ISSN No. 0976-5697

Volume 3, No. 7, Nov-Dec 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Extraction of Class Model from Software Requirement Using Transitional SBVR format at Analysis Phase

Prof. D.M.Thakore
Department of Computer Engineering
BVU College of Engineering Pune, India
deventhakur@yahoo.com

Ravi P.Patki*
Department of Computer Engineering
BVU College of Engineering Pune, India
patkiravi@gmail.com

Abstract: Object oriented Analysis accomplished by building several representation of system such as use case model, class model. To obtain the basic building blocks of such representation from the unstructured textual requirement specification expressed in English like natural language is not a simple task. Analyzing requirements and generating the class model artefacts to build analysis model are huge and complex task which need automated support. In the last two decades, major tools that can automatically analyze the Natural Language requirement specification and generate the class models are developed. Most of the attempts are concentrating on generation of incomplete class model. Also none of these tools cannot be used in real time software development as they provide with quite less coverage and accuracy (60% to 75%) in generating software artefacts. The key reason of lesser accuracy that has been identified by various researchers is ambiguous and informal nature of natural languages.

To overcome some of this problem in creating class model this paper proposes techniques that aim at to automatic generation of class model elements at analysis phase. Initially this technique converts the NL requirements in to some formal, controlled middle representation of software requirement such as Semantic Business Vocabulary and Rules (SBVR) Language (Standard introduced by OMG) to increase in accuracy of generated artefacts and models. Then it focuses on identifying the class model elements such a, classes, attributes, methods, relationships, multiplicity and many more to generate analysis phase class models. Finally this technique generates XML Metadata Interchange (XMI) Files to visualize generated models in UML modeling tool having XMI import feature.

Keywords: Class Diagram, POS Tagging, OOA, UML, XMI.

I. INTRODAUCTION

In software analysis phase of software development natural language are used to describe the exact business problem need to be solved. But the natural language are often complex, vague and ambiguous, sentences are vague when they contain generalization. Some time they are missing important information such as subject or object needed by the verb for completeness or contains pronouns. All these difficulty arise when any one discuss the business problem in using natural language. On other hand software requires more precision, correctness and cleanness that are not found in natural language. The main goal of Object Oriented Analysis (OOA) is to capture complete, exact and constant picture of the requirement of system and what system must do to satisfy the user requirement and needs. This is accomplished by constructing several models of system. In this process Software Requirement Specification (SRS) is created in natural language (NL). After that this natural language (Such as English) SRS are translated to the formal specifications such as UML models. This translation consists of generation of structural class model of the system [1].

However requirement (SRS) explained in NL can often uncertain, imperfect, and incoherent. It is usually work of requirement analyst to detect and repair potential ambiguities, discrepancy in such natural language SRS. But due to business analyst can fail to notice faults in SRS document in Natural language which can lead to multiple understanding and difficulties in recovering implicit requirements if analysts do not have enough domain knowledge. Thus evaluating requirements and generating the class model are massive and difficult task which need some automated support.

In last few years there are so many attempts has been made to transform the natural language business models in to platform independent models. It includes CM Builder, LIDA, GOOAL, LOLITA (NL-OOPS), NL-OOML, Event Extractor, Li, SUGER, and many more. But these tools are not used in real time software development process due there lesser accuracy and coverage in generating the formal models of system. Such tools produce 65 to 70 percentages of accuracy and coverage. The main reason behind failure of these tools is ambiguous and casual nature of natural languages. Also business model described in natural language are very complex to computational process due to inherent semantic inconsistencies in a natural language. A better solution to this problem is to convert the natural language business model in to some formal representation which is very simple for computation or machine process and also easy to understand by human being as natural language. Semantic Business Vocabulary and rule (SBVR) specification is the standard developed by Object Management Group fulfills this need.

This specification defines the vocabulary and rules for documenting the semantics of business vocabularies, business information, and business rules. This specification is applicable to the domain of business vocabularies and business rules of all kinds of business activities of all kinds of organizations.

Thus to analyze, extract and transform the hidden facts in natural language to some formal model has so many challenges and obstacles. To overcome some of these obstacles in software analysis there should be some mean or a technique which aims at to generate software artifacts to build the formal models such as UML class diagrams. Initially such technique should convert the NL business requirements in to some formal intermediate representation to increase in accuracy of generated artifacts and models. Then it focuses on identifying the various class model artifacts to generate analysis phase models. Finally this technique provide output in the format understood by model visualizing tool.

This paper proposes such approach to analyze, extract, transform and generate software artifacts from natural language business model to build the formal semantic models of the system. .

II. BASIC CONCEPTS

In this section, a brief introduction about the basic concepts of the OOA, UML Class Model, SBVR is provided

A. Object Oriented Analysis (OOA):

Analysis is concerned with devising a precise, concise, understandable model of the real world business. Object oriented analysis consist of identifying, extracting the needs of business and what system must do to satisfy the business requirements. The goal of object oriented analysis first is to understand the system's responsibilities by understanding how the user (Actor) use or will use the system. Next, the artifacts or elements (classes) that make up the system must be identified and their responsibilities and relationships among them. OOA concentrate on the describing what system does? Rather than the how it does it? This is accomplished by constructing the several models of the system from fuzzy set of system description such as use case model and class model. Use case model represent the user's view of the system or user's needs. Another activity in the OOA is to identify the classes and subparts such as attributes, methods and relationships among them in the system.[1]

B. UML Class Model:

The UML class model is the main static analysis model. This model shows the static structure of the system to be analyzed[1]. A class model is nothing but the collection of the static modeling artifacts such as classes and their relationships, and multiplicity among them connected as graph to each other and to their contents. The key element of class model is he classes and relationships among them. The class can have sub artifacts as attributes, and methods. Such model represents the mapping of objects in the real world to actual objects to be used in computer program.

C. SBVR:

SBVR is a short form of "Semantic Business Vocabulary and Rules" which has been introduced by Object Management Group (OMG) to reduce the gap between Business analyst and IT persons. [9] This is an contemporary an better way of capturing the business requirements in natural language like structure which is very easy to understand for human beings and also very simple to machine process due to its higher order of logic foundation. One can generate a business model of the system using the SBVR with the same communicative influence of standard natural language. In SBVR all specific expressions and definition of facts and concepts used by an organization in course of business are considered as vocabulary. Also in SBVR a formal presentation under the business influence are considered as rules which are used to express the operation of particular business entity under certain conditions[9].

III. BACKROUND AND COMPARATIVE ANANLYSIS

Many approaches and techniques have been proposed up till now to automate the process of various model generations from natural language requirement specification. However theses approaches are not used in real world system development due to their limitations in coverage and accuracy generation. Also majority of models concentrates on the class model only and require the high order of human interaction to complete the generated models

CM-Builder[2] aims at supporting the analysis stage of development in an Object-Oriented framework. CM-Builder uses robust Natural Language Processing techniques to analyze software requirements texts written in English and build an integrated discourse model of the processed text, represented in a Semantic Network. This Semantic Network is then used to automatically construct an initial UML Class Model. The initial model can be directly input to a graphical CASE tool for further refinements by a human analyst.

CM- Builder analyzes the requirements text and build initial class diagram only. This model can be visualized in graphical case tool by converting it into standard data interchange format where human analyst can make further refinements to generate final class model. Also CM-builder makes the extensive use of NLP techniques.

A Natural Language Object Oriented Production System (NL- OOPS) [3] generates object oriented analysis model from SemNet obtained by parsing NL SRS document. It considers noun as objects and identifies the relationships among objects using links. This approach lacks in accuracy in selecting the objects for large systems and cannot differentiate between class nouns and attribute nouns.

Linguistic assistant for Domain Analysis (LIDA)[4], provide linguistic assistance in the model development process. It presents a methodology to conceptual modeling through linguistic analysis. Then gives overview of LIDA's functionality and present its technical design and the functionality of its components. Finally, it presents an example of how LIDA is used in a conceptual modeling task.

This tool identifies model elements through assisted text analysis and validates by refining the text descriptions of the developing model. LIDA needs extensive user interaction while generating models because it identifies only a list of candidate nouns, verbs and adjectives, which need to be categorized into classes, attributes or operations based on user's domain knowledge.

"GOOAL" (Graphic Object Oriented Analysis Laboratory) [5] receives a natural language (NL) description of problem and produces the object models taking decisions sentence by sentence. The user realizes the consequences of the analysis of every sentence in real time. Unique features of this tool are the underlying methodology and the production of dynamic object models. GOOAL produces the class diagram by considering the validation threshold of 50% and its coverage accuracy (Precision matrices) is very minimum that is 78%

NL-OOML [6] presents an approach to extract the elements of the required system by subjecting its problem statement to object oriented analysis. This approach starts with assigning the parts of speech tags to each word in the given input document. The text thus tagged is restructured into a normalized subject-verb -object form. Further, to resolve the ambiguity posed by the pronouns, the pronoun resolutions are performed before normalizing the text. Finally the elements of the object-oriented system namely the classes, the attributes, methods and relationships between the classes, the use-cases and actors are identified by mapping the 'parts of speech-tagged' words of the natural language text onto the Object Oriented Modeling Language elements using mapping rules. But approximately 12.4 % of additional classes and 7.4 % of additional methods are identified in all the samples taken each of around 500 words. These additionally identified candidates are those that will usually be removed by human by intuition. Since the system lacks this knowledge, they were also listed as classes. Coverage accuracy is 82%

An evaluation methodology proposed by Hirschman and Thompson [8] is used for the performance evaluation of all above existing tools. According to this methodology the most enduring metrics of performance that have been applied to information extraction are termed as recall (Coverage of tool) and precision (Coverage Accuracy of tool).These metrics may be viewed as judging effectiveness from the application user's perspective. In the case of in information extraction, a correct output is a relevant fact.

Recall = no of Relevant-returned facts / actual relevant facts

Precision = no of relevant-returned facts / total no. of returned facts

Following TABLE I show the comparison of results of available tools that can perform automated or semi-automated analysis of the Natural Language requirement specifications. Recall value was not available for some of the tools.

Table I - A Comparison of Performance Evaluation of Existing Available Tools

Tools	***Recall Value***	***Precision Value***
CM-Builder (Harmain, 2003)	73.00%	66.00%
GOOAL (Perez-Gonzalez, 2002)	---	78.00%
NL-OOML (Anandha, 2006)	----	82.00%
LIDA (Overmyer, 2001)	71.32%	63.17%
Extract (only Event Extraction) (2009)	92.00%	85.00%

As the existing system uses natural languages as direct input to tool so that their recall and precision values are very less. Such results are there due to problems associated with Natural Languages such as

a. Ambiguous and informal nature
b. Inherent semantic inconsistencies
c. Complex to machine process.
d. Informal sentence structure

Moreover, the various functionalities supported by existing tools are also compared as shown in TABLE II

Table II – Functionality Support Comparison of Existing Available System

Tool Functionality	***NL-OOPS***	***CM-Builder***	***LIDA***	***GOOAL***	***NL-OOML***	***Extract***
Classes	***YES***	***YES***	***YES***	***YES***	***YES***	***YES***
Attributes	***YES***	***YES***	***YES***	***YES***	***YES***	***YES***
Methods	***YES***	***YES***	***YES***	***YES***	***YES***	***YES***
Associations	***YES***	***YES***	***YES***	***In Semi NL***	***NO***	***NO***
Multiplicity	***NO***	***YES***	***YES***	***NO***	***NO***	***NO***
Aggregation	***NO***	***YES***	***NO***	***NO***	***NO***	***NO***
Generalization	***YES***	***NO***	***NO***	***NO***	***NO***	***NO***
XMI Support	***NO***	***YES***	***NO***	***NO***	***NO***	***NO***
Normalize Requirement	***NO***	***YES***	***NO***	***NO***	***NO***	***NO***
User Interaction	***High***	***Medium***	***High***	***High***	***High***	***Low***

Table II shows that there are very few tools those can extract information such as multiplicity, aggregations, generalizations, instances from Natural language requirement. None of the tool generates the OCL representation of output model and thus they are inadequate to capture the non-functional requirements.

As every event itself is a single interaction i.e. atomic unit of interaction (Below Use-case) that captures functional requirements in terms of an interaction from a scenario. Proposed system supports the functionality to capture the events from requirement statement and generates Event Meta Model/Template to capture the event from natural language requirement

Thus, the results of this initial performance evaluation show that there is still potential for automation and provide motivation for approach proposed here.

IV. PRPOSED SYSTEM METHODOLOGY

This section describes the used methodology to identify the artifacts which are used to generate the models at analysis phase from natural language. This methodology consist of automatic conversion of natural language software requirement specification conversion to controlled intermediate SBVR format and secondly to identification of software artifacts and model generation, finally visualization of generated models. Used methodology works in different phases organized in pipelined fashion as follows.

a. Preprocess Analysis:

This phase stars with the by reading the given English input and tokenizing the whole input in to individual tokens. To do so java tokenizer class [11] is used. After tokenizing each token is stored in separate array list. While tokenizing the

English input sentence splitter is used to identify the boundary of each sentence.

b. Tagging :

This processed text is further given as input to Part Of Speech (POS) tagger [10] to identify the basic POS tags. To do so Standard POS tagger is used which identifies the 44 basic POS tags.

c. Morphological Analysis :

To remove the suffixes attached to noun phrases and verb phrases this type of analysis is performed on the tagged output from pervious phase. In this type of analysis WordNet [13] is used to convert the plural into singular form also suffixes attached to verb phrases such as "ed" are also removed.

d. Pronoun Resolution :

In this phase JavaRAP [12] is used to replace all possible pronouns with correct noun form up to third person.

e. Role Labeling and Element/Concept Identification:

In this phase role labels are identified from preprocessed text such as performer, co actors, events, objects and receiver in the sentences. Also in this phase SBVR concept identification is done according to some identification constraints such as all proper nouns are identified to individual concepts, all common nouns are identified as noun concepts or object type, all action verbs are identified as verb concepts, all auxiliary verbs are identified as fact types, possessed nouns are identified as characteristics or attributes, indistinct articles, plural nouns and cardinal numbers are identified as quantification. Output of this phase is stored in an array list.

f. Rule Generation:

To generate the SBVR rule we have to first produce fact types, in the form of sentences which represents some relationships between the concepts identified in the previous phase. For that purpose use the template such as noun-verb-noun to establish the relationship between two concepts. Thus a fact type is created by combining the noun concepts and verb concepts from pervious phase array list. Generated fact type is used to create the SBVR rule by applying various logical formulations such as use of logical expression AND, OR and NOT etc, Quantification token conversion rules, possibility and obligation formulation rules are used.

g. Applying Notations:

In this phase SBVR notations are applied to generate rules such as noun concepts are underlined, verb concepts are italicized, keywords are bolded, individual concepts or attributes are double underlined[9].

h. Artifacts Extraction:

In this phase produced SBVR vocabulary and rules are further processed to extract the basic building blocks or artifacts of class models. All SBVR noun concepts and object type are tends to be classes for class model. All verb concepts associated to noun concepts are tend to be methods for the class model. All SBVR characteristics associated with the noun concepts and object type are mapped to the data items in class model. SBVR Quantifications identified with respective noun concepts are mapped to the multiplicity between two classes.

i. XMI Generation and Model Visualization:

Finally in this phase the output of above phases are generated in the form of XML metadata interchange (XMI) file format. Such file is further given as input to UML modeling tool having XMI import feature to visualize the generated models.

Following fig1 shows the process architecture of the proposed methodology.

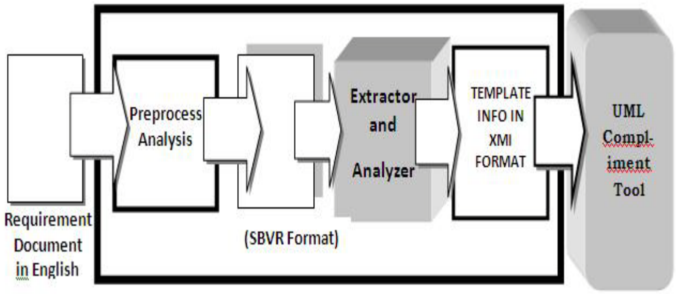

Figure 1 Process System Architecture

V. SYSTEM WORKFLOW

Take the input from user, a document which is written in English like natural language. Then we do the preprocessing of this natural language specification document using different natural language processing tools and technologies to do tagging, morphological analysis, pronoun resolution and parse tree generation. After doing so this preprocessed text is converted into controlled intermediate format such as SBVR Concepts and Rules which are then mapped to software artifacts to build the models at analysis level. These models are finally visualized using the UML compliment tool having XMI import features. Following fig2 shows the sequenced steps to be followed to generate the analysis level models of software specification.

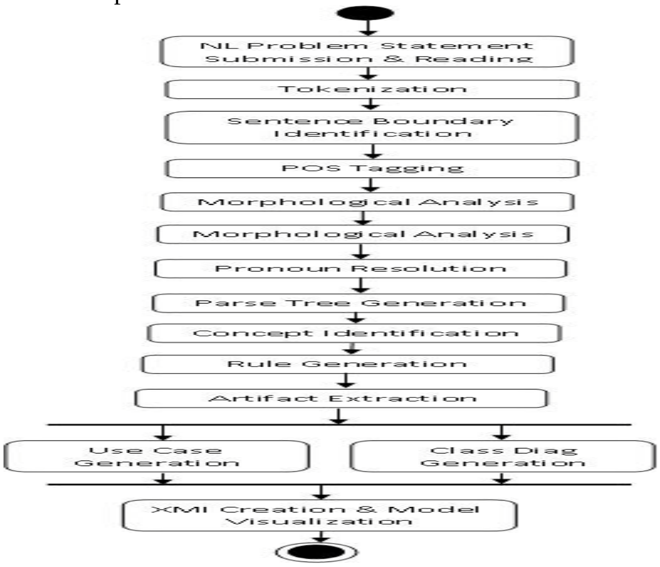

Figure 2 Workflow of System

VI. CONCLUSION

This approach describes a computerized way to take out the software element at analysis phase. It uses Natural Language Processing techniques to consider business level software requirements and builds an incorporated analysis level class model.

This approach can be used for the identification of software elements such as classes, their attributes, and the static relationships among them with increase in accuracy due to use of intermediate format SBVR.

VII. REFERENCES

[1] Ali Bahrami, Chapter 6, Object Oriented Analysis Process, in Object Oriented System Development.

[2] H. M. Harmain and R. Gaizauskas, CM-Builder: An Automated NL Based CASE tool, in IEEE International Conference on automated software engineering (2000)

[3] Mich L., NL-OOPS: From natural language to object oriented requirement using natural language processing system (1996)

[4] Overmyer, S. P., Benoit, L. and Owen R., Conceptual modeling through linguistic analysis using LIDA. International Conference of Software Engineering (ICSE), (2001)

[5] Hector G perez-Gonzalez and Jugal K. Kalita, GOOAL : A Graphical Object Oriented Analysis laboratory, ACM 1-58113-626-9/02/0011 (2002)

[6] G.S. Anandha Mala, J. Jayaradika, and G. V. Uma, Restructuring Natrual Language Text to Elicit Software Requirements, in proceeding of the International Conference on Cognition and Recognition (2006)

[7] Sanddep K. Singh, Reetesh Gupta, Sangeeta Sabharwal, and J.P. Gupta, E-xtract : A tool for extraction, Analysis and Classification of Events from Textual Requirements, in IEEE 2009 international Conference on Advances in Recent technologies in communication and Computing.

[8] Hirschman L., and Thompson, H.S. 1995. Chapter 13 Evaluation: Overview of evaluation in speech and natural language processing. In Survey of the State of the Art in Human Language Technology.

[9] OMG. 2008. Semantics of Business vocabulary and Rules. (SBVR) Standard v.1.0.

[10] Stanford Log-linear Part-Of-Speech Tagger: The Stanford Natural Language Processing Group

[11] Java Primer StringTokenizer Class by Scott MacKenzie

[12] JavaRAP, last accessed 2nd December, 2010,

http://aye.comp.nus.edu.sg/~qiu/NLPTools/JavaRAP.html,.

[13] WordNet 2.1, last update http://wordnet.princeton.edu/wordnet/, 27th October, 2010