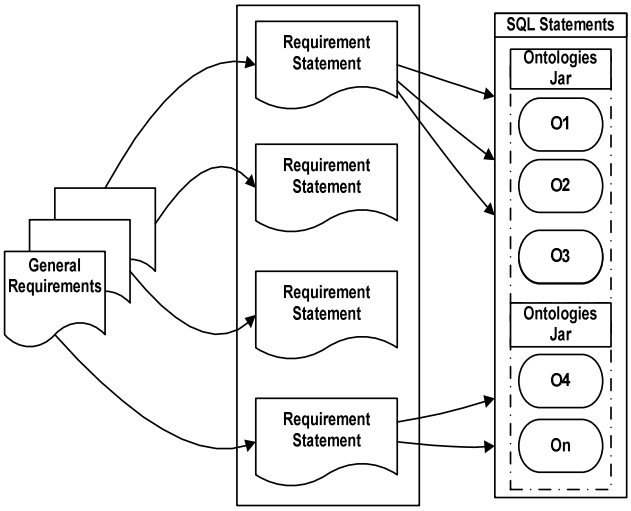

Fig.6 RKB Representation in CBASeRA Framework.

3.4 Tree Based Semantic Analyzer

Tree Based Semantic Analyzer gets the RKB as an input to construct a tree. Normally relationships between entities become even levels of tree and entities on odd levels. This is a valid scenario if we have binary relationships (as shown in fig. 7), but when there came multiplicity in relationships then this rule is no longer remaining applicable. It increases the possibility to have relationships on either level. But one thing is sure either in binary relationship tree or in multiplicity relationships oriented tree, top node on level zero is always a relationship node. This starting node can't be entity in any case. After generating a well-structured tree, now information of KB is in a well presentable form for any level of study. At this

instant, any type of processing is applicable on this tree of information.

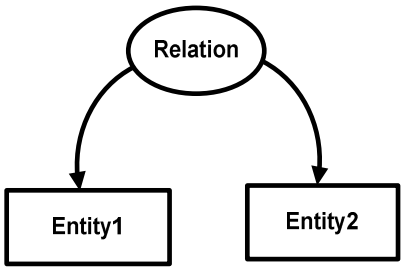

Fig.7. Binary Tree Representation

3.5 Requirements Generator

Now requirement generator process is going to extract requirements via tree parsing. Tree has entities and relationships between these entities nodes. Requirement generator process is using same information to go through this tree and extract requirements. These requirements can be a string or in some other narrations for requirements. Here these requirements are the outputs from our framework of RE.

3.6 CBASeRA Algorithm

The algorithmic steps of CBASeRA are given as:

MAIN CALL

1- Initialize requirements file as **<file>** object
2- Initialize lexical class and TreeGraph class as **<Lex>** and**<Tree>**
3- Read the requirements strings line by line from the **<file>** till to the end of the file.
4- Call the **lex.InfoExtractor<file.cirruentline>**
5- Call to **lex. Highlighter<lex.infoExtractor<file.currentline >>**
6- Save the requirements output file as knowledgebase in some file.
7- Now read the highlighted knowledgebase file as **<filehighlihter>**
8- Generate call to **lex.Ontologies<filehighlighter.currentline>**
9- Keep creating jars and put ontologies into it.
10- Read ontologies one by one and put them into the **tree.Branch**

End call from main

3.7 Library Descriptions

Library description of CBASeRA is as follows:

Lexical Class

DataStructure Dictionary

1- Load each data base file of dictionary into <FileVerb,FileNoun,FilePreposition,FileAdverb, FileAdjective>
2- Create an object of this structure

Function InfoExtractor<Line > as string

1- Read the line word by word till to the end of line appears.
2- Make check either the current word is match to <fileAdjective or FilehelpingVerb or filePreposition, word>
3- if above check passes then remove the current word to compact the information
4- if not the condition describe in 2 then skip current and move to next

Function Highlighter <Line > as string

1- Initialize a <state> object as integral information to 1 as starting state.
2- Read the <Line> object word by word till to the end of line appears.
3- On state 1 check either the current word is some subject then does highlighting and get next word from the current line and update the state object to 2. Otherwise raise error of constraint violation.
4- On state 2 matches if the current word is a verb and next appearing word is not a proper verb. Then update the state to 3 and do highlighting.
5- If next appearing word is a proper verb then keep the state to 2.
6- If the next word is “AND” or “OR” then just update state to 4.
7- On state 3 check for object, if founded then do highlighting and remain on same state.
8- If next word is “AND” or “OR” then stay state5 and search for object again.
9- If there still remains some input in the string then raise error.

10- If the string comes to end it means operation successes.

End to Class Lexical

Class TreeGraph

Function TreeBraches<info, position>

1- If the position is 0 (zero) then put on parent branch of binary tree.
2- If position is 1 then put the info on left child of binary tree.
3- If the position is 2 then put the info on right child of binary tree.

End to class TreeGraph

4. Case Study

Following is the case study of a pizza shop's SMS based Order Placing System (SMS OPS). We use natural language to describe the requirements. This case study describes implementation of our proposed framework toward semantic based requirement engineering. In this scenario user can do the following things with pizza shop's SMS OPS.

Requirements:
1. Customers can place order.
2. Customers can place from menu.
3. Customers can place order from regular deals.
4. Customer can register itself.
5. Registered Customer can place order from menu.
6. Register customers can place order from member menu.
7. Register customer can place order from regular deals.
8. Register customer place order from member deals.

Blank (␣) is a proper string in NLP. Above are the English statements which describe the requirements for SMS OPS functionalities. These statements describe each and every requirement properly and separately. These requirement statements can be in any natural language. Now we are going to apply our framework on the above mentioned requirements to get the furnished requirements.

4.1 Lexical Analyzer

This process reads all string one by one separately and highlights the relationships and entities from them. This highlight process uses the object, subject, verbs recognition pattern to highlight the words.

1. *Customers* can *place order.*
2. *Customers* can *place* from *menu.*
3. *Customers* can *place order* from *regular deals.*
4. *Customer* can *register* itself.
5. *Registered Customer* can *place order* from *menu.*
6. *Register customers* can *place order* from *member menu.*
7. *Register customer* can *place order* from *regular deals.*
8. *Register customer* place *order* from *member deals.*

Highlighted (Italic style) words provide the meaning of each requirement. This process treats each requirement individually but later we merge them to get the results as a whole.

4.2 Ontologies Recognizer

Highlighted strings (output of Lexical Analyzer) are the inputs for this process. It recognizes the highlighted words and creates the ontologies. These ontologies also consist of some additional information which is attached with them like either the word is verb, object or subject. This additional information is helpful in constructing the tree. We get the ontologies from this process are given as:
Here E denotes to Entity (Italic words) and R to Relationship (Bold word).

1. [*Customers(E)*][**Place order(R)**].
2. [*Customers(E)*][**place order(R)**][*Menu (E)*].
3. [*Customers(E)*][**place order(R)**][*Regular deals (E)*].
4. [*Customers(E)*] [*Register (E)*].
5. [*Registered customers(E)*][**Place order(R)**][*Menu (E)*].
6. [*Registered customers(E)*][**Place order(R)**][*Menu (E)*].
7. [*Registered customers(E)*][**Place order(R)**][*Deals (E)*].
8. [*Register customers(E)*][**Place order(R)**][*Deals (E)*].

4.3 Requirement Knowledge Base

This process creates Requirement Knowledge Base (RKB) by using ontologies. SQL statements fills this database oriented RKB with requirements entities with respect to relationships between then. This RKB also helps to form final tree.

Our scenario based SQL statements are as:

1. INSERT INTO RKB VALUES(customer);
2. INSERT INTO RKB VALUES(customer, menu);
3. INSERT INTO RKB VALUES(customer, deals);
4. UPDATE RKB SET customer='register customer';
5. INSERT INTO RKB VALUES(customer, menu) WHERE customer= (SELECT customer FROM RKB WHERE customer='register customer');
6. INSERT INTO RKB VALUES(customer, menu) WHERE customer='register customer';
7. INSERT INTO RKB VALUES (customer, deals) WHERE customer='register customer';
8. INSERT INTO RKB VALUES(customer, deals) WHERE customer='register customer';

Real RKB forms through relationships taken from ontologies recognizer process's output and entities from RKB database. Now real RKB looks like this:

Place order(*customer*)
Place order(*customer, menu*)
Place order(*customer, deal*)
Register customer(*customer*)
Place order(**register customer** *(customer), menu*)
Place order(**register customer** (*customer*), *menu*)
Place order(**register customer** (*customer*), *deals*)
Place order(**register customer** (*customer*), *deals*)

This is the final representation of RKB which is further used for tree construction.

4.4 Tree Based Semantic Analyzer

Tree Based Semantic Analyzer takes each statement from RKB as a separate input. It initially forms a tree for the individual requirement and at the end it forms a final tree as its output. Relationship between entities becomes parent nodes and left and right nodes are the entities associated with that relationship. So tree representation of each requirement statement is illustrated as:

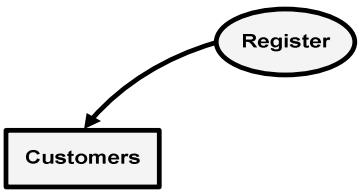

Fig.8. Tree for requirement 1.

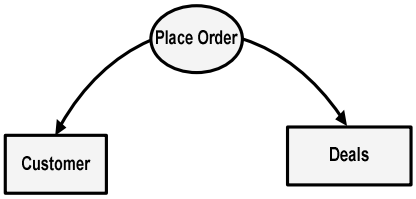

Fig.9 Tree for requirement 2.

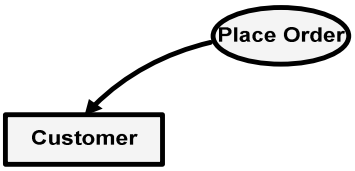

Fig.10. Tree for requirement 3.

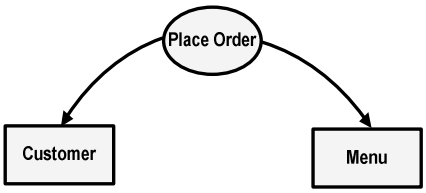

Fig.11. Tree for requirement 4.

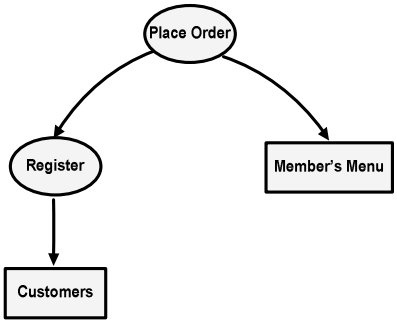

Fig.12. Tree for requirement 5.

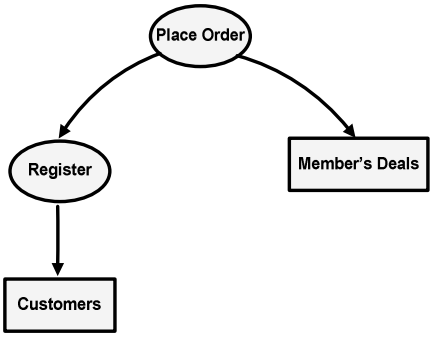

Fig.13. Tree for requirement 6.

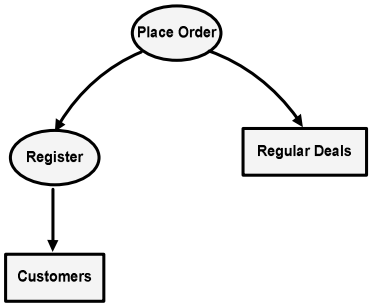

Fig.14. Tree for requirement 7.

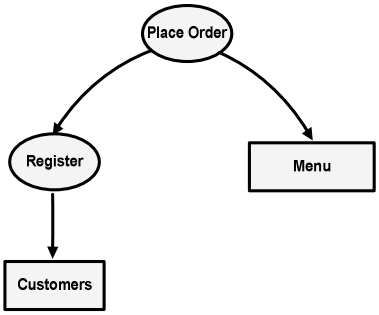

Fig.15. Tree for requirement 8

Final tree forms on bases of relationship. Here all nodes merges on the basis of similar relationship nodes and then removes the ambiguities like repeated or similar branches from these newly connected nodes of tree (figure 16).

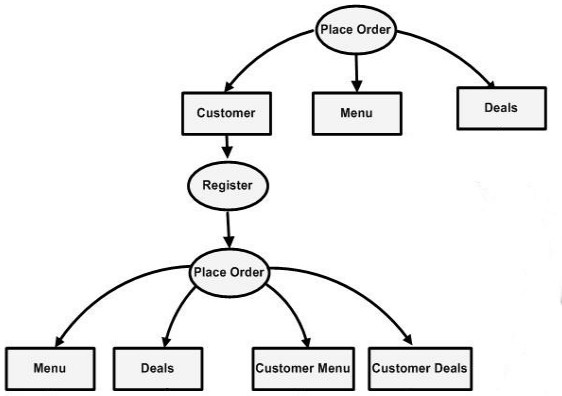

Fig.16. Final tree representation after mapping.

4.5 Requirements Generator

In this process, we get furnished requirements in any format from tree based semantic analyzer. By parsing a tree, final requirement can be either in simple English language statement or even can export complete tree of figure 16 by using some data structures like link list, queue etc.

We have designed these formulas for requirements verification:

Formula 1: Completeness = Com = Com / T
Formula 2: Ambiguous =AMB= Amb/T + Blank/ T
Formula 3: correctness = T – AMB
Formula 4: consistency = T - AMB - Blank

Where *Blank* is the total blank requirements in requirements specification document and we find this when there is no tree built for any requirement. *Amb* is the total ambiguous requirements and find when we get no proper tree for requirement. *Com* is the total number of complete requirements and this parameter is measured by counting the total requirements which has proper trees. Because blank is considered as a proper string in NLP and no tree is forms.

To check the completeness of requirements we count how much requirements forms the well structures trees. The following conclusions are drawn from using above formulas:

1- Completeness = 6/8 = 75%.
2- Ambiguous = 1/8+0/8 = 12.5 %
3- Correctness = 87.7%
4- Consistency = 87 %

5. CBASeRA Prototype

We have designed and implemented a system of our proposed approach for semantic based requirements analysis and verification (Figure 17). System consists of three parts: opening for stakeholder's requirements document and providing some functions such as editing; preceding to platform for highlight the information such as subjects, objects and verbs, recognizing the ontologies from highlighted information and then constructing the ontologies jar; platform for separate trees for each requirement specification.

5.1 CBASeRA Interface

The main interface which is used for opening and editing the requirements document is shown in figure 17.

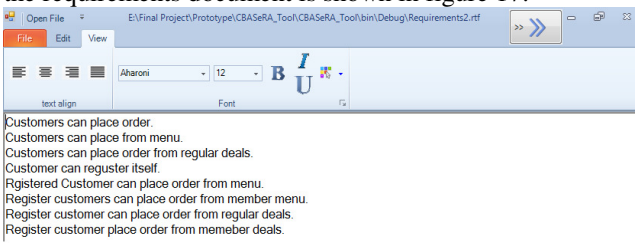

Fig. 17 Main Interface of CBASeRA

5.2 Interface for ontologies recognizer

When the requirements document is opened then click on the proceed button on right corner of figure 17. We get the output window which has three parts. The first part of output window, information such as entities (subjects or object) and relationships (verbs) and highlighted. Now this information are come into ontologies recognizer where red color indicate relationship, blue and cyan colors represents the entities. Last part of interface generates the ontologies jar where the bracket [] with each ontology indicate its requirement number. These ontologies in a jar are separated based on their colors.

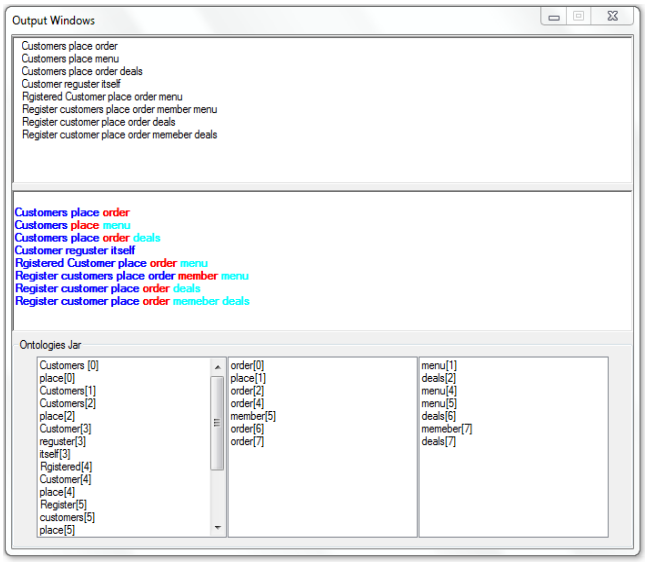

Fig. 18 Interface for ontologies recognizer and ontologies jar generation

5.3 Interfaces for requirements trees generation

Tree graph interface for each requirement shown as:

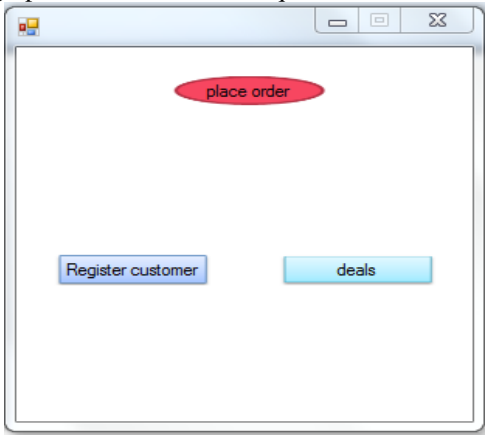

Fig. 19 Tree graph interface for requirement 5

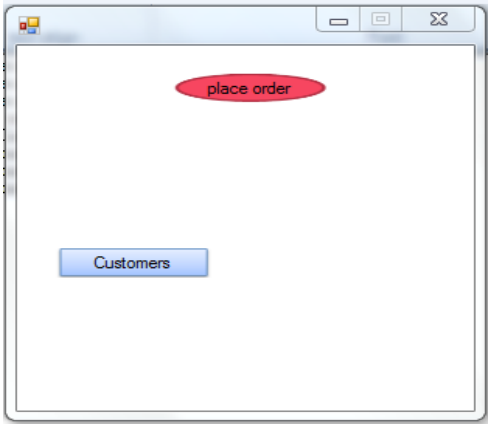

Fig. 20 Tree graph interface for requirement 3

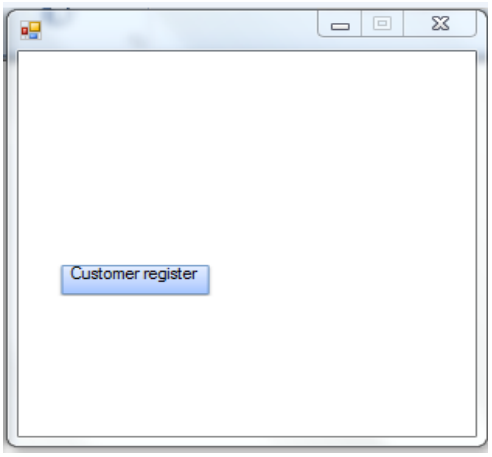

Fig. 21 Tree graph Interface for requirement 4

6. Conclusion and Future Work

This paper explores the idea for requirements analysis and verification which follows compiler based approach. Requirements are in simple English language statements and ontologies are highlighted during Lexical Analyzer phase from them. SQL commands create Knowledge Base on the bases of ontologies (entities and their relationship). Now tree based semantic analyzer construct a tree of information from these ontologies and requirement generator generates final requirements.

In future, we will try to define output format by using some data structure. Furthermore, we will try to export final tree of Tree Based Semantic Analyzer as output by using link list data structure where each node will hold some information in its data portion which will describe either this is a relation node or Entity node and some addresses which will establish connection between these list nodes.

References

[1] J. Zhi, L. Lin, and J. Ying, Software Requirements Engineering: Principles and Methodology, Science Press. 2008, pp. 4-5.

[2] D. Zowghi, and C. Coulin, "Requirements Elicitation: A Survey of Techniques, Approaches, and Tools", Engineering and Managing Software Requirements, Heidelberg, Germany, 2005, pp. 19-46.

[3] T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web", Scientific American, Vol. 184, No. 5, 2001, pp. 29-37.

[4] L. Shanping, Y. Qiwei, H. Yujie, et al., "Overview of Researches on Ontology", Journal of Computer Research and Development, Vol. 41, No. 7, 2004, pp. 1041-1052.

[5] E. Dominique, N. Chris, and Z. Andrew, "Towards Ontology-based Natural Language Processing", Proceedings of the 4th Workshop on NLP and XML, Barcelona, 2004, pp. 59-66.

[6] L. Jiang, T. Topaloglou, A. Borgida, et al., "Goal-Oriented Conceptual Database Design", Proceedings of the 15th International Conference on Requirements Engineering IEEE, New York, 2007, pp.195-204.

[7] T. Seidl, and H. Kriegel, "Efficient User-Adaptable Similarity Search in Large Multimedia Databases", VLDB97, 1997, pp. 506-515.

[8] B. Braunmueller, M. Ester, H. P. Kriegel, and J. Sander, "Efficiently Supporting Multiple Similarity Queries for Mining in Metric Databases", Proceedings of the 16th International Conference on Data Engineering, 2000, pp. 256--.

[9] R. Baeza-Yate, and B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley Publishing Company, 2008.

[10] H. Hu, L. Zhang, and C. Ye, "Semantic-based Requirements Analysis and Verification", International Conference on Electronics and Information Engineering, 2010, pp. 241-246.

[11] H. Bart, and P. Liang P, "A survey of semantic wikis for requirements engineering", Software Engineering and Architecture Group, Department of Mathematics and Computing Science, University of Groningen., Tech. Rep. RUG-SEARCH-09- L03, 2009.

[12] N. Weston, R. Chitchyan, and A. Rashid, "A Formal Approach to Semantic Composition of Aspect-Oriented Requirements" Proceedings of the 16th International Conference on Requirements Engineering, 2008, pp. 173-182.

[13] A. Liu and J. Tsai, "A Knowledge-Based Approach to Requirements Analysis", TAI '95 Proceedings of the Seventh International Conference on Tools with Artificial Intelligence, Washington, 1995, pp.26-33.

[14] R. Burlon, B. Cardile, M. Conti, F. Pietri, P. Puncello, and P. Torrigiani, "A knowledge based tool for the requirements analysis", Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences, 1989, Vol.2, pp.78-84.

[15] C. Liu, Y. Wang, and Z. Jin, "Elicit the Requirements on Software Dependability: A Knowledge-Based Approach", Software Engineering Conference (APSEC '09), Asia-Pacific, 2009, pp.233-240.

[16] W. S. Davis, and D. C. Yen, "The Information System Consultant's Handbook: Systems Analysis and Design", CRC Press LLC, ISBN: 0849370019, Dec. 1998.

[17] S. Fengdi, Z. Yuzhu, W. Jizhe, et al., "User-Driven Requirements Elicitation Method with the Support of Personalized Domain Knowledge", Journal of Computer Research and Development, Vol. 44, No. 6, 2007, pp. 1044-1052.

[18] H. Kaiy, and M. Saeki, "Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach", QSIC 2005, 2005, pp. 223-230.

[19] Y. Yanwu, F. Xia, W. Zhang, X. Xia, Y. Li, and X. Li, "Towards Semantic Requirement Engineering" ,WSCS '08, 2008, pp. 67 – 71.

[20] B. Nuseibeh, and S. Easterbrook, Requirements Engineering: a roadmap, ACM Press, 2000, pp. 35-46.

[21] A. L. Opdahl, E. Dubois, and K. Pohl, "Ten years of requirements engineering: Foundations of software quality—outcomes and outlooks", Proceedings of REFSQ'04, 2004, pp- 73-93.

[22] C. Damas, B. Lambeau, P. Dupont, and A. Lamsweerde, "Generating Annotated Behavior Models from End-User Scenarios", IEEE Transaction on software Engineering, 2005, Vol. 31, No. 12, pp. 1056-1073.

[23] Z. Yan, H. Jun, Y. Xiaofeng, et al., " Scenario-Driven Component Behavior Derivation", Journal of Software, (in Chinese), Vol. 18, No. 1, 2007, pp. 50-61.

[24] V. A. Alfred and R. Sethi, Compiler Design and Construction, Hardcover 2nd edition, Van Nostrand Reinhold, 1987.

Atifa Rafique is a student of the Master of Science in Computer Sciences at the University of Sargodha. Her research interests include Software Engineering, Semantic Web and other topics.

Kashif Ayub is a student of Bachelor's of Science in Computer Sciences. at the University of Sargodha. His research interests include Software Engineering, and Semantic Web

Muhammad Ilyas is an assistant professor at the University of Sargodha, Pakistan, PhD awarded by Johannes Kepler University, Linz Austria.

Josef Kueng is a professor at Institute for Application Oriented Knowledge Processing, Johannes Kepler University, Linz Austria.