Natural Language Processing based Automatic Multilingual Code Generation

Imran Sarwar Bajwa, M. Shahid Naveed, M. Abbas Choudhary

Balochistan University of Information Technology and Management Sciences
Quetta, Pakistan.

Imransbajwa@yahoo.com, shahid_naweed@hotmail.com, abbas@buitms.edu.pk

Abstract

Unified modeling language is being used as a premier tool for modeling the user requirements. These CASE tools provide an easy way to get efficient solutions. This paper presents a natural language processing based automated system for generating code in multi-languages after modeling the user requirements based on UML. UML diagrams are first generated by analyzing the given business scenario provided by the user. A new model is presented for analyzing the natural languages and extracting the relative and required information from the given requirement notes by the user. User writes the requirements in simple English in a few paragraphs and the designed system has conspicuous ability to analyze the given script. After compound analysis and extraction of associated information, the designed system draws various UML diagrams as activity diagrams, sequence diagrams, class diagrams and Uses cases diagrams. The designed system has robust ability to create code automatically without external environment. The designed system provides a quick and reliable way to generate UML diagrams and generate respective code to save the time and budget of both the user and system analyst.

Keywords: Automatic Code generation, Multi-lingual Code, Information extraction, Natural language processing.

1. Introduction

In the current age, the tools and techniques of software engineering has been changed to an adequate extent. Now every step of software engineering follows the rules of object oriented design patterns. Same the case is with Software process which uses Unified Modeling language for modeling the user requirements. In recent times, there is no software which provides services to draw UML diagrams more efficiently except Rational Rose, Smart Draw etc and there is no doubt that these are reasonably good software but has many drawbacks.

Conventionally, the system analyst has to do a lot of work for deducing the business logic and understanding the user requirements before drawing the UML diagrams by using orthodox CASE tools. Hence, there is wastage of so much time due to the dull nature of the available CASE tools for the required scenario. In today's world everybody needs a quick and reliable service. So it was needed that there should be some sort of intelligent software for generating UML based documentation to save time and budget of both the user and system analyst.

In order to resolve all such issues, we need software, which facilitates both users and software engineers. As far as this software is concerns the time, it takes to explore all the facilities and services, should be quite less than a minute and this information is quite useful for the users.

2. Problem Description

The problem specifically addressed in this research is primarily related to the software analysis and design phase of the software development process. Few years ago data flow diagram's were being used to symbolize the flow of data and represent the user's requirements. But in current age, unified modeling language is used to model and map the user requirements, which is more comprehensive e and authentic way to of representation and it is beneficial for the later stages of software development.

After modelling and mapping the user requirements, next phase is to create the programming code in certain computer language. Hand written code by a programmer is a conventional and orthodox solution of the problem which is time consuming. Modern software engineering requires quick and automated solutions which may have ability to create more than the half code, so that the programmer may create the application after making appropriate adjustments and alterations in the automated generated code with less effort in less time as compared to the traditional approaches.

3. Problem's Solution

The conducted research provides a robust solution to the addressed problem. Multi-lingual Code Generator (MCG) provides the solution of the problem. The functionality of

the conducted research was domain specific but it can be enhanced easily in the future according to the requirements. Current designed system incorporate the capability of mapping user requirements after reading the given requirements in plain text and drawing the set of UML diagrams as Class Diagram, Activity Diagram, Sequence Diagram, Use case diagram and Component Diagram.

After drawing UML diagrams, designed system has profound ability to create code in various languages as Java, Visual Basic, C # and C++. An Integrated Development Environment has also been provided for User Interaction and efficient Input and output.

4. Natural Language Processing

The understanding and multi-aspect processing of the natural languages that are also termed as "speech languages", is actually one of the arguments of greater interest in the field artificial intelligence field [6]. The natural languages are irregular and asymmetrical. Traditionally, natural languages are based on un-formal grammars.

There are the geographical, psychological and sociological factors which influence the behaviours of natural languages [17]. There are undefined set of words and they also change and vary area to area and time to time.Due to these variations and inconsistencies, the natural languages have different flavours as English language has more than half dozen renowned flavours all over the world. These flavours have different accents, set of vocabularies and phonological aspects. These ominous and menacing discrepancies and inconsistencies in natural languages make it a difficult task to process them as compared to the formal languages[13].

In the process of analyzing and understanding the natural languages, various problems are usually faced by the researchers. The problems connected to the greater complexity of the natural language are verb's conjugation, inflexion, lexical amplitude, problem of ambiguity, etc. From this set of problems the problem which ever causes more difficulties is problem of ambiguity. Ambiguity could be easily solved at the syntax and semantic level by using a sound and robust rule-based system.

4.1. The problem of ambiguity

During the interpretation process, we have talked about previously we use to distinguish such ambiguities in four classes: lexical ambiguities, syntactic ambiguities, semantic ambiguities and pragmatic ambiguities. Such distinction is not exhaustive but it is useful to focus on the problem and for practical purposes [7].

4.1.1. Lexical Ambiguity:

The problem of words written or pronounced not correctly is omitted in this problem scenario. These kinds of errors are simply solvable through the comparison with the expressions contained in a dictionary. Lexical ambiguity is created when a same word assumes various meanings [3]. In this case that ambiguity is generated from the fact that which meanings will be incorporated in which scenario. As an example we consider the "cold" adjective in the following sentences:

"That room is cold."

"That person is cold."

It turns out obvious that the same "cold" adjective assumes, in the two phrases different meanings. In the first sentence it indicate a temperature, in the second one a particular character of a person.

4.1.2. Syntactic ambiguities:

Syntax analysis is performed on world level to recognize the word category. The syntactic analysis of the programs would have to be in a position to isolate subject, verbs, objects and various complements. It is little complex procedure.

"Mario eats the apple."

In this example, the actual meanings are that "Mario eats apple" but ambiguity can be asserted if this sentence is conceived as "the apple eats Mario".

4.1.3. Semantic ambiguities:

To analyze a phrase from the semantic point of view means to give it a meaning. This should let you understand we arrived to a crucial point. Semantic ambiguities are most common due to the fact that generally a computer is not in a position to distinguish the logical situations.

"The car hit the pole while it was moving."

All of us would surely interpret the phrase like "The car, while moving, hit the pole.", while nobody would be dreamed to attribute to the sentence the meant "the car hit the pole while the pole was moving ".

4.1.4. Pragmatic ambiguities:

Pragmatic ambiguities born when the communication happens between two persons who do not share the same context. As following example:

"I will arrive to the airport at 12 o'clock."

In this example, if the subject person belongs to a different continent, the meanings can be totally changed.

5. Object-Oriented Analysis and Design

Analysis and design of an information system relates to understand and intend the framework to accomplish the actual job. Typically, design is relates to manage and control the complexity parameter in a domain. A robust design method also helps to split big tasks into controllable breakups (Condamines, 2001). In software engineering, design methods provide various notation usually graphical ones. These notations allow to store and communicate the perpetual design decisions. Object-oriented design has overruled the typical analysis and design techniques as structured design and data-driven design (Androutsopoulos, 1995). As compared to old style design paradigms, object-oriented design models the every active entity of the problem domain using concept of objects. Objects have:

State (shape and condition)
Behaviour (What they perform)

Object-oriented languages use variable to manifest the state of an object and methods or procedures to implement the behaviour of an object. For example, a ball could be an object. There are different parameters of shape as colour, size, diameter, shape, type, etc. This object can also have behaviour as throw, roll, catch, hit, etc. The major task in analysis and design phase is to identify the valid objects and specify there states and behaviours. In conventional methods, system analyst performs this tough job and then maps this information into UML using some graphical tool as Visio or Rational Rose.

In the context of this research, objects are automatically identified from a problem domain. User provides the input text in English language related to the business domain. After the lexical analysis of the text, syntax analysis is performed on word level to recognize the word category [4]. First of all the available lexicons are categorized into nouns, pronouns, prepositions, adverbs, articles, conjunctions, etc. The syntactic analysis of the programs would have to be in a position to isolate subject, verbs, objects, adverbs, adjectives and various other complements. It is little complex and multipart procedure.

"Zia is playing with the red ball."

For this example, following is the output.

Lexicons	Phase-I	Phase –II
Zia	Noun	Object
is	Helping-Verb	-------
playing	Verb	Method
with	Preposition	-------
the	Article	-------
red	Noun	Attribute
ball	Noun	Object

This is the final output of lexical assessment phase and all nouns are marked as objects and verbs are marked as methods and all adjective are marked as states of that particular object. In the above example, there are two objects 'Zia' and ball. '*Playing*' is method of '*Zia*' and '*red*' is the concerned attribute of the object '*ball*'.

6. Used Methodology

Conventional natural language processing based systems user rule based systems. Agents are another way to address this problem[8]. In the research, a rule-based algorithm has been used which has robust ability to read, understand and extract the desired information. First of all basic elements of the language grammar are extracted as verbs, nouns, adjectives, etc then on the basis of this extracted information further processing is performed.

In linguistic terms, verbs often specify actions, and noun phrases the objects that participate in the action [16]. Each noun phrase's then role specifies how the object participates in the action. As in the following example:

"Robbie hit a ball with a racket."

A procedure that understands such a sentence must discover that Role is the agent because he performs the action of hitting, that the ball as the thematic object because it is the object hit, and that the racket is an instrument because it is the tool with which hitting is done.

Thus, sentence analysis requires, in part, the answers to these actions: The number of thematic roles embraced by various theories varies probably. Some people use about half-dozen thematic roles [9]. Others use more times as many. The exact number does not matter much, as long as they will great enough to expose natural constraints on how verbs and thematic-instances form sentences.

- **Agent:** The agent causes the action to occur as in "Zahid hit the ball," Zahid is agent who performs the task. But in this example a passive sentence, the agent also may appear in a prepositional "The ball was hit by Zahid."
- **Co-agent:** The word with may introduce a join phrase that serves a partner in the principal agent. They two carry out the action together "Zahid played tennis with Ali."
- **Beneficiary:** The beneficiary is the person for whom an action has bee performed: "Ali bought the balls for Zahid." In this sentence Suzie is beneficiary.
- **Thematic object:** The thematic object is the object the sentence is really all about— typically the object, undergoing a change. Often the thematic object is the same as the syntactic direct object, as "Zahid hit the ball." Here the ball is thematic object.

- **Conveyance:** The conveyance is something in which or on which travels: “Aslam always goes by train."
- **Trajectory:** Motion from source to destination takes place over at trajectory. ID contrast to the other role possibilities, several prepositions can serve to introduce trajectory noun phrases: "Ali and Zahid went to Islamabad through Lahore."
- **Location:** The location is where an action occurs. As in the trajectory role, "several prepositions arc possible, which conveys means in addition to serving as a signal that a location noun phrase is "Aslam and Ahmed studied in the library, at a desk, by the wall, a picture, near the door."
- **Time:** Time specifies when an action occurs. Prepositions such at, before, and after introduce noun phrases serving as time role fill "Ahmed and Ali left before noon."
- **Duration:** Duration specifies how long an action takes. Preposition such as for indicate duration. "Aslam and Ahmed jogged for an hour.”

7. Architecture of Designed System

The designed UMLG system has ability to draw UML diagrams after reading the text scenario provided by the user. This system draws diagrams in five modules: Text input acquisition, text understanding, knowledge extraction, generation of UML diagrams and finally multi-lingual code generation as shown in following fig.

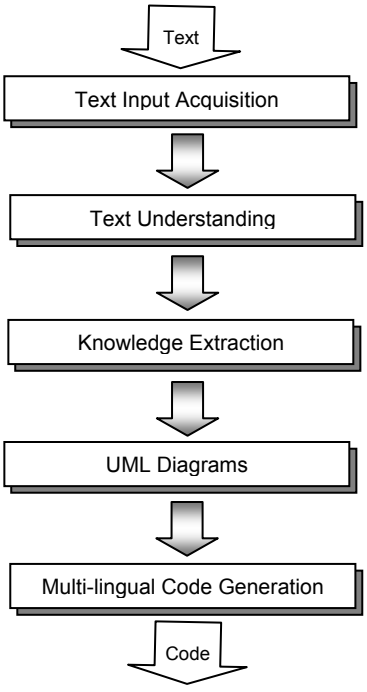

Fig 1: Architecture of the designed MCG system

7.1. Text input acquisition

This module helps to acquire input text scenario. User provides the business scenario in from of paragraphs of the text. This module reads the input text in the form characters and generate the words by concatenating the input characters. This module is the implementation of the lexical phase. Lexicons and tokens are generated in this module.

7.2. Text Understanding

This module reads the input from module 1 in the from of words. These words are categorized into various classes as verbs, helping verbs, nouns, pronouns, adjectives, prepositions, conjunctions, etc.

7.3. Knowledge extraction

This module, extracts different objects and classes and their respective attributes on the basses of the input provided by the preceding module. Nouns are symbolized as classes and objects and their associated attributes are termed as attributes.

7.4. UML diagram generation

This module finally uses UML symbols to constitute the various UML diagrams by combining available symbols according to the information extracted of the previous module. As separate scenario will be provided for various diagrams as classes, sequence, activity and use cases diagrams, so the separate functions are implemented for respective diagram.

7.5. Multi-lingual Code generation

This is the last module, which ultimately generate code in the different popular languages as Java, C#.Net and VB.Net to help the programmer. The generated code is structured according to the knowledge extracted in the previous modules and the UML diagrams generated..

8. Accuracy Evaluation

To test the accuracy of the diagrams generated by the designed system four parameters had been decided. Each generated diagram from each category was checked. Maximum score was declared 25. According to the wrong nominations and extractions, the points were detected. A matrix of results of generated diagrams is shown below.

Dig. Types	Objects	Attributes	Sequence	labeling	Total
Class	22	24	20	19	85%
Activity	23	21	16	20	80%
Sequence	21	22	13	18	74%
Use case	21	24	21	22	88%

Table 1 - Testing results of different UML Diagrams

A matrix representing UML diagrams accuracy test (%) for class, activity, sequence and use case diagrams has been constructed. Overall diagrams accuracy for all types of UML diagrams is determined by adding total accuracy of all categories and calculating average of it. Following graph is showing the accuracy ratio of various diagram types in terms of objects, attributes, sequence and labeling parameters.

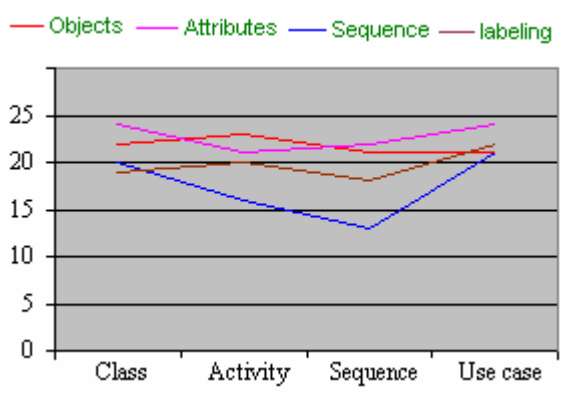

Fig 2: Accuracy Ratio of various diagrams types

9. Conclusion

This research is all about the dynamic generation of the UML diagrams and their respective code by reading and analyzing the given scenario in English language provided by the user. The designed system can find out the classes and objects and their attributes and operations using an artificial intelligence technique such as natural language processing. Then the UML diagrams such as Activity dig., Sequence dig., Component dig., Use Case dig., etc would be drawn. The accuracy of the software is expected up to about 80% with the involvement of the software engineer provided that he has followed the pre-requisites of the software to prepare the input scenario. The given scenario should be complete and written in simple and correct English. Under the scope of our project, software will perform a complete analysis of the scenario to find the classes, their attributes and operations. It will also draw the diagrams as class diagrams, activity diagrams, use-case diagrams and sequence diagrams.

An elegant graphical user interface has also been provided to the user for entering the Input scenario in a proper way and generating UML diagrams.

10. Future Work

The designed system for generating UML diagrams and their respective code in multiple languages was started with the aims that there should be a software which can read the scenario given in English language and can draw the all types of the UML diagrams such as Class diagram, activity diagram, sequence diagram, use case diagram, component diagram, deployment diagram. But last two of them component diagram, deployment diagram are still untouched.

There is also some margin of improvements in the algorithms for generating first four types Class diagram, activity diagram, sequence diagram, use case diagram. Current accuracy of generating diagrams is about 80% to 85%. It can be enhanced up to 95% by improving the algorithms and inducing the ability of learning in the system.

11. References

[1] Allen,J. (1994) *Natural Language Understanding*. Benjamin- Cummings Publishing Company, New York.

[2] Biber, D., Conrad, S., & Reppen, R. (1998). *Corpus Linguistics: Investigating Language Structure and Use*. Cambridge Univ. Press, Cambridge, U.K.

[3]Blaschke,C., Andrade,M.A., Ouzounis,C. and Valencia,A. (1999) *Automatic extraction of biological information from scientific text: protein–protein interactions*. Ismb, 60–67.

[4] C. A. Thompson, R. J. Mooney and L. R. Tang, Learning to parse natural language database queries into logical form, in: *Workshop on Automata Induction, Grammatical Inference and Language Acquisition* (1997).

[5] Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, *2*(2), 137–167.

[6] Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Mass. Chow, C., & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, *IT-14*(3), 462–467.

[7] Fagan, J. L. (1989). The effectiveness of a non-syntactic approach to automatic phrase indexing for document retrieval. *Journal of the American Society for Information Science*, *40*(2), 115–132.

[8] J. M. Zelle and R. J. Mooney, Learning semantic grammars with constructive inductive logic programming, in: *Proceedings of the 11th National Conference on Arti_cial Intelligence* (AAAI Press/MIT Press, Washington, D.C., 1993), pp. 817ñ822.

[9] Kowalski, G. (1998). *Information Retrieval Systems: Theory and Implementation*. Kluwer, Boston.

[10] Krovetz, R., & Croft, W. B. (1992). Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems*, *10*, 115–141.

[11] Losee, R. M. (1988). Parameter estimation for probabilistic document retrieval models. *Journal of the American Society for Information Science*, *39*(1), 8–16.

[12] Losee, R. M. (1996a). Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering:

An empirical basis for grammatical rules. *Information Processing and Management*, *32*(2), 185–197.

[13] Manning, C. D., & Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Mass.

[14] Maron, M. E.,& Kuhns, J. L. (1960). On relevance, probabilistic indexing, and information retrieval. *Journal of the ACM*, *7*, 216–244.

[15] Partee, B. H., Meulen, A. t., &Wall, R. E. (1990). *Mathematical Methods in Linguistics*. Kluwer, Dordrecht, The Netherlands.

[16] Salton, G., & McGill, M. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

[17] S. Weiss, C. Apte, F. Damerau, D. Johnson, F. Oles, T. Goetz and T. Hampp, *Maximizing text-mining performance, IEEE Intelligent Systems* 14 (1999) 63ñ69.

[18] Strzalowski, T. (1995). Natural language information retrieval. *Information Processing and Management*, *31*(3), 397–417.

[19] Van Rijsbergen, C. (1977). A theoretical basis for use of co-occurrence data in information retrieval. *Journal of Documentation*, *33*(2), 106–119.