An Approach towards Automation of Requirements Analysis

Vinay S, Shridhar Aithal, Prashanth Desai

***Abstract*-Application of Natural Language processing to requirements gathering to facilitate automation has only limited explorations so far. This paper describes a Natural Language based tool which aims at supporting the analysis stage of software development in an object oriented framework. This paper is built on the foundation of existing mappings between natural language elements and Object oriented concepts. The tool named R-TOOL analyses software elicited requirements texts written in English to generate actors, use cases, classes, attributes, methods and relationship between the classes leading to the generation of class diagrams. This paper discusses initial experimental results which are encouraging and outlines further research plan to help to improve the system which will have the potential to play an important role in the software development process.**

Index Words- Object oriented technology, Natural language processing, Requirements Engineering, Use case.

I. INTRODUCTION

Requirements engineering (RE) is concerned with the identification of the goals to be achieved by the envisioned system, the operationalization of such goals into services and constraints, and the assignment of responsibilities for the resulting requirements to agents such as humans, devices, and software. The processes involved in RE include domain analysis, elicitation, specification, assessment, negotiation, documentation, and evolution. Getting high quality requirements is difficult and critical. Recent surveys have confirmed the growing recognition of RE as an area of utmost importance in software engineering research and practice.

Manuscript received Jan 01st, 2009.

Vinay S, Senior Lecturer, N.M.A.M Institute of Technology, Nitte, India and research scholar at MIT, Manipal, India.(Ph:+919986515835, E-mail:vinaymanyan@gmail.com)

Sridhar Aithal is currently guiding research scholars at Manipal University, Manipal. (E-mail: drsaithal@gmail.com)

Prashanth Desai, PG student at Dept of CSE, N.M.A.M.I.T, Nitte, India. (E-mail: prashanth_desai@yahoo.com)

Object-Oriented Technology (OOT) has become a popular approach for building software systems. Many object oriented methods have been proposed and in these methods, Object-Oriented Analysis process is considered one of the most critical and difficult task [1]. It is critical because subsequent stages rely on Object Analysis and it is difficult because most of the input to this process is in the form of natural language English which is inherently ambiguous [2].

During the analysis phase of software development, requirements analyst interview clients about system process, gather data and write a description in English of the system under development. A graphical Computer Aided Software Engineering (CASE) tool is typically used to document the output of the analysis. Such a tool helps developers assess whether the software requirements Specification (SRS) contains any inconsistency or incompleteness that might negatively impact subsequent object modeling which is one of the most crucial and difficult task in software engineering.

The Artificial Intelligence (AI) subfield of Natural Language Processing (NLP) suggests approaches which may assist software engineers in the analysis of software development [2]. Many researchers have begun to see potential benefits from adding natural language processing (NLP) capabilities to CASE tools. The objective is to create an NLP module that helps automatically identify classes, attributes, methods and relationships implied in the software requirements specification [3].

In this paper we describe an attempt towards automation of use case driven requirements analysis. Section 2 describes the existing NLP based CASE systems and analyses its strengths and weakness. Section 3 deals with our approach towards automation of object-oriented systems and its implementation and section 4 discusses the results of R-TOOL by taking an ATM system as case study.

Section 5 describes evaluation methodology of our system followed by conclusions and future work.

II. RELATED WORK

We present a brief survey of existing NLP based approaches for providing automated tools to support analysis and design phase of software development.

Abbot [4] proposed a technique attempting to produce a systematic procedure to produce design models from NL requirements. It produced static analysis and design modules which required high user intervention for making decisions.

Saeki et. Al. [5] described a process of incrementally constructing software modules from object-oriented specifications obtained from NL requirements. Nouns were considered as classes and their corresponding verbs as methods. These were automatically extracted from the informal descriptions but the importance of the words under the given context was not given adequate importance for the construction of the formal specification.

NL-OOPS [6] and CM-BUILDER [2] directed at construction of object oriented analysis models from natural language specifications. But the major hindrance is the informal nature of natural language where in the input descriptions often lack preciseness, completeness and consistency. Hence the output is only an initial Object Oriented model which necessitates further communication with stake holders to resolve ambiguities [7].

An approach to write software specifications from a controlled subset of a natural language was undertaken by ASPIN [8]. Controlled language approach imposes restrictions on the authors of software requirements documents as they must learn and use a specialized language controlled [2].

REVERE [9] makes use of a lexicon to clarify the word senses. It obtains a summary of requirements from a natural language text but do not attempt to model the system.

We can draw the following inference from the survey of the related work. A completely automated tool that aims to replace the analyst is unlikely in the near future given the present state of the language processing technology. A tool can assist the analyst by making proposals in an effective manner. Without the participation of stakeholders such NLP based systems will not make the desired impact on software development.

In this paper, we outline our approach to the problem in the next section which is mainly use case driven and discuss initial results obtained from R-TOOL.

III. APPROACH OF R-TOOL

The goal of object-oriented analysis is to understand the domain of the problem and the system's responsibilities by understanding how the users use or will use the system. The object oriented analysis phase of software development is concerned with determining the system requirements and identifying classes and their relationship to other classes in the problem domain. Ivar Jacobson [10] came up with the concept of Use case, his name for a scenario to describe the user-computer system interaction. Thus use case became the driving point for gathering requirements in an object oriented way.

The R-TOOL NLP based CASE tool takes a requirements elicited document as input and produces the elements of object oriented systems namely classes, attributes, methods and relationships between classes leading to the generation of the class diagram as output. Our approach draws inspiration from [2] and [7]. The basic block diagram of NLP based R-TOOL is shown in fig 1.

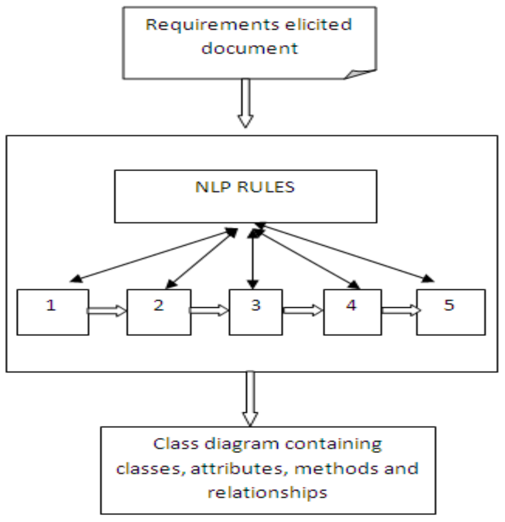
Requirements elicited document
NLP RULES
1
2
3
4
5
Class diagram containing classes, attributes, methods and relationships

Fig 1: R-TOOL Block Diagram

The basic steps in R-TOOL can be summarized as follows.

- The input to R-TOOL is a problem description of the application to be developed in English
- NLP rules are used to syntactically and semantically analyze the input document
- Produce a class diagram comprising classes, attributes, methods and relationships between classes.

3.1 The Elicited Input Document

R-TOOL takes a plain test file containing the elicited requirements written in English. We impose no restrictions on the input document.

3.2 R-TOOL NLP System

It includes five major processing steps numbered 1 to 5 in the block diagram shown in fig 1.

1.Tokenizer: The tokenizer splits a plain text file into tokens. This includes separating words, identifying numbers.

2. Pronoun Resolver: The presence of pronouns poses difficulty in identifying actors and use cases. This ambiguity is resolved by scanning the input document for pronouns and replacing the pronoun with the noun or the subject in the previous sentence.

Consider the sentence *Bank Manager takes the daily stock of money available in the ATM. He is responsible for loading the money into the ATM.*

While scanning the word he creates ambiguity. This is identified with the pattern *missing noun* in the sentence. He is then substituted with the noun in the previous sentence which in this case is Manager. Alternatively the system raises a question, ***who** is responsible for loading the money into the ATM?* In this way pronoun ambiguity is resolved by R-TOOL.

3. Identify Actors and Use cases: Nouns in the input document become candidate actors. The list of candidate actors is pruned by the frequency of occurrence and the final list of actors is obtained.

The input document is scanned again by looking for each actor and its role. The associated verb part of the actor becomes the candidate Use case. This process is repeated for all the actors identified leading to identification of all the use cases.

4. Identifying Responsibilities and generating use case report: Once a use case is identified, the responsibilities and the descriptions of that use case are determined by using keyword based search in the input document.

This keyword based search is performed by taking the root word. Consider an identified Use case *Withdraw money*. We need to identify responsibilities or functionalities of this Use case. We scan the input document for the keyword withdraw. For example, a verb form like "*Withdrawing*" will be analyzed as "withdraw + ing". The document is scanned for withdraw keyword and the corresponding sentence becomes the responsibility or describes the functionality of that use case. For example, a sentence specifying the conditions of withdrawal now becomes part of the Withdraw money use case.

All the identified use cases along with its functionality are generated to form a Use case report. This Use case report is then fed into Classifier.

5. Classifier: The input to Classifier is the generated Use case report. The processing steps of Classifier can be summarized as follows.

i) For every identified class find its frequency in the text (i.e. how many times it is mentioned) The most frequent candidates suggest a class. Redundant classes, adjective classes are eliminated. A statement of purpose is identified for each of the class identified.

ii) A simple set of rule is used to find out 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.

iii) A noun, which does not have any attributes, need not be proposed as a class.

iv) Attributes can be found using some simple heuristics like the possessive relationships and use of the verbs *to have, denote, identify.* Attributes also correspond to nouns followed by prepositional phrase such as *cost of* the soup.

v) Relationships between classes can be of three types: Association, Aggregation and Generalization.

A dependency between two or more classes may be an association. Association is corresponds to a verb or prepositional phrase such as " part–of ", " next–to ", " father–of ", "works–for ", " contained–in ". For example, the sequence *Client has an account,* matches the pattern noun-verb-noun. Client and account being the class, *has* becomes the association. Determiners are used to identify the multiplicity of roles in an association.

Generalization: A Top down approach is followed looking for noun phrases composed of various adjectives in a class name. For example, consider the sentence *A client can have a savings account and a checking account.* It denotes a case for inheritance with account being the base class and two types of accounts being the sub class.

Aggregation: Sentence pattern such as *something contains something, something is part of something, something is made up of something* denote aggregation relationships.

3.3 Implementation

The R-TOOL software is developed using open source technologies. JAVA SWING and MySQL are used for developing R-TOOL. Swing is a graphical user interface (GUI) toolkit for Java.

IV. A CASE STUDY

We have taken the elicited requirements document of bank ATM system as a case study. We then compare the performance of our system with that of result obtained manually in this section. The extract of the description of bank ATM requirements is as follows:

4.1 Extract of the Problem Statement

The bank client must be able to deposit an amount to and withdraw an amount from his or her accounts using the bank application. Each transaction must be recorded, and the client must have the ability to review all transactions performed against a given account. Recorded transactions must include the date, time, transaction type, amount and account balance after the transaction.

A bank client can have two types of accounts. A checking-account and a saving-account. For each checking account, one related saving-account can exists. The application must verify that a client can gain access to his or her account by identification via a personal identification number (PIN) code.

Neither a checking-account nor a saving-account can have a negative balance. The application should automatically withdraw funds from a related saving-account if the requested withdrawal amount on the checking-account is more than its current balance. If the saving-account balance is insufficient to cover the requested withdrawal amount, the application should inform the user and terminate the transaction.

4.2 Comparison of results

The following tables compare the result obtained for manual and automated analysis approach.

Table I: Identifying actors

Manual Result	Automated Result
Bank client ATM-card ATM-machine System	Bank Card Customer Machine System

Table 2: Identifying use cases

Manual Result	Automated Result
Bank ATM transaction Approval process Deposit amount Deposit savings Deposit checking Withdraw amount	Make cash withdrawal Make deposit Transfer money between account Make balance enquiry

Table 3: Identifying classes

Manual Result	Automated Result
ATM machine Bank Client Bank Account Saving-account Checking-account Transaction	Machine Customer Bank Account Savings-account Checking-account Transaction System

4.3 Attributes, Methods and Relationships

i) *Class:* Machine, *Attributes:* address

ii) *Class:* Customer, *Attributes:* Name, Card Number, PIN number, *Methods:* Verify password

iii) *Class:* Bank

iv) *Class:* Account, *Attributes:* number, balance, *Methods:* Withdraw, Deposit, Transfer

v) *Class:* Savings account

vi) *Class:* Checking account

vii) *Class:* Transaction, *Attributes:* date, time, type, balance, amount

Aggregation: Bank class is an aggregation of account and Machine class.

Account is the base class and Checking and Savings account are its derived class.

A customer can have 1 or 2 account depicts association between customer and account along with its multiplicity.

4.4 Use case responsibilities or description

Consider the use case *Withdraw Money*. The responsibility or the functionality of this use case identified by R-TOOL is as follows:

The bank client must be able to deposit an amount to and withdraw an amount from his or her accounts using the bank application. The application should automatically withdraw funds from a related saving-account if the requested withdrawal amount on the checking-account is more than its current balance. If the saving-account balance is insufficient to cover the requested withdrawal amount, the application should inform the user and terminate the transaction.

4.5 Discussion

By comparing the results, we can infer the following:

- The system is able to identify actors, use cases, classes satisfactorily.
- The use cases obtained in the manual approach are more in number after applying the concepts of include and extend association.
- We also have made use of the concept of repository, which keeps track of classes obtained in previous projects and when a class in the present working project is similar to an existing class in the repository, the system not only displays the particular class information but also provides an option to add certain or all the information about the class in the present working project.

V. EVALUAITON OF R-TOOL

Generating requirements using NLP based approach is an active emerging research area. R-TOOL differs from the existing tools by focusing on ensuring that use cases and its responsibilities are clearly identified which then makes the task of identifying classes, attributes and methods much easier.

The R-TOOL identifies few irrelevant classes during analysis. This drawback can be overcome by imposing following constraints in the input document or developing NL rules to overcome following constraints.

- The sentence must be in active voice.
- Compound sentence must be split into two simple sentences rather than joining them using a conjunction

We can infer the following benefits from R-TOOL

- R-TOOL can supplement the manual approach and serve as a useful tool in identifying inconsistencies between manual approach and automated approach, there by making sure that system requirements are identified properly.
- Reusability is ensured by maintaining Repository of classes of different projects. The user can search for a particular class and can tailor the class identified from the repository to his needs.

VI. CONCLUSION AND SCOPE FOR FUTURE WORK

Using NLP to generate correct requirements is a difficult task considering the inherent ambiguity in natural language. Identifying effective use cases is the key to generating complete list of classes eliminating irrelevant classes.

The R-TOOL system developed using open source technologies Java and MySQL is under constant improvement and future enhancements are being carried out in the following areas.

- Providing support for creation of different UML diagrams.
- Identifying goals (higher level strategic objectives of a system) from the elicited documents and linking the identified goals with Use cases. Mapping of use cases to goals helps in ensuring that the requirements are complete and provides requirements traceability.

- Using efficient algorithms in NLP to reduce generation of unnecessary classes.
- Providing a comprehensive evaluation methodology to qualitatively evaluate the effectiveness of NLP tools.

REFERENCES

[1] G. Booch, *Object-Oriented Analysis and Design with applications,* The BC publishing company Inc., second edition, 1994

[2] Harmain, H.M. and Gaizauskas R. "*CM –Builder: An Automated NLP-based CASE Tool*", The Fifteenth IEEE International Conference on Automated Software Engineering, 2000.

[3] Generating Clas Models through Controlled Requirements, Reynaldo Giganto, NZCSRSC 2008 April, Christchurch, New Zealand.

[4] Abbot R J, *"Program design by informal English description"*, ACM Vol 26, 1983, 882-894

[5] Saeki, M., Horai, H., Toyama K., Uematsu, N., and Enomoto H. *"Specification framework based on natural language"*, In Proc. of the 4th Int'l Workshop on Software Specification and Design, 1987, pp. 87-94.

[6] Mich, L. and Garigliano R. "NL-OOPS: A Requirements Analysis tool based on Natural Language Processing". In the Proceedings of Conference on Data Mining 2002, Vol. 3, pp. 321- 330, Southampton, UK:WIT Press.

[7] K Li, R G Dewar, RJ Pooley, *"Computer-assisted and Customer-oriented Requirements elicitation"*, 13th IEEE International Conference on Requirements Engineering, 2005

[8] W Cyre, *"A requirements sublanguage for automatic analysis"*, International conference of Intelligent systems, 10(1), 665-689, 1995

[9] Sawyer P, Rayson, Garside, *"REVERE: support for requirements synthesis from documents"*, Information systems Frontiers Journal, Vol 4, 2002, 343-353

[10] Jacobson, I., Booch G., Rumbaugh, J. *The Unified Software Development Process,* Addison-Wesley, USA.1999. pp 135