REQUIREMENT ANALYSIS USING NATURAL LANGUAGE PROCESSING

ABINASH TRIPATHY, ANKIT AGRAWAL AND SANTANU KUMAR RATH
Department of Computer Science and Engineering, National Institute of Technology, Rourkela, Odisha, India
Email: abi.tripathy@gmail.com; agrawala96@gmail.com; skrath@nitrkl.ac.in

ABSTRACT: Software Requirement Specification (SRS) is the formal document by which the customers share their requirements with the development team. SRS is written in any of the natural language (NL). The text written in SRS is observed incomplete and ambiguous in many cases. From the incomplete and ambiguous SRS, the requirement analyst team, make an intelligent analysis with its detail information using Natural Language processing (NLP) techniques. This paper proposes an approach to help the analysis phase, particularly conducting object oriented (OO) analysis by generating class diagram and its details from SRS, in an automated manner. A standard case study of ATM operation in Bank is considered for the model creation and evaluation.

KEYWORDS: Software Requirement Specification (SRS), Natural Language (NL), Natural Language Processing (NLP)

INTRODUCTION

Software Development Life Cycle (SDLC) starts with eliciting the user's requirement in the form of Software Requirement Specification (SRS). As customers use their own language to state their requirements, it is observed to be very often ambiguous and confusing for the development team to analyze. In order to help the development team, natural language processing (NLP) concepts are found to be helpful. The NLP system processes the data written in natural language in an intelligent manner and generate an information which is comparatively easier for the development team to analyze.

NLP deals with the combined efforts of Computer Science and Linguistics. NLP is concerned with how the computer and human language interact with each other and help to obtain a useful result [1]. This study intends to explore NLP applications for object-oriented (OO) software development methodology where object and associated class is the basic unit. Unified Modeling Language (UML) is used for the diagrammatic representation of the concepts of OO development system [2]. XML which stands for eXtensible Markup Language, is used for storing and data exchange. In order to display the stored data of XML, HTML is used. The structure of the XML document is described in XML Schema Definition (XSD). XSD represents the XML document in a tree structure with root and its child elements. XSD supports data types and namespaces which help to identify the uniqueness of XML documents.

During the course of the paper, an attempt has been made to analyze the SRS document written in NL and generate the class diagram with its data members and member functions. While generating the class diagram, the candidate for the class and it details are obtained using NLP techniques. The

obtained details are then transformed to XML. XSD is generated from XML and using reverse engineering process class diagram is generated from XSD.

The structure of the paper is defined as follows: Section 2 gives literature survey. Section 3, presents the methodology about NLP and its different approaches. In Section 4, the proposed approach is further explained. Section 5 describe about its implementation. Section 6 provides an empirical evaluation and a comparative study of the proposed approach. Finally, Section 7 concludes the paper and presents the scope of the future work.

LITERATURE SURVEY

The object oriented application of NLP was initiated by Abbott, where he proposed a method based on linguistic analysis of informal strategies written in English [3]. His proposed method consists of following three steps

i. Developing informal strategy using Natural Language.

ii. Formalizing it by identifying data types, objects and operators.

iii. Isolating the solution into two parts. Such as package that contains formalization of the problem domain, and subprogram(s) containing steps for solving particular problem.

Congruent with the Object-Oriented approach, his work focused on the use of nouns and noun-phrase as a reference in NL. According to Abbott, few NL elements refer to object-oriented elements, i.e., common nouns refer to data type, proper nouns and references refer to objects, verbs, predicate and descriptive expression refer to operators, control structure of English language refer to its object-oriented equivalent.

Saeki et al. suggested the development of OO software design using NL concepts [4]. They proposed a step-by-step method of derivation of formal specification from informal specifications written in NL. Their process consists of two steps, i.e., design and elaborate. The informal English description is used during design activity to extract the structure of the software model based on the OO model. The elaborating activity consists of refining and rewriting the informal specification based on derived module design document. Finally the conversion is achieved through *elaborate- design activity cycle.*

Cordes and Carver, made one of the first attempt to apply automated tools to requirement analysis and generate an object model from it [5]. Their paper describes the development of an environment which provides knowledge based assistance to requirements phrase from OO prescriptive. According to the paper, initial requirement provided in the SRS is converted to suitable knowledge base through human intervention. Then the knowledge base is being translated to object model to remove ambiguity. Thus the gap between informal requirement specifications and formal model is bridged by requirement analysis, its process and tools.

Juristo and Moreno, proposed an approach using linguistic information obtained from informal requirement specification SRS [6]. This informal specification is analyzed semantically as well as syntactically and a semi formal procedure is being employed to obtain OO system's component. They categorize eight different NL structures. These NL structures can be identified as `` is type of '' denoting the bottom-up simple inheritance; top-down simple inheritance denoted by ``can be''; multiple inheritance denoted by ``is a and a''; binary association denoted by ``does and'';

identification association denoted by `` is identified by"; n-ary association denoted by `` does to on"; aggregation is denoted by ``contains and" and ``are part of". The approach suggested in the paper is manual i.e., the structures are to be searched manually and finally the OO system's component is being obtained.

Russel and Dewar in their paper introduce an XML encoded reverse engineering transformation from Java to UML [7]. In this paper from the Java code, XML based Java code is being generated known as Java Markup Language (JavaML). This JavaML taken as input for ArgoUML which is an OO design tool and this tool finally produce the required UML diagram. In this paper, they observed a missing link between JavaML that represents the source code and XMI represents the design information.

Martin Necasky proposed a method of reverse engineering to obtain the conceptual diagram of the model from XML schema [8]. The author provides a semi-automatic algorithm that provides a mapping between the components of XML schema and conceptual diagram. As the paper only provides the mapping, manual participation is required through domain expert. In this paper the reverse engineering concept is being applied to generate different conceptual diagrams.

METHDOLOGY

Use of Natural Language Processing (NLP) helps to find out:

- Grammar describing the syntax of Natural language.
- Lexicon describing the lexical information about the words.
- Semantic component used to construct the literal meaning of the sentence.
- Pragmatic component used to construct non-literal meaning of a sentence.
- Parser generating the phrase tree structure of the sentence of the constitutional document.

Application of POS in NLP

Part of Speech (POS) tagging is a semantic analysis approach and deals with assigning one or more part of speech to a given word. A Tagger is a computer program that performs the job of assigning POS to words. The different types of POS are as follows:

- Noun: It is the POS which mainly used for names that may be of person, place, thing, quality or action. It can function as subject, object in sentences and can be used in its singular or plural form.
- Verb: This POS informs about the state or action of the noun or subject. Verb specifies what noun is doing and in which situation it is.
- Adjective: This POS describe or modify words. It can be used to identify or quantify another persons or things in the sentence.
- Adverb: It modifies adjective and appears before the word it modifies.
- Article: These words describe and modify other words. There are three different articles present, i.e., a, an, the.
- Conjunction: It acts as a connector between two words, sentences, phrases etc.
- Interjection: Without having no grammatical meaning, interjection is very rarely used in written. These words are mainly used while speaking.
- Preposition: It shows nouns relationship to another word in a sentence by preceding it.

- Pronoun: It is used as a replacement to the noun. Without the use of pronoun, the same noun must be repeated many times.

Identification of all these POS makes the task of NLP simpler.

Application of XML in NLP

SRS, when transformed to a desired format automatically, then the candidate for class name is being found out and subsequently the class diagram is developed. To develop the class diagram, a person having domain knowledge is required. In order to avoid the use of domain expert or manual intervention, initiative has been taken using XML and XSD.

1. XML stands for eXtensible Markup Language. It is mainly used to store and transport the data. XML encodes the document in a format, i.e., not only comprehensible by the human but also by the machine. XML provides the data which are platform, language and media independent. XML can provide multiple services related to NLP as follows:
    - Extensible: In XML different tags are being used to store data. So, whenever it needs to extend the document, few more tags can be added and extensibility can be achieved.
    - Structural: The XML document is arranged in a particular order, in which `root' is the top most element and child elements are present under it. So, a structure is maintained and complexity of the document can be easily handled.
    - Validation: Validating a structural document is necessary as sometime structure can be very complex. XML helps to validate the structural document.
2. XML Schema Document (XSD) helps to define the legal building block of XML document. Like the global variables in programming concepts, XSD has global construct which reuses the schema within same or other schema using xsd:include and xsd: import. All global constructs have a target namespace and a name. Namespace is provided in XSD to uniquely separate different documents.

In this study, Python language is being used. Python language provides different third party open source library like ``lxml'' for generating the XML [9]. Stanford Parser for parsing the NL text [10] and LancasterStemmer for stemming the words into its root word [11] are being employed for analysis.

PROPOSED APPROACH

The approach used in this paper to generated class diagram from SRS can be described as follows:

- An intermediate output is obtained from SRS document using grammatical construct of the sentence and Object Oriented principles of design.
- The obtained intermediate output is again processed using the concept of XML and XSD to generate the desired class diagram, **which is a novel approach**

The steps carried out to obtain the required result from the SRS can be explained in the Figure 1
In this paper, the SRS of Bank ATM system has been considered as a case study to identify the elements of classes, constraints of different methods, responsibilities indicating collaboration between classes.

- The SRS of Bank ATM is taken as input and the SRS goes through Stanford Parser which assigns POS to each word [10].

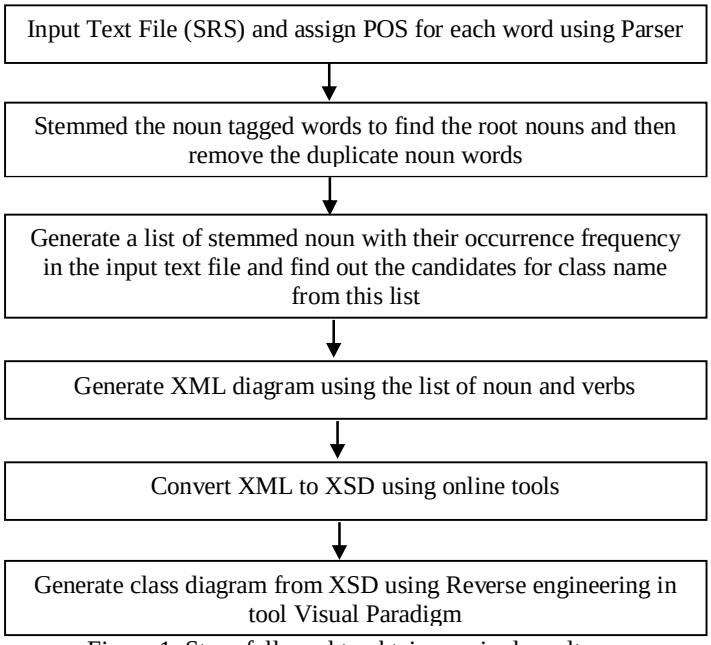

Figure 1: Steps followed to obtain required result

- Each word is stored according to their assigned POS. The list of nouns contains both the noun in its singular and plural form. The nouns, present in the plural form is stemmed to singular form and again stored in the noun list. It can be found out that the stemmed noun list may contain repeated words which increases the overload only. For the stemming of the noun, LancasterStemmer tool is being used and a unique list of nouns is found out [11].
- Using this list, another list is created with these words with the number of occurrence of each word is identified. From the list, only the nouns with higher frequency are considered as a candidate for class names.
- Finally the list of nouns, the candidate for the class name and list of verbs, the candidate for member function of class is identified as input and an XML document is being generated using an open source library `lxml' [9].
- The generated XML is then converted to XSD using on-line tool ``free formatter" [12] . XSD helps in validating the structure of XML.
- The XSD generated is given as input to a software design tool named as ``Visual Paradigm" in order to obtain the class diagram of UML [13].

IMPLEMENTATION

In this paper, the problem statement of the Bank ATM is considered as input [2]. The problem statement is being modified as per the requirement of the implementation. The input statements considered are

``Design the software to support a computerized banking network including both human cashiers and automatic teller machines (ATM) to be shared by consortium of banks. Each bank provides its

own computer to maintain its own accounts and process transactions against them. Cashier stations are owned by individual banks and communicate directly with their own bank computers. Human cashiers enter account and transaction data. Automatic teller machines communicate with a central computer which clears transactions with the appropriate banks. An automatic teller machine accepts a cash card, interact with the user, communicate with the central system to carry out the transaction, dispense cash, and prints receipts. The system requires appropriate record keeping and security provision. The system must handle the concurrent access to the same account correctly. The bank will provide their own software for their computers. The withdrawal amount should be in multiple of 100. The withdraw amount has a limit of maximum 50000 per day. If account balance is less than 1000 withdraw is not allowed. You are to design the software for the ATMS and the network. "

This text is taken as input to the Stanford parser. The parser parses each word and provide an appropriate POS tag to each word.

After assigning the POS to each word, the words having POS noun and verb are being separated out and stored. An example of the verb words found out from the input text shown in Fig 2(a) and also of the noun words found out after parsing the input text is shown in Fig 2(b) respectively.

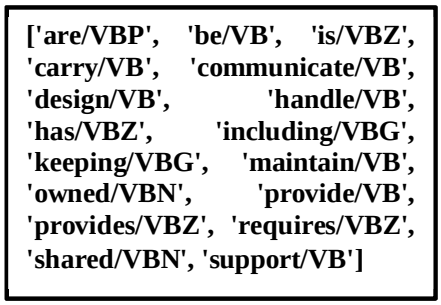

Figure 2(a) Example of verb tagged words of Bank ATM file

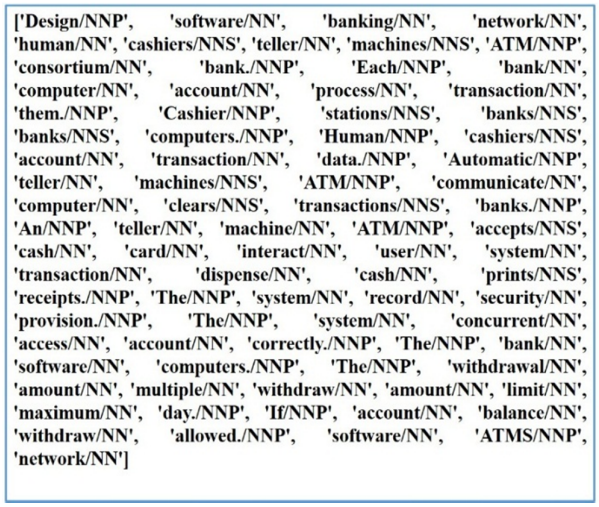

Figure 2(b) example of noun tagged words of Bank ATM file

It is being found out that many duplicate nouns are present in the list both in its singular and plural form. So, the words need to be stemmed, duplicate words are removed and finally a list is generated, showing the frequency of the unique nouns in the input text. For stemming of words, Lancaster Stemmer tool is being used, which converts the words into its root form [11]. The list is showing the stemmed words and its corresponding frequency is shown in Figure 3.

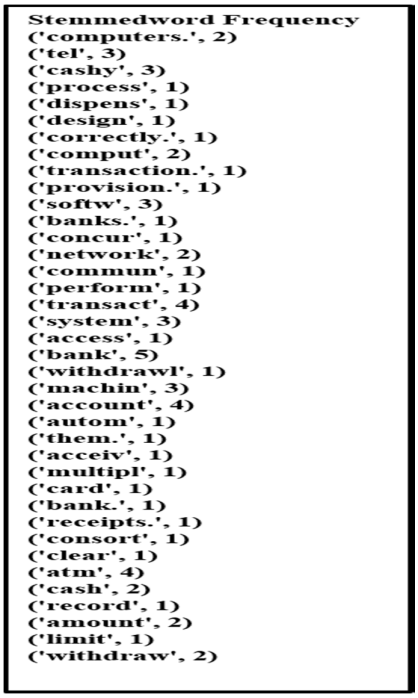

Figure 3 noun and frequency list

From the generated list, the noun words with the highest frequency is being selected as possible candidate for class names. While finding out the candidate for class name, human intervention may be needed as there are many words which are identified as a noun and have higher frequency also, but they are not helpful for fixing up major functionalities of SRS. So, from the Figure 3 words having higher frequency are eligible for candidate for class such as ``Bank, Transaction, Account, ATM". The no of occurrence of Bank is five, Transaction and Account is four and ATM is two. Apart from that list another word is being selected from ATM context i.e., Customer.

After selecting the nouns and the verbs present, the structure of the class diagram is planned using XML, XSD, and Visual Paradigm. The XML generated is shown in Figure 4 (a) and the XSD generated after transforming XML online is shown in Figure (b).

The class diagram as shown in the FIGURE 5. In this paper, five different classes are generated, i.e., ``Account, ATM, Bank, Transaction, Customer" and a Central_system which controls the over-all system.

PERFORMANCE EVALUATION

Different performance metrics such as Precision, recall, F-measure are used to evaluate the performance of the proposed approach. The correctness or the relevance of the classes identified in the conceptual model is indicated by precision. The ability of the automation to generate all classes

```
<Central_System>
<Bank>
  <function>maintain</function>
  <function>process</function>
</Bank>

<ATM>
  <function>accepts</function>
  <function>interact</function>
</ATM>

<Transaction>
  <function>withdraw</function>
  <function>print_receipt</function>
</Transaction>

<Account>
  <function>update</function>
  <function>null</function>
</Account>

<Customer>
  <function>register</function>
  <function>deposit</function>
</Customer>

</Central_System>
```

Figure 4(a) XML generated from the program

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="Central_System">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Bank">
     <xs:complexType>
      <xs:sequence>
       <xs:element type="xs:string" name="function" maxOccurs="unbounded" minOccurs=
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="ATM">
     <xs:complexType>
      <xs:sequence>
       <xs:element type="xs:string" name="function" maxOccurs="unbounded" minOccurs=
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="Transaction">
     <xs:complexType>
      <xs:sequence>
       <xs:element type="xs:string" name="function" maxOccurs="unbounded" minOccurs=
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="Account">
     <xs:complexType>
      <xs:sequence>
       <xs:element type="xs:string" name="function" maxOccurs="unbounded" minOccurs=
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="Customer">
     <xs:complexType>
      <xs:sequence>
       <xs:element type="xs:string" name="function" maxOccurs="unbounded" minOccurs=
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Figure 4(b) XSD generated after online transformation of XML

is indicated by the recall percentage. Over-specification indicates number of unnecessary but correctly identified classes. The formula for these measures are given below:

$$\text{Recall} = \frac{\text{Ncorrect}}{\text{Ncorrect} + \text{Nmissing}} \qquad \text{Equation 1}$$

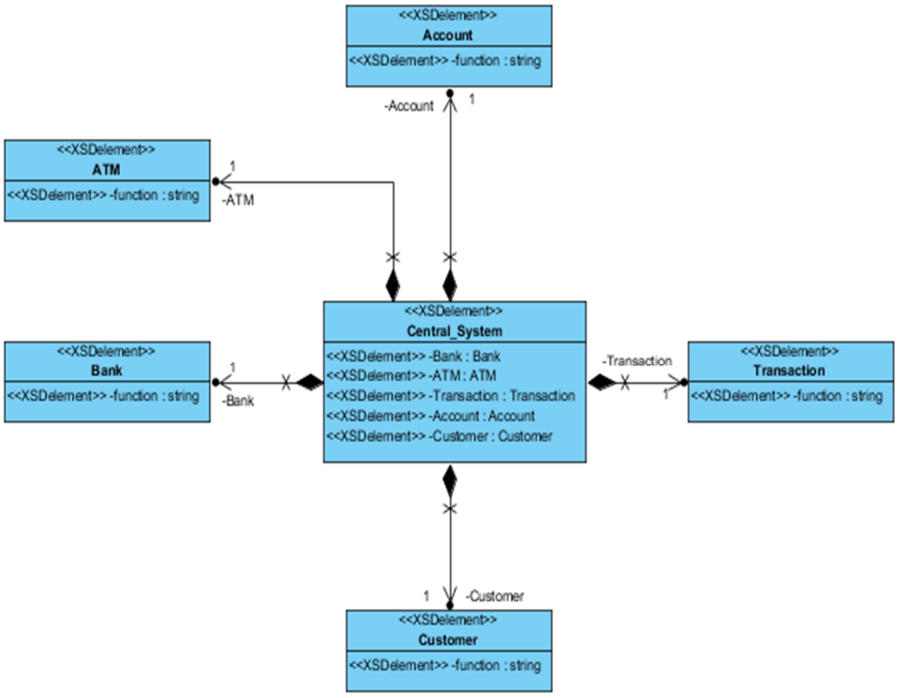

Figure 5 Class Diagram Generated from XSD using reverse engineering

$$\text{Precision} = \frac{\text{Ncorrect}}{\text{Ncorrect} + \text{Nincorrect}} \qquad \text{Equation 2}$$

$$\text{Recall} = \frac{\text{Nextra(valid)}}{\text{Ncorrect} + \text{Nmissing}} \qquad \text{Equation 3}$$

$N_{correct}$ is the no of correct classes identified; $N_{incorrect}$ is the no of correct classes identified as wrong; $N_{missing}$ is the number of classes extracted by the human expert and missing by the system; $N_{extra\ (valid)}$ is the number of extra valid classes. The recall and precision percentage are supposed to be as high as possible, i.e., nearly 100 % and on the other hand, over-specification is supposed to be as low as possible, i.e., nearly 0 %.

The following Table 1 shows a comparison of the result obtained using proposed approach and a present paper [14]. From the table, it can be concluded that the proposed method shows better result.

Table 1: Evaluation Result

Case Study	Present Approach [14]			Proposed Approach		
	Recall	Precision	Over-spec	Recall	Precision	Over-spec
ATM [2]	100	91.67	9.09	**100**	**93**	**9**

CONCLUSION

During the course of the paper, an attempt has been made to reduce the involvement of human and automate the system. In this paper role of domain expert is minimized to select words from the list of words and its frequency that became a candidate for class names.
The proposed research work intend to focus further on:

- Applying machine learning techniques in order to optimize the parsing process.
- Generalizing the proposed approach for multiple case studies

REFERENCES

E. Kumar, Natural Language Processing, IK international Pvt. Ltd., 2011.

J. Rambaugh, J. I. and B. G., The Unified Modeling Language Reference Manual, Pearson Higher Education, 2004.

R. Abbott, "Program Design by Informal English Description," Communication of the ACM, vol. 26, no. 11, pp. 882-894, November 1983.

M. Saeki, H. Horai and E. H., "Software Development Process from Natural Language Description," in 11th International Conference on Software Engineering, ICSE'89, New York, USA, 1989.

D. Carver and D. Cordes, "An Object-oriented framework to support architectural design development," in Twenty-third annual Hawaii International Conference, Kaiula-Kona, Jan 1990.

N. Juristo, A. Moreno and M. Lopez, "How to use linguistic instruments for object-oriented analysis," IEEE software, vol. 17, no. 3, pp. 80-89, 2000.

C. Russell and R. Dewar, "XML Encoded Reverse Engineering of Java to UML," Citeseer, 2003.

M. Necasky, "Reverse engineering of XML schemas to conceptual diagrams," in Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modeling-Volume 96, 2009.

S. Behnel, "lxml - XML and HTML with Python," [Online]. Available: http://lxml.de/.

"The Standford Parser: A Statistical Parser," 2013. [Online]. Available: http://nlp.stanford.edu/software/lex-parser.shtml.

C. Paice and R. Hooper, "Lancaster Stemmer," 2005. [Online]. Available: http://www.comp.lancs.ac.uk/computing/research/stemming/index.htm.

"XML to XSD convertor: Tool for Formatters, Converter, Validators," [Online]. Available: http://www.freeformatter.com/xsd-generator.html.

"Visual Paradigm: Software Design Tool for Agile Software Development," [Online]. Available: http://www.visual-paradigm.com.

V. Sagar, R. Vidhu Bhala and S. Abirami, "Conceptual modeling of natural language functional requirements," Journal of Systems and Software, vol. 88, pp. 25--41, 2014.