AutoClass: Automatic Text to OOP Concept Identification Model

Fatma Bozyiğit
Celal Bayar University
Department of Software Engineering
Manisa/Turkey

Özlem Aktaş
Dokuz Eylül University
Department of Computer Engineering
İzmir/Turkey

Deniz Kiliç
Celal Bayar University
Department of Software Engineering
Manisa/Turkey

ABSTRACT

Requirement specification is one of the most crucial steps in Software Development Life Cycle (SDLC). Well-written requirement documents include expressive keywords such as names of classes, attributes and methods to be used in software design. Knowledge extraction from these documents may be helpful in order to design conceptual model for Object Oriented (OO) systems. Recently, automatic text-to-UML (Unified Modelling Language) transformation applications are used widely in software engineering. These automatic modelling systems make the work of software teams easier and reduce total cost of software development process. This paper presents a CASE tool called AutoClass which extracts class diagrams and generates C# source code from the requirement documents. Natural Language Processing (NLP) techniques and rule-based model are used to implement automatic concept identification model in the study. The experimental results indicate that the proposed system is more accurate in terms of extracting class diagrams when it is compared to other studies. Moreover, it has an accuracy rate that is over 90%.

General Terms

Natural Language Processing (NLP), rule-based model, Object Oriented (OO) Systems, Unified Modelling Language (UML).

Keywords

Software Requirement Analysis, OO based concept identification, Automatic text to UML transformation process.

1. INTRODUCTION

Gathering and analyzing users’ requirement is the most critical phase in Software Development Life Cycle (SDLC) [1]. Requirement analysis is a systematic procedure that enable to conceive the needs of customers during the SDLC. As a result, the success of requirement analysis and requirement management directly affects the other steps in SDLC. It can be easily concluded that well-documented requirement document has a significant impact on increasing performance of final software product. Requirements which are written in Natural Language (NL), are turned into a well-known design by performing some processing steps. For example, business logic and needs of all users must be understood by a system analyst and transformed into Unified Modelling Language (UML) [2].

standard by using a CASE tool within Object Oriented Design (OOD) [3] paradigm.

OOD is an approach that has specific characteristics and behaviors including class structures and relationships between the classes. Attributes of classes hold information about the object. Behaviors in class, also called methods, describe what can be done with an object. While attributes and methods give information about semantic of classes, relationships between classes (composition, aggregation, inheritance, association, instantiation, etc.) give clue about semantic of all systems. After the completion of OOD, obtained classes, their attributes, relations, and operations must be kept and moved from analysis to test stage. With this perspective, it can be speculated that UML transformation from NL-based requirements is an important step in OOD [4].

By recent advances, usage of automatic text to UML transformations in software development becomes more prevalent and expediates preparation phase of software. The aim of automatic transformation is creating a model to reduce workload of software development and gain time. However, the development of automatic transformation system which use textual data as input can be a challenging task in terms of the following points [5].

- It is not possible to model concepts that is not clearly defined. There may be missed classes, attributes, methods etc. in created UML diagrams.
- Words which have same meaning (synonym) can be represented in different ways. Therefore, complexity may increase while finding classes and attributes.
- The distinction of attribute and class names does not lead to actual results every time.
- Deciding the types of inter-class relationships can be challenging because of different semantics.

In this paper, automatic concept identification is proposed for the linguistic aspects of the English language. A CASE tool called as AutoClass is developed to perform the study of concept identification. The approach in the study consists of two parts: Natural Language Processing (NLP) techniques and rule-based model. The tool processes NL-based software requirements to build class diagram and generate C# program code in OOP frame. The evaluation results show that the accuracy rate of AutoClass is more accurate than reviewed studies in terms of finding classes in OO model.

The structure of this paper is organized as follows. In next section, a survey of the related works which implement automatic concept identification is presented. Following section presents general methodology of the available system and explains the details of steps in processing pipelines. Section 4 gives information about the design of system, dataset and evaluation results. Section 5 concludes the paper and gives information about our future work.
2. RELATED WORKS

There are several approaches for automation of conceptual modeling. In this part, a review of 10 studies which utilize automatic concept identification, is performed. The studies are evaluated in terms of applied methodologies and outputs in their study.

Bajwa et al. propose a study to obtain both UML diagrams and program code from the requirement analysis of a software [6]. System in the study is called as UMLG (UML Generator). They use a rule-based model, NLP techniques and semantic analysis to analyze NL text and extract various OO modelling elements. They use a dataset that includes three text documents to test the study. The documents are split into three categories; simple, average and complex. They claimed that accuracy rate of study is between 80% and 85%.

In another study, Bhala et al. aim create rule based model to identify classes and relationships by using functional specifications [7]. They generate 38 rules that are gathered under three categories; class, attribute and relation. They focus on creating a model to be used by analysts and help stakeholders to understand the requirement before they conclude requirement phase of software project. Design components (classes, attributes, entities and relationships) are extracted from textual data as output of study. The test of study is done by using only three documents.

Tripathy et al. use NLP techniques to obtain information from incomplete and ambiguous requirement documents [8]. It is proposed to create class diagrams and generate source code by using this information. First, Stanford Parser is used for dividing text into a sequence of tokens and intermediate data is obtained by using grammatical construct of the sentence (POS). Then, Lancaster Stemmer for stemming the words into roots is utilized. Obtained words are stored in database according to their assigned POS tags (noun, verb, adjective and conjunctions). Frequencies of all words in dataset are counted and most frequent nouns are specified as candidates class names. Most frequent verbs are selected as method names. In the study, XML is used for storing and exchanging data. Then, HTML becomes a part of activity as showing storing data. They evaluate only problem statement of the Bank ATM with their study and additional data set which contains requirement analyses are not used. Therefore, test results of the study are inadequate. It is pointed that accuracy rate is accounted as 96%.

Shinde et al. propose a method to construct class and sequence diagrams from user modelling information with strong semantic support [9]. Their study also supports code template generation in Java. The methodology works by analyzing the requirement specification. Firstly, tokenization step splits a plain text file into tokens. POS tagger is used to get nouns and verbs are passed to the stemmer which returns the root and suffix of each word. The study makes use of WordNet as ontology to get semantic relation between the classes and attributes. This analysis helps in finding and identifying classes, attributes, methods, actors and messages between them.

Li et al. claim that current approaches are inefficient to extract all semantic and design UML diagrams from requirements forms in detail [10]. It is implied that existing NLP systems for automatic OOA have some gaps, so they can’t give successful results many times. In the study, the purpose is showing how the problems related to NL can be addressed in OOA by using NLP tools.

Arelлоnα et al. indicate that ontology is so effective during validation of the completeness of system design that is deduced from textual requirement analyses [11]. They proposed a model which uses a new framework for an aircraft system and present a web based application which is called ‘TextReqVerification’. The new approach in the study combines standard NLP techniques (tokenization, POS, stemming, etc.) and predefined ontology model.

Mu et al. propose a system to extract functional requirements by analyzing text based software requirement specification [12]. Starting point of the study is analyzing linguistic characteristics of documents belong to each processes in work pipeline. After that, rules that contains 10 cases to make clear semantics of system, specified and applied associated with NLP techniques. Difference of the study from the others in literature is trying to obtain customized format for specific functions in documents. That is, the conceptual model is not obtained as an output.

Yue et al. present a survey of the approaches for transformation between requirements in the form of text and analysis model [13]. They reviewed and evaluated 16 primary studies in the literature. They specify research question to compare available systems. According to analysis results, make suggests to improve the reviewed approaches.

Ibrahım and Ahmad propose a model called ‘RACE’ for translating software requirements to OO model by using NLP methods [14]. The system obtains class diagrams of UML by evaluating requirement analyses written in NL. It is mentioned that the rule-based system is built at the start of study, then NLP techniques are applied to textual data with specified rules. Finally, domain ontology is used to refine candidate classes in the diagram to obtain more accuracy rates.

Zhou and Zhou use domain knowledge with NLP spider model techniques to extract information from written requirements to build class diagrams for OO designed system [15]. It is claimed that using domain ontology empowers performance of concept identification. It is seen that study is successful especially in categorizing relationships such as association, aggregation, and generalization. The approach in the study is attributed to semantic relationship among class structure.

3. BACKGROUND

Framework of the study which integrates NLP techniques and rule-based model to transform textual requirements into conceptual model is shown in Figure 1. In consequence of general architecture, after completing standard NLP steps, rule-based model becomes part of activity. That is outputs of NLP methods are used as input for rule-based model.
3.1 NLP Methods

Systems for generating OOD from textual requirements generally take advantages of NLP methods to ease requirement analysis.

Firstly, tokenization step that divides texts which contains software requirements, into words, is realized. Then, POS tagging is performed to identify which role each token has in the sentences. After morphological analysis of given text is performed to define structure of words, stemming process which reduces derived words to their base forms, is completed.

3.1.1 Tokenization

Tokenization is the identification of each “atomic” unit / word in a sentence [16]. This process enables consisting of tokens arranged in a syntactically valid combinat ion. So, in order to analyze a text document, tokenization must firstly be performed and groups of word must be obtained. Thus, it gets easy to make sense out of the textual documents.

3.1.2 POS Tagging

The process of assignment one of assignment to the word in a sentence is called POS tagging [17]. It is a semantic analysis approach that expresses grammatical rules and how a word is used in a sentence. So, intermediate data is obtained by using grammatical construct of the sentence. The relationships between tokens can also be identified by using POS tagging. POS tags include nouns, verbs, adverbs, adjectives, pronouns, conjunction and their sub-categories. An example of POS tagging is shown in Figure 2.

In this study, we used Stanford Parser to obtain the POS tags of all words in documents [18]. Words are classified into nouns, verbs, adjectives or adverbs and stored according to their assigned POS. Further, nouns are classified into proper and common nouns.

![Figure 2: Representation of POS tagging](image)

3.1.3 Stemming

A stemming algorithm is a computational procedure which reduces derived words to their base forms [19]. Derivational affixes and inflectional suffixes are removed and words are transformed into root morph. For example, as seen in Figure 2, words like “computing” and “computed” are converted to “compute” after the stemming process.

After POS tagging, Lancaster stemmer is used to stem tagged words [20]. Then, duplicate words are removed from database and unique list of tags is obtained.

![Figure 3: Representation of stemming](image)
3.2 Rule Based Model

The extraction of design elements that belong to OO models is the base of concept identification model. In this study, class diagrams and source codes are generated by using elements (classes, attributes, methods and relations) in OO model. The aim of using a rule-based model is to specify the valid components of the conceptual model. A set of rules is formed to create the rule-based model. While some rules are taken from reference’s paper, some are created specifically by authors of this study. The rules are explained as following:

- Nouns in sentences are candidate for class and attribute names.
- Proper nouns are removed from candidate pool of class, property and method names.
- The nouns which have the maximum frequency are favorite candidates for class names.
- Verbs in sentences are included by pool of methods’ names.
- Linking verbs between two nouns may indicate a relationship.
- Verbs included by phrasal verb inform about relationships in model most of times.

Example: “The family bring up a child.” → Family is class 1, bring up indicates relation, child is class 2.

- If a verb is included by the following list {include, involve, contain, consist of, etc.}, there may be aggregation and composition relationship type.
- Succession of the nouns in the sentences (if there are no elements between them except space) are aggregated and they are formed into single name.

4. DESIGN OF SYSTEM

4.1 Experimental Dataset

The evaluation of the concept identification model is a challenging task, because extracting design elements from requirement analysis may be different according to the analysts. In order to compare and obtain more accurate results, a standard requirement text is chosen as used in previous studies. The input requirement is an ATM problem statement which is used in Bhala’s study to test their designed system [7]. Bhala et al. indicate that they generate the problem text by changing Rumbaugh’s ATM model [21]. Figure 4 shows the ATM problem which is used in Bhala’s study. In this study, we 3 examples of requirement which are categorized as simple, average and complex, are created by modifying Bhala’s ATM problem to test AutoCode tool.

The system must support a computerized banking network that includes both human cashiers and ATMs. The computerized banking network will be shared by a consortium of banks. Each bank provides a computer that maintains the bank's accounts and processes transactions against the accounts. Cashier stations are owned by individual banks and communicate directly with the bank’s computers. Human cashiers enter the account data and transaction data. An ATM communicates with a central computer. The central computer clears transactions with the banks. An ATM accepts a cash card and interacts with the user. An ATM communicates with the central computer to carry out transactions. An ATM dispenses cash, and prints receipts. The system requires appropriate record-keeping and security provisions. The system must handle concurrent access to the same account correctly. The banks will provide the bank’s own software for the bank’s own computers.

Fig 4: ATM Problem Statement (used by Bhala et al.)

4.2 Evaluation Results

*Precision* is defined as the ratio of *TP* to the total number of the system’s outputs (*TP + False Positive (FP))*. Equation of *Precision* is defined in (2):

\[ Pr = \frac{TP}{TP + FP} \]  

(2)

The *F-measure* of the system is defined as the harmonic mean of its precision and recall.

\[ F_{measure} = \frac{2(Pr \times Re)}{Pr + Re} \]  

(3)
F-measure is used to evaluate performance of generated conceptual model in the study and accuracy rate is calculated for ability of finding classes, attributes, methods and relations separately. Table 1 shows accuracy rates of all documents in class, attribute, method and relationship categories.

<table>
<thead>
<tr>
<th>Document</th>
<th>Class</th>
<th>Attribute</th>
<th>Method</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>100%</td>
<td>100%</td>
<td>80%</td>
<td>95%</td>
</tr>
<tr>
<td>Average</td>
<td>100%</td>
<td>100%</td>
<td>74%</td>
<td>89%</td>
</tr>
<tr>
<td>Complex</td>
<td>100%</td>
<td>100%</td>
<td>70%</td>
<td>78%</td>
</tr>
</tbody>
</table>

Extracted OOD elements by AutoClass are compared to evaluation results of Bhala’s study. Number of correct classes and their respective attributes, methods, relations which are obtained by AutoClass and Bhala’s study are shown as Figure 5. According to view of the results shown in Figure 5, it can be easily seen that AutoCode gives more accurate results than Bhala et al. in terms of finding classes and attributes.

**5. CONCLUSION**

Transforming requirements into conceptual identification model is a vital step in software development. Although mostly done manually, there are available approaches to automate this step of SDLC. In this paper, a case tool which integrates NLP methods and transformation rules to do text to UML transformation for English NL based requirement documents.

Based on the review results, it is easily realized that there are some gaps that must be fulfilled in literature. For example: relationship identification among classes is not completed properly in existing works and they are weak in finding association relationship. Also, it is seen that the accuracy of the current studies can be increased up by using specific rules. The experimental results demonstrate that AutoClass is more accurate than studies in literature in terms of extracting correct classes and relations between classes. The proposed model achieve over 90% accuracy.

As future works, it is aimed to design a new system which extends the previous studies for following functionalities;

- Generating other diagram types such as object, sequence, activity
- Generating source code more than one programming language
- Utilizing a large scale dataset that includes various requirement examples to test study.

In accordance with this purpose, semantic analysis will become the most important part for our designed approach.

**6. REFERENCES**


Methodologies. Advanced Topics in Databases Series. Pages 216-240.


