Automatic Construction of Object Oriented Design Models [UML Diagrams] from Natural Language Requirements Specification

G.S. Anandha Mala[1] and G.V. Uma[2]

Department of Computer Science and Engineering, College of Engineering,
Anna University, Guindy, Chennai, Tamil Nadu, India-600025
malamanosuke@yahoo.co.in, gvuma@annauniv.edu

Abstract. Application of natural language understanding to requirements gathering remains a field that has only limited explorations so far. This paper presents an approach to extract the object oriented elements of the required system. This approach starts with assigning the parts of speech tags to each word in the given input document. 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, sequence of actions, the use-cases and actors are identified by mapping the 'parts of speech- tagged' words onto the Object Oriented Modeling Language elements using mapping rules which is the key to a successful implementation of user requirements.

1 Introduction

As already several attempts have been made to semi automate the process of requirements capture, this is yet another approach of automatic construction of object oriented design model [UML diagram] from the natural language requirement specification. The paper begins with a review of the advances in the field of requirements engineering in section 2. The proposed methodology is explained in section 3. Our implementation and results are explained in section 4. The conclusion and future work is contained in Section 5.

2 Related Work

The first relevant published technique attempting to produce a systematic procedure to produce design models from NL requirements was Abbot [1]. Abbot suggested a non-automatic methodology that only produces static analysis and design products obtained by an informal technique requiring high participation with that of users for decisions**.** Methods to bring out a justified relationship between the natural- language structures and OO concept is proposed by Sylvain [9] who show that computational linguistic tools, are appropriate for preliminary computer assisted OO analysis. Sawyer in their REVERE [5] makes use of a lexicon to disambiguate the word senses thus obtaining a summary of requirements from a natural language text but do not

Q. Yang and G. Webb (Eds.): PRICAI 2006, LNAI 4099, pp. 1155 – 1159, 2006.

attempt to model the system. Liwu Li [6] also presents a semi-automatic approach to translate a use case to a sequence diagram. It needs to normalize a use case manually. Overmyer [8], also present only a complete interactive methodology and prototype. However, the text analysis remains in good part a manual process. Liu [3] present an approach, which uses formalized use cases to capture and record requirements. Ke Li [4] also semi-automate the process of requirement elicitation where the text is matched with predefined statements. If there is no match then get help from user to clarify incomplete/ambiguous data. Participation of domain experts, customer are needed in class identification process in contrast to our fully automatic methodology.

3 The Proposed System

In all the earlier works mentioned, the requirement elicitations are not fully automatic. The proposed methodology includes the automatic reference resolution, which eliminates the user intervention as in the previous works. The system architecture is shown in fig. 1. The system named as 'REQUIREMENTS ELICITOR'. The given input problem statement is split into sentences by the sentence splitter for sentence tagging. Then each sentence is subjected to tagging in order to get the parts of speech marker for every word. The noun and the verb phrases are identified for the tagged text by chunker based on simple phrasal grammars. To remove ambiguity posed by pronouns, they are resolved to their respective noun phrases by reference resolver. The text has to be simplified into the following constructs by the normalizer to ease the task of mapping the words onto the Object Oriented system constituents.

- Conditional: Conditional syntax is *If* aCondition transaction [*else* other Transtions]
- Iteration: Iteration syntax is *While* condition transactions *endwhile.*
- Concurrency: Concurrency syntax is

  Start *concurrency* transaction 1 … *concurrent* transaction k
  end *concurrency*

  which executes transaction l, to transaction k concurrently.
- (Synchronization) A synchronization syntax is

  Start *Synchronization* transaction 1 … *synchronized* transaction k
  end *Synchronization*

Which requires synchronize transaction l, to transaction k.

All the transaction statements are simple. A number of patterns using conjunctions and their corresponding splits in the sentences are stored in the catalog. Each sentence is checked against the stored patterns and the corresponding split up is made.

For example "If the source and the destination of the request fall on the same route, the receptionist checks the seat that are available and issues the ticket to the passenger and blocks the seat" is normalized to

If the source and the destination of the request fall on the same route
    The receptionist checks the seat.
    The receptionist issues the ticket to the passenger
    The receptionist blocks the seat
End if

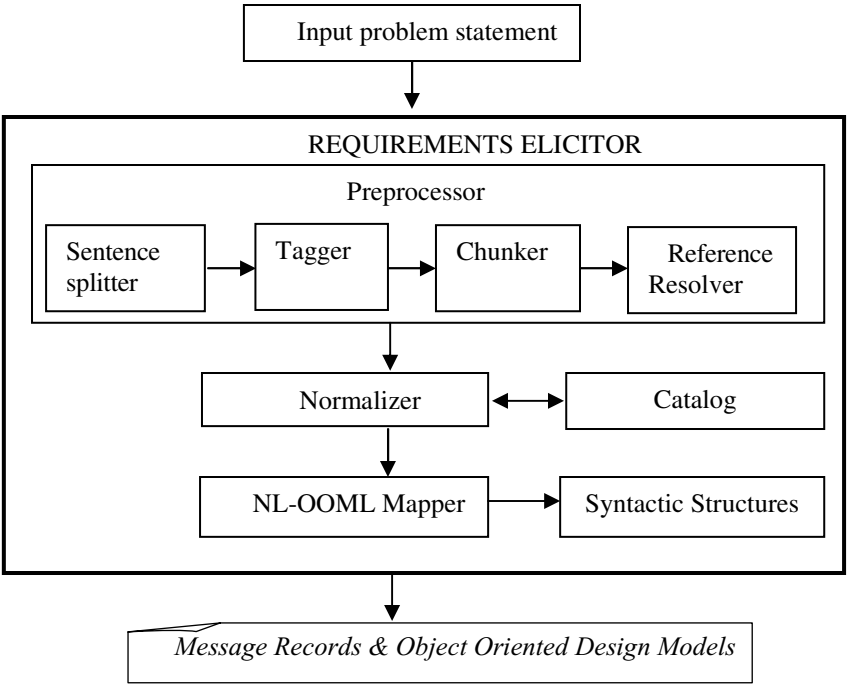

Fig. 1. System Architecture

The NL-OOML mapper accepts a normalized problem description as input. Using the syntactic structures as in table 1. it translates each normalized sentence into a message record. Simple rule based approach is followed for identifying OO elements.The rules are,

1: Translating Nouns to Classes. A noun, which does not have any attributes, need not be proposed as a class.
2: Translating *Noun-Noun to Class-Property* according to position. When two nouns appear in sequence in the text, the first Noun is translated to Class and the following Noun is translated to properties of this Class.
3: A simple heuristic is used to decide which nouns are classes, and which form the attribute. In Noun-Noun, if the first noun is already been chosen as the class then the second noun is taken as the attribute. The attributes are decided based on the verb phrase.
4: Translating the lexical Verb of a non-personal noun to a Method of this noun. Decide the sender, receiver classes and argument to this method based on the Table 1.
5: Translating the lexical Verb of a personal noun to a use case (or part of a use case) linked with an actor defined by this noun.
6: Matching a Noun to a Personal Pronoun as the nouns of previous sentence.

Table 1. Syntactic Structures of Simple Sentences

Syntactic Structure	Sender	Receiver	Action	Argument
subject verb object	subject	object	verb	-
subject verb object (to) verb1 (object1)	subject	object	verb1 (+object1)	(object1)
subject verb object participle (object1)	subject	object	participle verb(+object1)	(object1)
subject verb object adjective	subject	object	be + adjective	
subject verb object conjunctive to verb1 (object1)		subject	verb	object,verb1 (+object1)
subject verb gerund (object)		subject	verb	gerund verb (+object)
subject verb object preposition object1	subject	object1	verb + object	(object)
subject verb object object1	subject	object	verb	object1
subject verb (for) complement		subject	verb	complement
subject verb		subject	verb	
subject be predicative		subject	be + predicative	
subject verb preposition object		subject	verb+preposition	object

4 Implementation and Results

The 'Requirements Elicitor' was implemented using JAVA and validated using 100 problem samples each of around 500 lines. The result produced by the system was compared with that of the human output. The human outputs were the results that were obtained by conducting the noun-verb analysis on the text. It was considered as the baseline and taken as expert judgment. The system does not miss to identify any of the classes and methods. But approximately 12.4% of additional classes and 7.4% of additional methods are identified in the entire sample taken, those that are removed by human by intuition that they may not be classes. Since system lacks that knowledge, they are listed as classes. The missed out methods occur only if the tagger assigns a wrong tag to the word. Also the system perfectly identifies all the attributes, usecases and actors with out any additional, missed or miss assignments.

5 Conclusion and Future Work

The project presents an approach to restructure the natural language text into a modelling language in order to elicit the stated requirements of a system. Further the work can be extended for identifying the different modules present in the requirement specification by properly segmenting the input text, which will help us to identify the packages. The deficiencies in the tagger and the reference resolver can be overcome by building a knowledge base which can also improve the effectiveness of generation of the system elements.

References

1. Abbot.R.J.: "Program Design by informal English descriptions". *Communications of the ACM*, vol.26, (1983) 882 – 894.
2. Brill E.: "A simple rule-based part-of-speech tagger". *Proceedings of Third ACL Conference on Applied Natural Language Processing*, Trento, Italy, (1992) 152-155
3. Dong Liu, Kalaivani Subramaniam, Behrouz H. Far, Armin Eberlein: "Automating transition from use cases to class model", *MSc Thesis*, University of Calgary, (2003).
4. Ke Li: "Towards Semi-automation in Requirements Elicitation: mapping natural language and object-oriented concepts", *13th IEEE International Requirements Engineering Conference,* (2005)
5. Sawyer P., P Rayson, and R Garside: "REVERE: support for requirements synthesis from documents", *Information Systems Frontiers Journal*. Vol.4, (2002) 343 - 353.
6. Liwu Li: "A semi-automatic approach to translating use cases to sequence diagrams", *Proceedings of Technology of Object-Oriented Languages and Systems*, July (1999), IEEE CS Press, 184 –193
7. Mitkov. R: "Robust pronoun resolution with limited knowledge", Proceedings of the 18.th International Conference on Computational Linguistics (COLING'98)/ACL'98", Montreal, Canada, (1998) 869-875.
8. Overmyer ScottP., Lavoie.B.Rambow: "Conceptual modelling through linguistic analysis using LIDA", *Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001*, Toronto (2001).
9. Sylvain Delisle, Ken Barker, Ismaïl Biskri: "Object-Oriented Analysis: Getting Help from Robust Computational Linguistic Tools, in G. Friedl, H.C. Mayr (eds) *Application of Natural Language to Information Systems*, Oesterreichische Computer Gesellschaft, (1999) 167-172.