# GENERATION OF USE CASE DIAGRAMS USING NATURAL LANGUAGE PROCESSING

Janani Tharmaseelan

158253L

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

February 2019

## DECLARATION OF THE CANDIDATE AND SUPERVISOR

"I declare that this is my own work and this thesis/dissertation2 does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The supervisor/s should certify the thesis/dissertation with the following declaration.

The above candidate has carried out research for the Masters/MPhil/PhD thesis/ dissertation under my supervision.

Signature of the supervisor: Date

## ABSTRACT

"***DesignPlus***" is an automated Software and Test Design tool which is capable of generate UML diagrams, User scenarios and System level Test Cases for the given software requirement specifications which is written in natural language. When it comes to software development life cycle Software Design and Testing phases are required considerable high amount of time and human effort which means it is cost to the Software project. In System designing there are some diagrams needs to be drawn and user scenarios need to be written in order to improve the clarity of requirement specification.

***DesignPlus*** is capable of generating UML diagrams (Usecase Diagram) and user scenarios for the given requirement specification in a way accelerate the Software Design phase. And also designing Test cases also need be done in the Test phase of the software life cycle in order to do proper testing on different levels. The developed system is capable of generating system level test cases for the entered functional and non-functional requirement specifications. Technology has used for the Process requirement specification is Natural Language Processing (OpenNLP). Taking altogether I call this is an Integrated Software Design Accelerator since this maintains a link between Software design and Test Design and also this accelerate the process of Software development in a way to save time and reduce cost.

## ACKNOWLEDGEMENT

We have taken many efforts in this research. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them. First, we are grateful to our project supervisor DR. Charith Chitraranjan for his kind guidance, inspiration and constructive suggestions that were helpful for me in the preparation of this project and as well as this document.

Finally, my  thanks goes to my  friends and parents who helped me  in many ways, and others who are not mentioned here, for the support and assistance provided in all the ways to make this a successful project.

Thank you.

## Table of contents

## List of figures

## List of Tables

## List of Abbreviations

| **Abbreviation** | **Description** |
|---|---|
| Xml | Extensible Markup Language |
| NLP | Natural Language Processing |
| NL | Natural Language |
| SDLC | Software Development Life Cycle |
| IT | Information Technology |
| UML | Unified Modeling Language |
| CDAP | Comprehensive Design & Analysis Project |
| SAX | Simple API for XML |
| DOM | Document Object Model |
| ID | Identification |

# 1. INTRODUCTION

## 1.1 Background Context (Literature Survey)

There are various software development approaches that are defined and designed and used during the software development process, these approaches are also referred to as “software development process models." (Such as Waterfall model, agile model and etc.) To ensure success in software development processes, each process model follows a particular life cycle. Software life cycle models describe the phases of the software life cycle and the order in which those phases are executed. There are six phases in the software life cycle model .They are

- Requirement gathering and analysis
- Design
- Implementation
- Testing
- Deployment
- Maintenance

User requirements are converted to system design, then code will be produced considering the design created from the requirements. After the implementation it will be tested, using different types of testing ways, against the requirements. When building a Software, Requirements of the Client is main thing we should consider about. Because if we build a software that does not fulfill the client requirement, obviously that is going to be rejected.

When it comes to Testing very first thing is designing test cases for the various levels of testing. Basically the test design do based on business requirements and it takes considerable amount of effort and time. Here the one of major problem I have identified is requirements can be change in the middle of implementation. So Business

Analyst has to change Requirement Specification document. After the change, every next phase have to be change accordingly. Usually this is a redundant work and again takes time and human effort. Because in each phase, changes have to be done manually.

The system I have developed mainly connect with Design and Test phases of the software life cycle with the aim of accelerating the software life cycle by automating the main task in the Design and Test phase. System consist with the five major views. Once user open up the system user first directs to the home page of the system which named as “Dashboard”. Dashboard is the place where user can create a project, get an existing project to the working panel and view recent projects. Second view is “Project” once user creates a project, they will be directed to the project view where user can enter requirement specification and generate system actors and relevant functionalities. “UML” is the third view of the system which is for generate UML diagram like Use case diagram fourth view is “Scenario” where user can generate user scenarios for the given requirement specifications fifth or final view is “Test” where user can generate test cases as functional and non-functional test cases.

## 1.2 Research Gap

Because a proper system that meets this satisfactory level cannot be found yet. There are some researches that can be found on Internet which are related to the domain of this software. Shown below is how these researches are differing from ours.

In Novel Approach to Generate Test Cases from UML Activity Diagrams done by Debasish Kundu and Debasis Samanta [5] they were using Activity diagrams to generate test cases. In this case their major setback was, when considering the activity diagrams, typically it carry a higher abstraction to the system and the functionality and shows mostly the logical overview only. Therefore the information that can be observable from activity diagrams is a bit more insufficient. Another problem is that, their approach can only be used to find faults like synchronization faults or faults in loop. It is also because of the insufficient information gatherable from an activity diagram.

And there is another major drawback identified about their approach which is, their system automates the process to derive test cases from activity diagrams but when it comes to designing those activity diagrams they have lost the idea of automating a process. So when the user has to design an activity diagrams in order to derive the test cases, the user could get confused like why I have to design an activity diagram to get my test cases when I can design test cases meanwhile. These mentioned problems will not occur in the proposed system because first, when I derive regression level test cases I am not relying on class diagrams alone or requirement specification alone, but I insist on using the both requirement specification and use case diagrams so it'll cover a larger area of testing and the accuracy of the test cases will be at a higher level since those are based on both requirement specification and use case diagrams.

The DrawPlus Application's NLP core will have following functionalities

- Actor identification

This process is to identify the actors out and is defined under NLPCore module. With the help of OpenNLP API, the Actor is taken out analyzing the requirements given. The extractions is done by a large scale algorithm and a set of pre-defined rules.

- Function identification

Belongs in the Major functionality criteria in NLPCore. Functions are referring to the actions executed by each identified actor and are extracted according to the actor's references with correspond to the pre-defined rules.

- Advance co-referencing

This functionality comes in the effort to maintain a higher accuracy to the output. The textual content will scored for the Co-referensive expressions. The co-reference are extracted with different set of rules.

- UML structure generation

The extracted actors and functions are passed into a XML file for further references. Co-relation of each function and actor is also mapped into the XML which will be used for the UML diagram generation section.

- Improved Accuracy

Rich Algorithms combined with state-of-the-art technology enables the Draw Plus to enhance & reach a higher level of accuracy.

- Improved Efficiency

Each element in the given requirement is process by a complex large-scale algorithm yet the unnecessary processing's are avoided to improve overall performance and efficiency. The more accuracy the software produce, the higher the complexity it gets.

## 1.3 Research Problem

There are so many problems with some existing software and manual drawing.

• Time Consuming

• Hard working

• System requirements may change quickly without informing

• Lengthy Requirements are difficult to read and remember

• It's hard to find out one tool or software that can draw UML diagrams, user Scenarios and test cases.

• Cost effective

Manual processing of reading requirements is very time consuming process. It takes more time to update starting from first phase to last phase of software development process when requirements are changing in the middle. In the IT industry business analysis draws UML diagram in design phase by reading and remembering special points which is mentioned in the requirement document. And also software testers re-read that document and create test cases as well as the user scenarios in the testing phase. Currently the Agile methodology is used from world popular software company to small work group. In that process user scenarios are the basic inputs to create test cases. User scenarios are that much important. And requirements are updated or added from sprint to sprint in this methodology. Updating requirements are normally happed in this process. In order to support that, diagrams and other stuffs should be modernized. Reading same document by few people in several times is time consuming. Another problem is in certain situation client may request to change requirements, In that case according to those changes diagrams, user scenarios and test cases should be updated. Nowadays there are some websites, software available for the drawing Use case diagrams and user scenarios separately. To use those tools business analyst need to have clear understanding in the given description before he/she draw diagram. It is rare to find out a system to find, which can draw them automatically. The problem with the currently available system is the accuracy level of the output as well as there is difficult to find a system to develop user scenarios. If available it can be provide a template to create user scenario instead of generating it. Regression level test cases are more essential part of the testing phase. Why because it is started as a first level of testing for a new software. Software development group may have to create test cases mean while developing the software. The huge problem is managing resources to do both tasks. While starting the development of software, the requirements might be changed by client with day to day updates.

DrawPlus will basically works as an accelerator between the software requirement specification and the relevant stakeholders like business analysts, test case designers and requirement analysts. During this process a considerable amount of time and resources are being spent by organizations because this process provide groundwork for almost every other phases and process of the software development lifecycle. The main research problem that I considered in this research is to develop a system that can accelerate this process at least up to 75% with rich algorithms to enhance and provide the outcome with a higher accuracy.

Other problem that was identified during the research time was, even though the all test case designs are based on system requirement specifications there is no such system which maintain a link between requirement specification and test cases. Reason that I have to have a link between requirement specification and test cases is in any phase of software life cycle,  requirement specification can be changed and if those changes were not properly communicated to the test designers there can be a miss match between the requirement specification and written test cases.

## 1.4 Research Objective

### 1.4.1 Implement an algorithm to match a set of certain grammar

The algorithm takes a set of grammars and text phrase and searches for a possible match. All these grammar can be identified by the rule number. The findings of the algorithm is the rule number. Then the rule number is passed into particular methods to acquire actors and functions.

- Implement algorithms to identify Actors and Functions

  Two algorithms are implemented for extracting actor and extracting function. The inputs for these methods are the output of the grammar matching algorithm.
  When the grammar number is passed the algorithm can the run necessaries to filter out actor and their relevant action.

- Improve accuracy of the NLPCore results

  The algorithm is not only producing an output but it has a greater accuracy. Accuracy is gained through having more effective filtrations inside these algorithms.
  The number of grammar rules is another factor that improves accuracy.
  But having more and more grammars will eventually reduce performances and therefor these factors are kept well balanced.

### 1.4.2 Develop a tool which Provide Assistance to software test designers

*DrawPlus* is well formatted to gather requirements specification as functional requirement and non-functional requirements and Testers can generate their test system level test cases as functional and non-functional test cases.

- Accelerate the Test design process in a way reducing considerable amount of resources and time.

Basically when designing test cases based on requirement specifications tester has to get a good idea about the system by reading the whole system requirement specification document. Then only tester can start designing test cases.

- Improve the reliability and the accuracy of the test design.

Accuracy is one of the main key when it comes to an automated system, it does not matter how easily or how fast the system works unless it gives accurate results. *DrawPlus* is a NLP (Natural Language Processing) based system which process Requirement specification written in natural language in to UML, User scenarios and Test Design. As developer of the system I use NPL techniques with my own algorithm in a way producing most accurate outputs.

- Reduce huge redundant works which occur due to requirement changes.

By this Objective once a user, generate test cases from the given requirement specification and after generating it if any changes happened in requirement specification, this system is capable to show how the changes effect to the already generated test cases.

### 1.4.3 Designing a tool to assist software testers

Improve a reliable and accurate system which easily can be generate use case diagram. So no need to waste more time reading and remembering things. Just users can enter the requirements and generate the diagram

- Develop an algorithm to draw actors and functions for Use case diagram in a certain area
- If Number of actors and function count is more than three then automatically change the side of actors and functions.
- Software testers/business analysts will be benefitted through the performances of this tool
- Recognition of the relationship between Actors and Functions
- Draw the functions and write the function name in the middle of the circle. When the function name is longer than the Circle height then the function name automatically write into two lines.
- Background of the diagram will be automatically adjusted according to the space it required.
- Defining specific rules for Use case diagrams referring specific requirements

# 2 METHODOLOGY

## 2.1 Methodology

The research is based on developing a tool for software design and testing phase.

I. Process Natural Language.

• Develop an algorithm which detects system Functionalities from given requirement (Text) with

Highest accuracy.

• Develop an algorithm which detects system Actors from given requirement

• Identify the relationships among detected system Actors and Functions

II. Generate UML Diagrams

• Generate a UML diagrams (Usecase diagram) for the system generated file (Xml).
• Develop an algorithm to draw actors and function avoiding overlapping on top of one.
• Identify relationship and rules using given requirement (Text).

III. Generate User Stories

- Generate a User scenario for the system generated file (Xml).
- Detect conditions and steps for the scenarios using given requirement (Text).

DrawPlus provide a desktop application to achieves above mentioned goals. The main purpose of this smart software is to maintain the software projects more efficiently and effectively without involving or less involving IT related people. Software companies can reduce lots of cost and time by using this system. This product can provide the more accuracy diagram and user scenarios to test any software product by using client documents, requirements etc. And also it has ability to create test cases as well. Currently there is many more software, web application to draw, generate UML diagram.

UML is a one of website which is used to generate use case diagram. But the problem with that is it is not intelligent. User has to provide keywords to draw diagram. The Lucidchart [2] is another type of web application which can be edited as drawer opinion.

Even though there are many software available, Drawplus has special place from all other software. It is a user-friendly one. Where from one button click use case diagram is generated. From few steps user scenarios, test cases are generated. That is the uniqueness of this product.

Figure 3.1 system diagram shows the main flow of this product. How it will work. Normally requirements will be referred by all the parties. Requirements and extracted data then will be passed as an input to generate use case diagram. Requirements, extracted data output and Use case diagram will be passed as an input for generating user scenarios. Requirements, extracted data, use case diagram and user scenario will be then taken as an input to generate test case.

Manual process is following these steps during SDLC

- Read and understand software requirements
- Identify actors, functions and relationship(Ex: Include) between functions
- Draw Usecase diagram

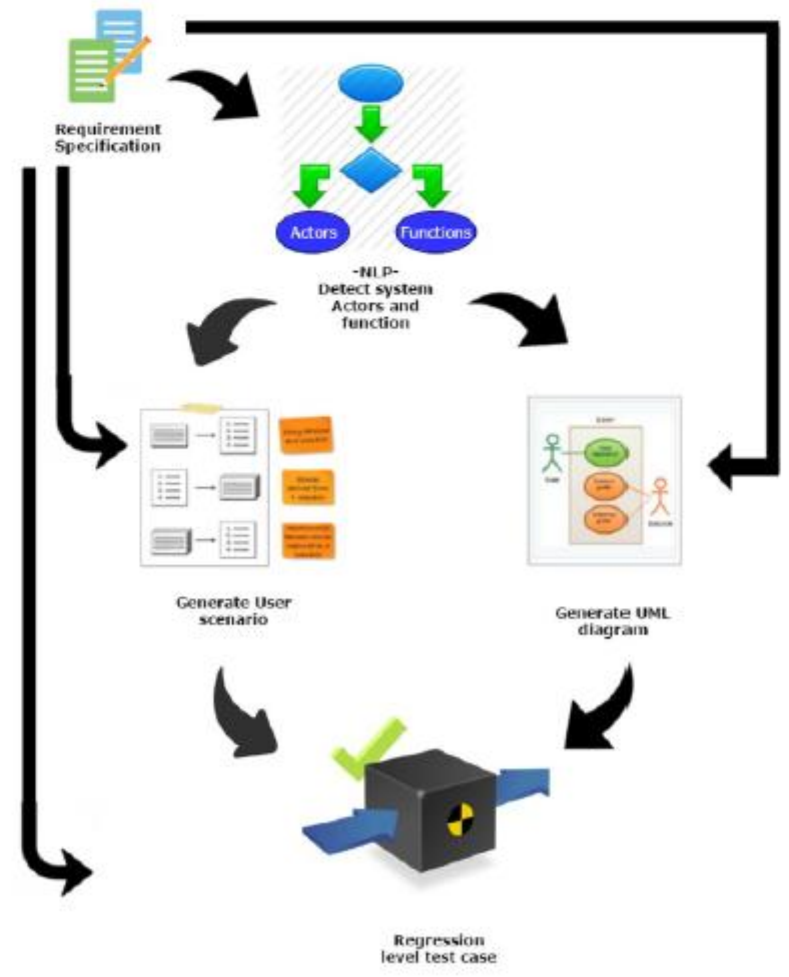

Requirement Specification
Actors
Functions
-NLP- Detect system Actors and function
Generate User scenario
Generate UML diagram
Regression level test case

*Figure 2-1 System Diagram*

All these steps are done by this tool. As a first input, it will read and study the requirements using Natural Language processing. Several xmls are created according to the each NLP process functions. The entire xml file has same format which can be read by SAX Parse or DOM parser. Figure 3.1 is representing sample xml file. Mainly there are three xml files are produced. One of them is used as an input for use case diagram which is providing all the details like actors, functions, functions name, reference number and number that need to draw the diagram. Another xml file is proved in details to create user scenarios. And last one to build test cases. It should have more description details about requirements since generating test case need to add some more text by additional. Where it has to identify relationship among the functions.Simply those two terms can be call as include and extend.

DrawPlus introduces a beneficial way to accelerate any IT organization’s business outcome by reducing long hours spent on Design and testing phase of the SDLC. The project has a considerable amount of research in the field of Natural Language Processing (NLP). The Entire DrawPlus application including the NLPCore is developed in JAVA combining existing OpenNLP libraries with set of algorithms newly developed. The algorithms are ensuring whether the input is complying with certain set of rules defined in the NLPCore. These rules are combined with pattern matching mechanism and there comes algorithms which are highly complex & advanced.

**Apache OpenNLP**

The groundwork for the Natural Language Processing functionalities in DrawPlus applications is provided by OpenNLP library released by Apache. The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co-reference resolution. These tasks are usually required to build more advanced text processing services. OpenNLP also includes maximum entropy and perceptron based machine learning. In the DrawPlus System, the most frequently used NLP functions are Sentence Detector, Tokenizer and POS tagger. These functions are used to derive and extract relevant information from a given input for further processing purposes. The extracted information will forwarded through advanced algorithms throughout the entire process.

Shown in the Table below is the NLP functions that are used in the application, further describing how they have applied

*Table 1 :NLP functions*

| Method Name | Description | Use |
|---|---|---|
| Sentence Detector | As in the method name mentions, sentence detector is for detecting sentence boundaries. Capable of detecting sentences lie throughout multiple lines | The initial step in the application is detecting sentences. The input is process sentence by sentence which enable the program to find and map the relation each section has, when it's needed. The input is divided into separate sentences. |
| Tokenizer | A given input text will be separated into tokens. Normally sentences are separated into works by using spaces, but in some case it is different. For an example if we take "isn't " it should be separated as "is " and "not" since it a brief format of "is not" | The sentences divided by the sentence detector are then divided again into small fractions call token. Tokenizing enables the program to go through each word and inspect them existence of an Actor. If an actor is found then any actions executed by the actor |
| POS tagger | Any text runs through the POS tagger will be given a tag at the end of each word. This helps to identify the content of a given text such as noun, verb , adjective etc. | POS tagger performs the magic inside the program. In order to apply the algorithms with grammar rules, there exist some positional parameters that needs to be identified with in the text input. With the entire sentence is tagged |

|  |  |  |
|---|---|---|
|  |  | by the POS tagger, the program sees where lies the Nouns, verbs, Prepositions , Modal etc. |

**File Structure**

In order for the DrawPlus to work properly, as all other regular applications The DrawPlus has to save some data for system operations. When a user is working in a project the user might want save the progress for resume the work in a later time. Therefor the system has to maintain some mechanism to save the progress of both application and the user’s work. For this purpose I defined a file structure. There are two main file structures,

• System file structure

• Project specific file structure

The system file structure is verified when the system is starting up. It contains recent projects and application usage logs. If the file verification fails the system will get terminated with giving an error message detailing about the fault. The project file structure is verified only when an existing project has loaded into the application. If the relevant files are not found in the project folder the project will not open up and prompt an error message to the user. In both file structure text data will be saved under XML file format. For as a better security mechanism the project specific XMLs are always kept encrypted.

### 2.1.1 **Addressing literature**

There are similar kinds of applications available in the world. The uniqueness of proposed system will be described here. The rules must have access to some level of syntactic knowledge, at least distinguishing major syntactic constituents such as subject, verb, and direct object. Otherwise CRYSTAL could not tell who is the Person In and who is the Person Out in \Mr. A succeeds Mr. B as chairman" and \Mr. B succeeds Mr. A as chairman". The rules will also need to is distinguish affirmative from negative phrases and distinguish active from passive verbs [6]. This document mentions that they can only access to some level of syntactic knowledge. It means that the rules are defined to identify Subject, Verb, and Object formats sentence only. When considering about UML diagrams these rules are not enough. Need to identify and define more rules to display <<include>> and <<extend >> relationships. The proposed system will consider all the factors and give good results to draw use case diagram.

**Natural language processing based automated system for UML diagrams generation[7]**

This research paper has proposed natural language processing based automated system for generating UML diagrams by analyzing input text. Their system extracting relative and required information by analyzing natural language which is written in simple English in a few sentience. They have illustrated their methodology of extracting relative and required information using an example as bellow.

*"Zia is playing with the red ball."*

| **Lexicons** | **Phase-I** | **Phase –II** |
|---|---|---|
| Zia | Noun | Object |
| Is | Helping-Verb | ………. |
| Playing | Verb | Method |
| With | Preposition | ………. |
| The | Article | ………. |
| Red | Noun | Attribute |
| Ball | Noun | Object |

In the mentioned research in order to find the ability to read and understand as well grab the desired information a rule based algorithm is designed. Normally for a language a grammar is necessary and it contains verbs, nouns, adjectives and etc. So those will be first extracted and then from the extracted details further processing will be done. They have done really a good methodology to demonstrate the research problem and solution, but further they have mentioned that their one not in 100% accuracy level and it can be improved further by using different algorithms.

### 2.1.2 Project development process

I decided to follow the agile methodology [02] for the development process. The Reason why it is selected this methodology is because this project has a higher risk since it is based on Natural Language Processing which is still under research level. And the other reason is that the outcome of this system required to reach a degree of accuracy more than 70% as I assumed. As according to the agile methodology this project is divided into small cycles known as sprint with a specific durations is given for each.

### 2.1.3 Feasibility study

I went through an analysis since this is a complex system I had to go via a feasibility study.

When a feasibility study is carried out, there are four main areas of consideration

- Technical => Is it technically possible to build up the system?
- Financial =>Will Carrying this out it financially possible ?
- Organizational => Will the new system works from the existing system?
- Ethical => Is it have any ethical Issues?

In order to answer these questions doing a feasibility study is important.

In technical feasibility study, when considering drawing part, main challenge was positioning actors and functions without replacing and overlapping. In every small point had to be managing with accurate X and Y location values. Reading XML document and only get most useful data was another area that tested in this section. At this point I developed a rich algorithm to avoid all the problems mention early.

## 2.2 Testing And Implementation

### 2.2.1 **Software implementation**

Drawplus is a desktop application. It should have user friendly interfaces. After brainstorming I came up with this interface which anyone can easily use to it. By doing the literature review I came up with most suitable tools and technologies. While designing the NLPCore, due to its higher level of complexity, the module is considered to be implemented as sub modules rather developing the entire module as one. The Figure   shows a high level abstract view of the NLPcore functionality, where the methods are been used for dependencies each sub module to one another. The sub modeling methods also allowed me the ease of updating & upgrading of the system. Means that any additional phase, lines of code, sub- modules can be added to the system later without much fuzz. With the sub modules, behavior of some larger and complex algorithms can be simply for better understating.

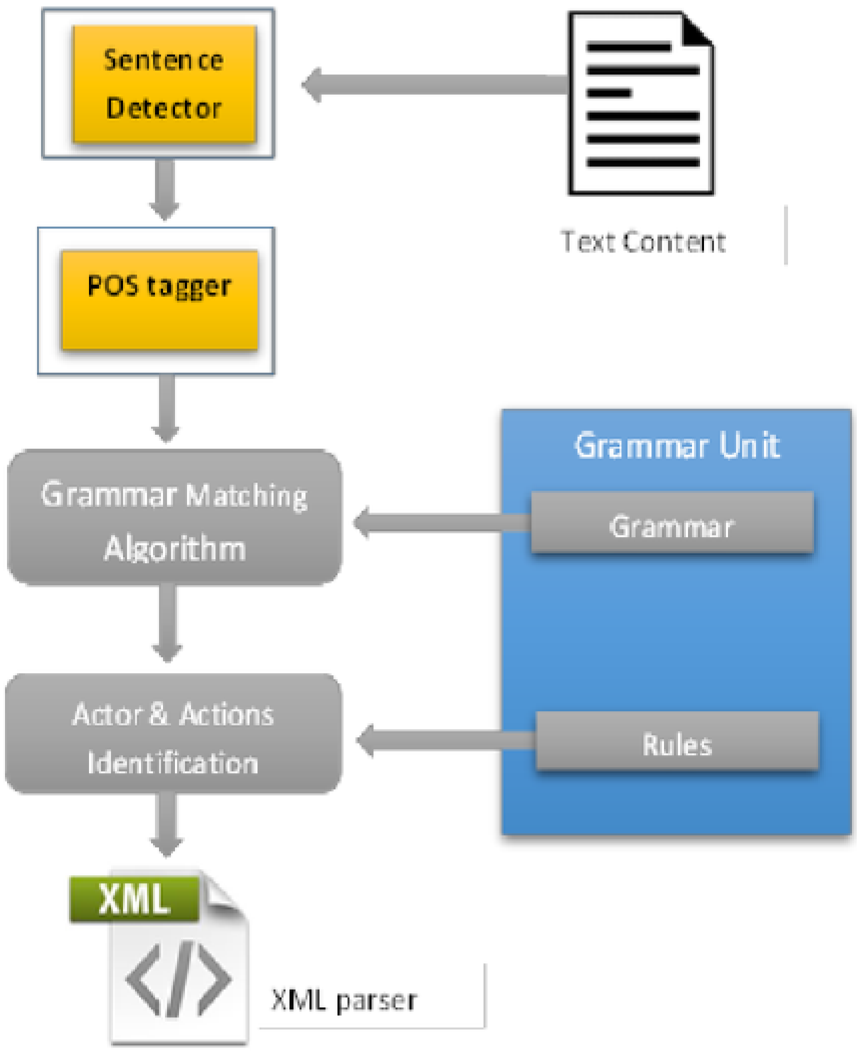


*Figure 2-2 NLP Core Functionality*

Grammar matching algorithm was one of a significant achievement in the NLPCore. The reason is, the grammar matching algorithm provides more of a framework for all of the parts where need to deal with Grammar matching. With this algorithm it was possible to define own way of grammars and check whether the grammar exists in a text content by parsing the grammar and the text phrase to the algorithm.

Bellow figure shows the simplified view of the grammar matching algorithm. The basic loops and conditions.

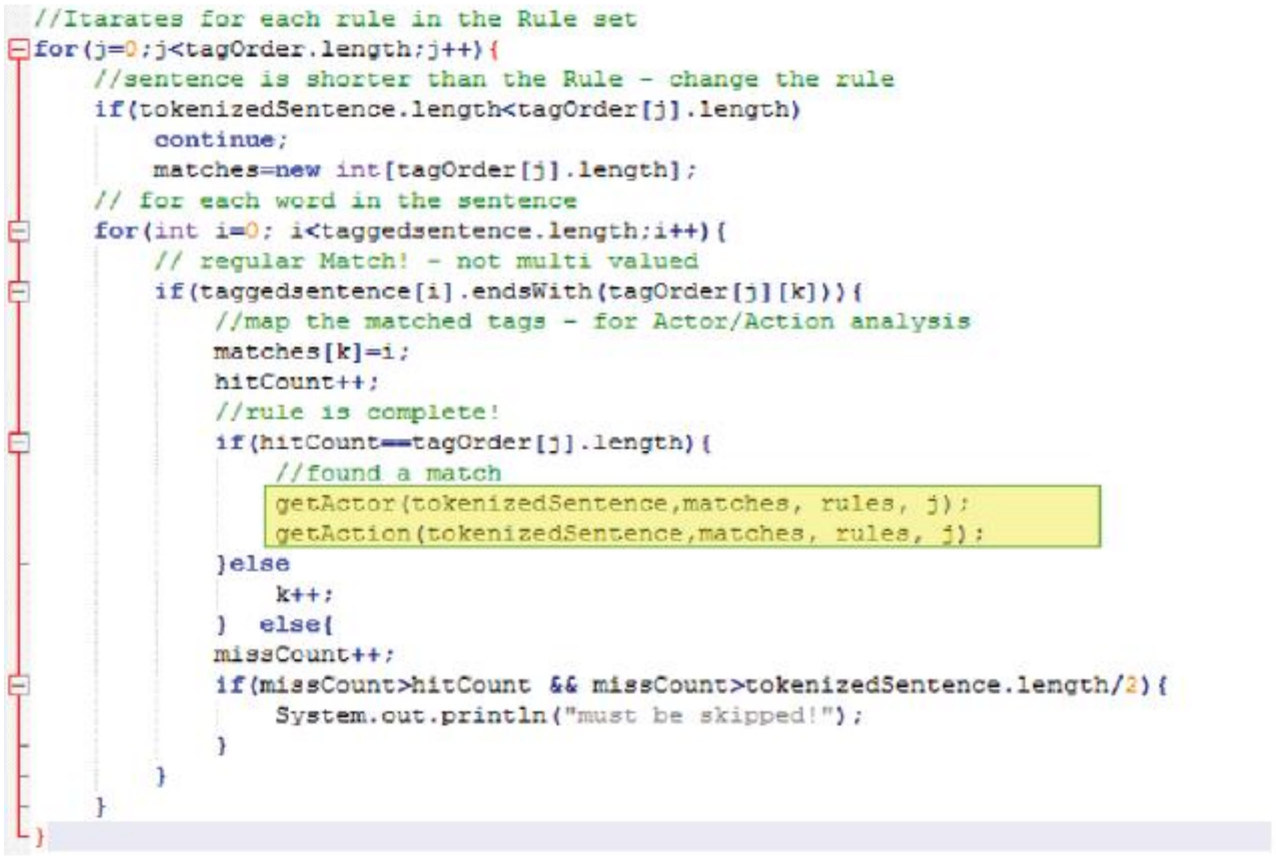
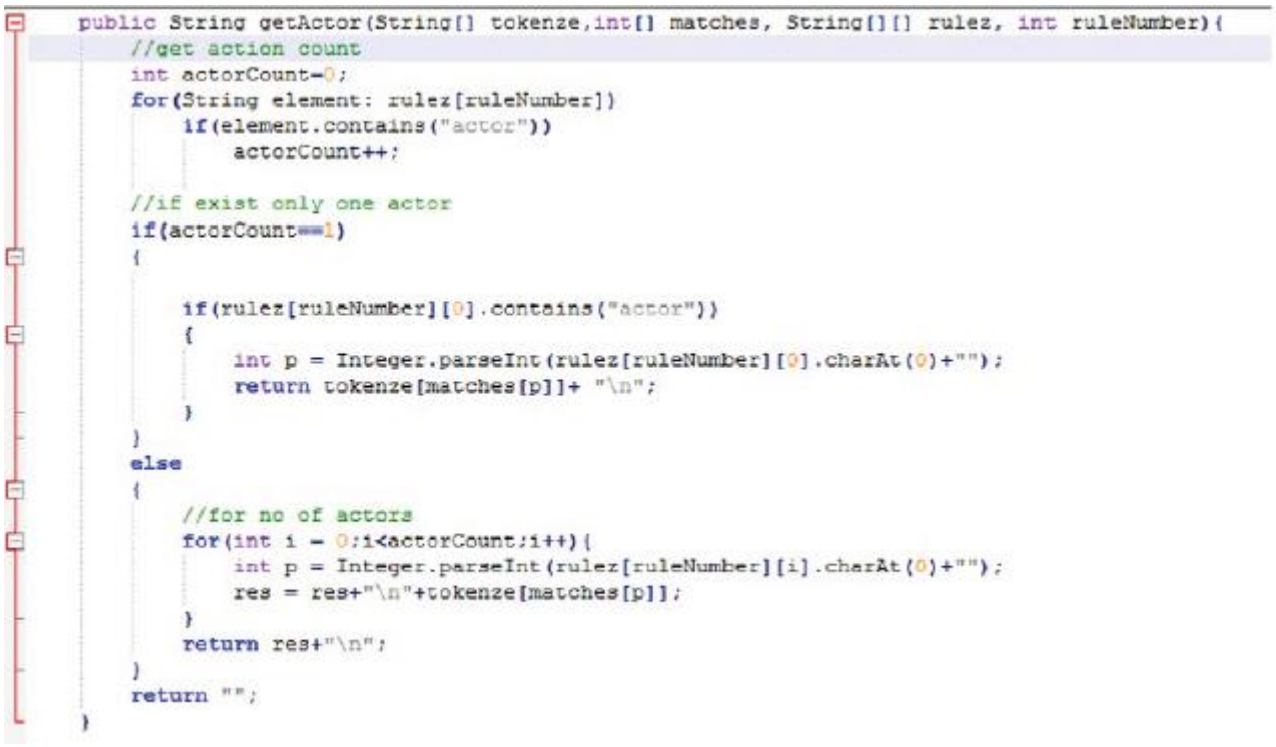

```
//Itarates for each rule in the Rule set
for(j=0;j<tagOrder.length;j++){
    //sentence is shorter than the Rule - change the rule
    if(tokenizedSentence.length<tagOrder[j].length)
        continue;
        matches=new int[tagOrder[j].length];
    // for each word in the sentence
    for(int i=0; i<taggedsentence.length;i++){
        // regular Match! - not multi valued
        if(taggedsentence[i].endsWith(tagOrder[j][k])){
            //map the matched tags - for Actor/Action analysis
            matches[k]=i;
            hitCount++;
            //rule is complete!
            if(hitCount==tagOrder[j].length){
                //found a match
                getActor(tokenizedSentence,matches, rules, j);
                getAction(tokenizedSentence,matches, rules, j);
            }else
                k++;
            }  else{
            missCount++;
            if(missCount>hitCount && missCount>tokenizedSentence.length/2){
                System.out.println("must be skipped!");
            }
        }
    }
}
```

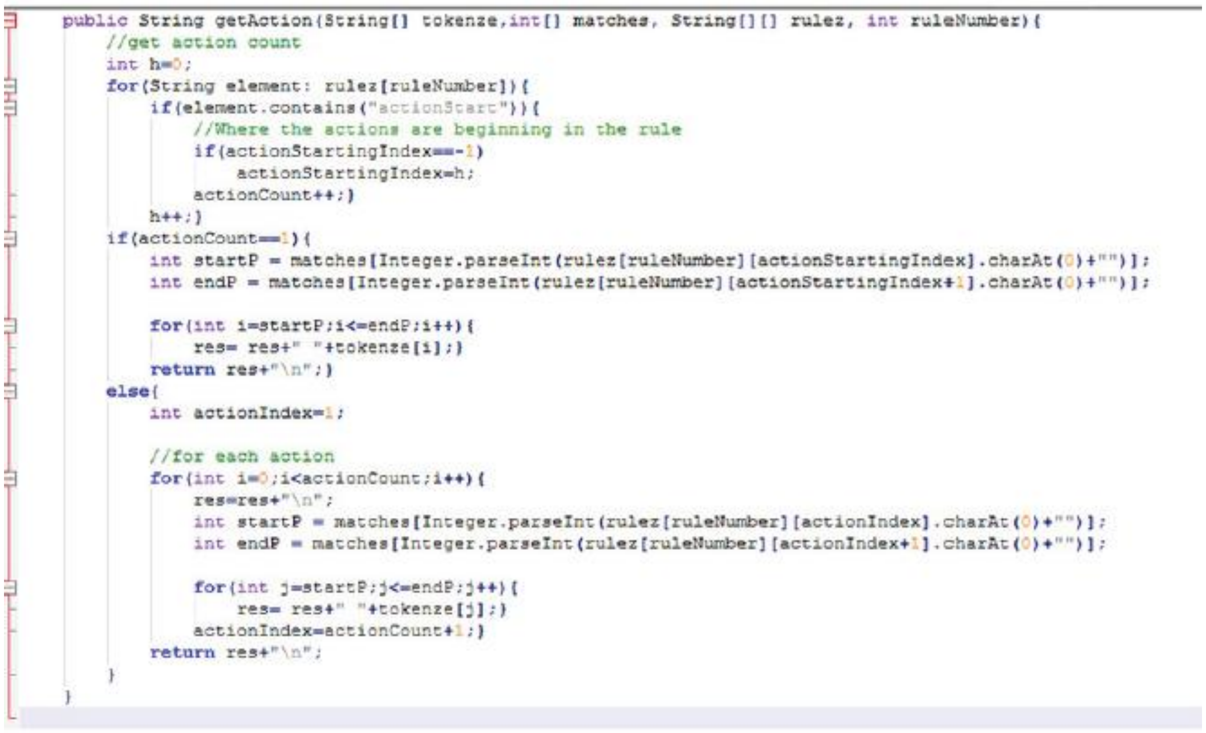
*Figure 2-3 Grammar matching algorithm*

Figure 3.4 shows in the highlighted area, where the getActor() and getAction() methods are been called. The figure 3.5 shows a simplified view of the getActor() method. After a rule has matched, the getActor() method is called. The matched rule no is an input to this function. Where it refers to the matched grammar and the respective rule and returns a string if the actor is found. After the actor has returned the getAction() is called to get the actions executed by the respective actor. Which also takes the matched rule number as an input. The figure 3.5 and figure 3.6 show an over simplified view into the above mentioned functions whereas Figure 3.5 is the getActor() method and the Figure 3.6  is getAction() method.



```java
public String getActor(String[] tokenze,int[] matches, String[][] rulez, int ruleNumber){
    //get action count
    int actorCount=0;
    for(String element: rulez[ruleNumber])
        if(element.contains("actor"))
            actorCount++;

    //if exist only one actor
    if(actorCount==1)
    {

        if(rulez[ruleNumber][0].contains("actor"))
        {
            int p = Integer.parseInt(rulez[ruleNumber][0].charAt(0)+"");
            return tokenze[matches[p]]+ "\n";
        }
    }
    else
    {
        //for no of actors
        for(int i = 0;i<actorCount;i++){
            int p = Integer.parseInt(rulez[ruleNumber][i].charAt(0)+"");
            res = res+"\n"+tokenze[matches[p]];
        }
        return res+"\n";
    }
    return "";
}
```

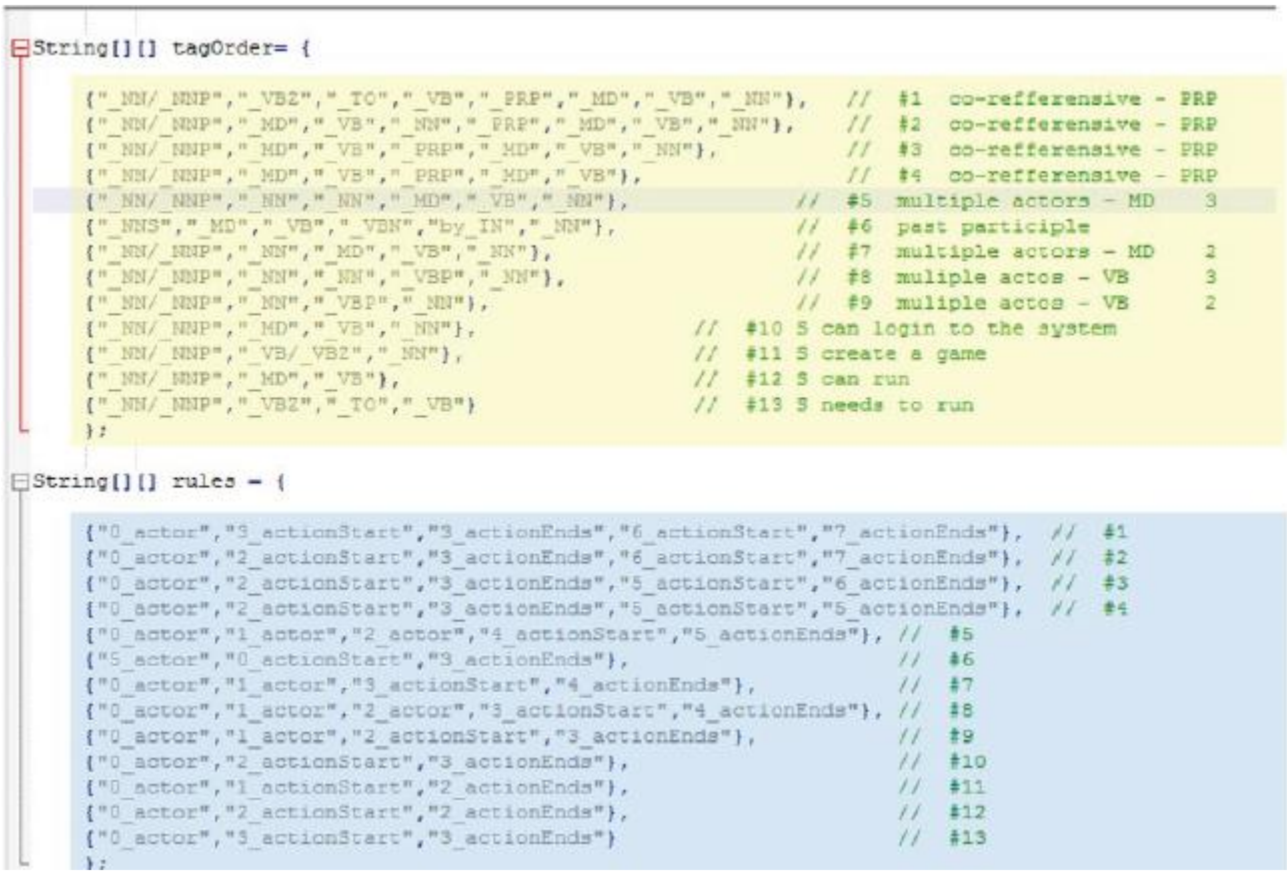
*Figure 2-4 getActor() method*



```java
public String getAction(String[] tokenze,int[] matches, String[][] rulez, int ruleNumber){
    //get action count
    int h=0;
    for(String element: rulez[ruleNumber]){
        if(element.contains("actionStart")){
            //Where the actions are beginning in the rule
            if(actionStartingIndex==-1)
                actionStartingIndex=h;
            actionCount++;}
        h++;}
    if(actionCount==1){
        int startP = matches[Integer.parseInt(rulez[ruleNumber][actionStartingIndex].charAt(0)+"")];
        int endP = matches[Integer.parseInt(rulez[ruleNumber][actionStartingIndex+1].charAt(0)+"")];

        for(int i=startP;i<=endP;i++){
            res= res+" "+tokenze[i];}
        return res+"\n";}
    else{
        int actionIndex=1;

        //for each action
        for(int i=0;i<actionCount;i++){
            res=res+"\n";
            int startP = matches[Integer.parseInt(rulez[ruleNumber][actionIndex].charAt(0)+"")];
            int endP = matches[Integer.parseInt(rulez[ruleNumber][actionIndex+1].charAt(0)+"")];

            for(int j=startP;j<=endP;j++){
                res= res+" "+tokenze[j];}
            actionIndex=actionCount+1;}
        return res+"\n";
    }
}
```

*Figure 2-5 getAction() method*

Shown in the Figure 3.7 is a sample set of grammars and rules as it is applied inside the application. The orange shaded highlight section shows the grammars while the blue shaded highlight section shows the rule for the respective grammar. These two are highly co-related to one another. For an instance if the grammar matching algorithm get a correct hit at grammar no #5 in the Grammar section, the rule to be applied in order to derive the actors & functions are mentioned in the Rules sections. After finding the correct match of a grammar, the algorithms shown in Figure 3.5 and Figure 3.6 will use the rules shown in Figure 3.7 to extract the correct output.



```
String[][] tagOrder= {
    {"_NN/_NNP","_VBZ","_TO","_VB","_PRP","_MD","_VB","_NN"},    //  #1  co-refferensive - PRP
    {"_NN/_NNP","_MD","_VB","_NN","_PRP","_MD","_VB","_NN"},     //  #2  co-refferensive - PRP
    {"_NN/_NNP","_MD","_VB","_PRP","_MD","_VB","_NN"},           //  #3  co-refferensive - PRP
    {"_NN/_NNP","_MD","_VB","_PRP","_MD","_VB"},                 //  #4  co-refferensive - PRP
    {"_NN/_NNP","_NN","_NN","_MD","_VB","_NN"},              //  #5  multiple actors - MD    3
    {"_NNS","_MD","_VB","_VBN","by_IN","_NN"},               //  #6  past participle
    {"_NN/_NNP","_NN","_MD","_VB","_NN"},                    //  #7  multiple actors - MD    2
    {"_NN/_NNP","_NN","_NN","_VBP","_NN"},                   //  #8  muliple actos - VB      3
    {"_NN/_NNP","_NN","_VBP","_NN"},                         //  #9  muliple actos - VB      2
    {"_NN/_NNP","_MD","_VB","_NN"},                  //  #10 S can login to the system
    {"_NN/_NNP","_VB/_VBZ","_NN"},                   //  #11 S create a game
    {"_NN/_NNP","_MD","_VB"},                        //  #12 S can run
    {"_NN/_NNP","_VBZ","_TO","_VB"}                  //  #13 S needs to run
    };

String[][] rules = {
    {"0_actor","3_actionStart","3_actionEnds","6_actionStart","7_actionEnds"},  //  #1
    {"0_actor","2_actionStart","3_actionEnds","6_actionStart","7_actionEnds"},  //  #2
    {"0_actor","2_actionStart","3_actionEnds","5_actionStart","6_actionEnds"},  //  #3
    {"0_actor","2_actionStart","3_actionEnds","5_actionStart","5_actionEnds"},  //  #4
    {"0_actor","1_actor","2_actor","4_actionStart","5_actionEnds"}, //  #5
    {"5_actor","0_actionStart","3_actionEnds"},                     //  #6
    {"0_actor","1_actor","3_actionStart","4_actionEnds"},           //  #7
    {"0_actor","1_actor","2_actor","3_actionStart","4_actionEnds"}, //  #8
    {"0_actor","1_actor","2_actionStart","3_actionEnds"},           //  #9
    {"0_actor","2_actionStart","3_actionEnds"},                     //  #10
    {"0_actor","1_actionStart","2_actionEnds"},                     //  #11
    {"0_actor","2_actionStart","2_actionEnds"},                     //  #12
    {"0_actor","3_actionStart","3_actionEnds"}                      //  #13
    };
```

*Figure 2-6 rules*

User can able to create a project and after the successful creation of the project user is automatically redirected to the project section where user can provide a scenario for the project. The same description will mostly be the main input to the system.

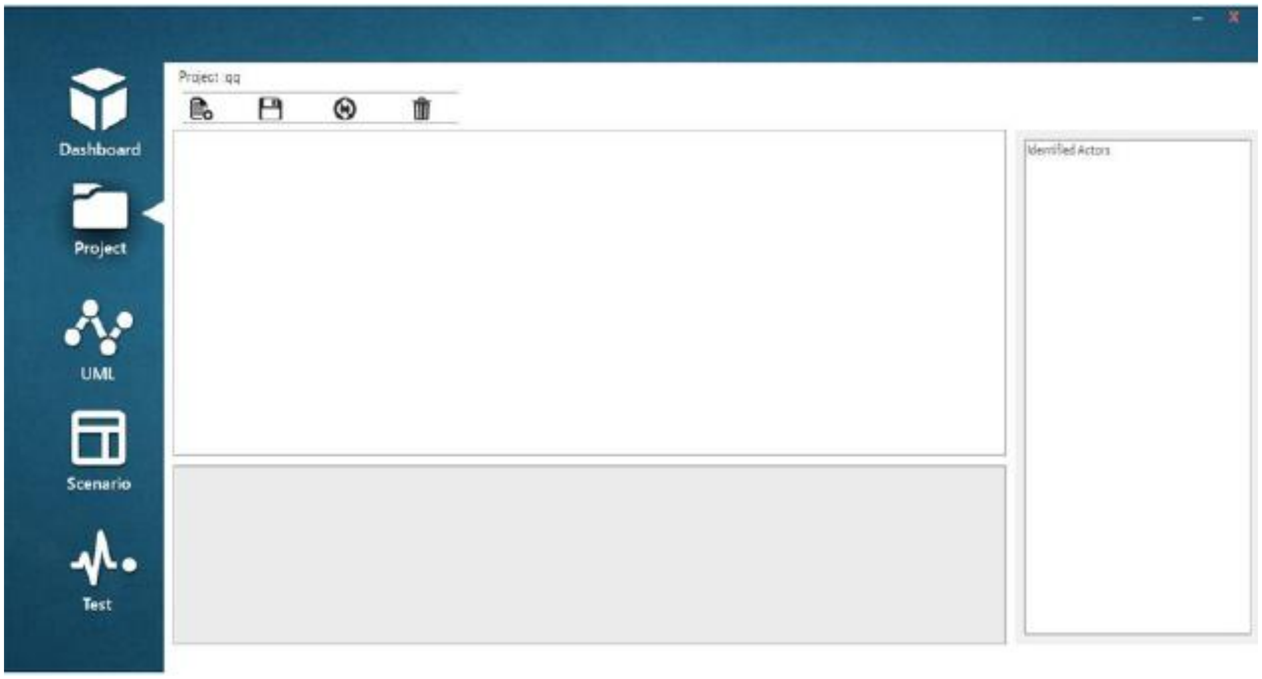


*Figure 2-7 Interface Home*

User can import a scenario directly into the system from a file or from the hard disk or from web. Allowed file types are (.doc / .docx  /.txt)

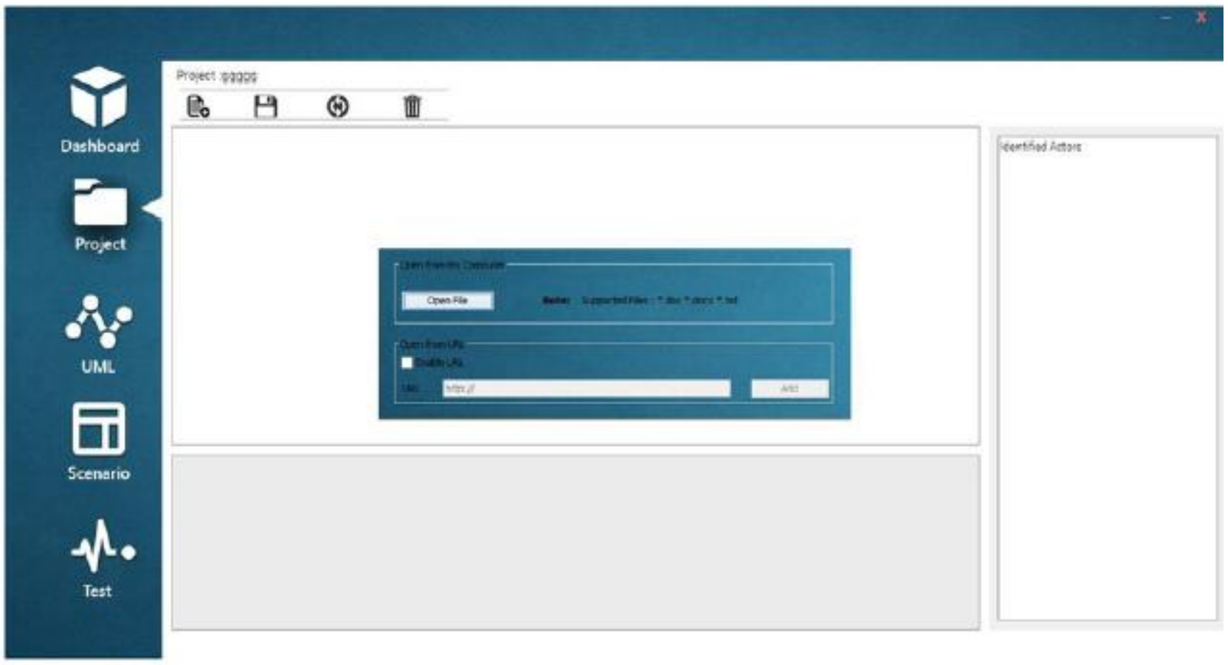
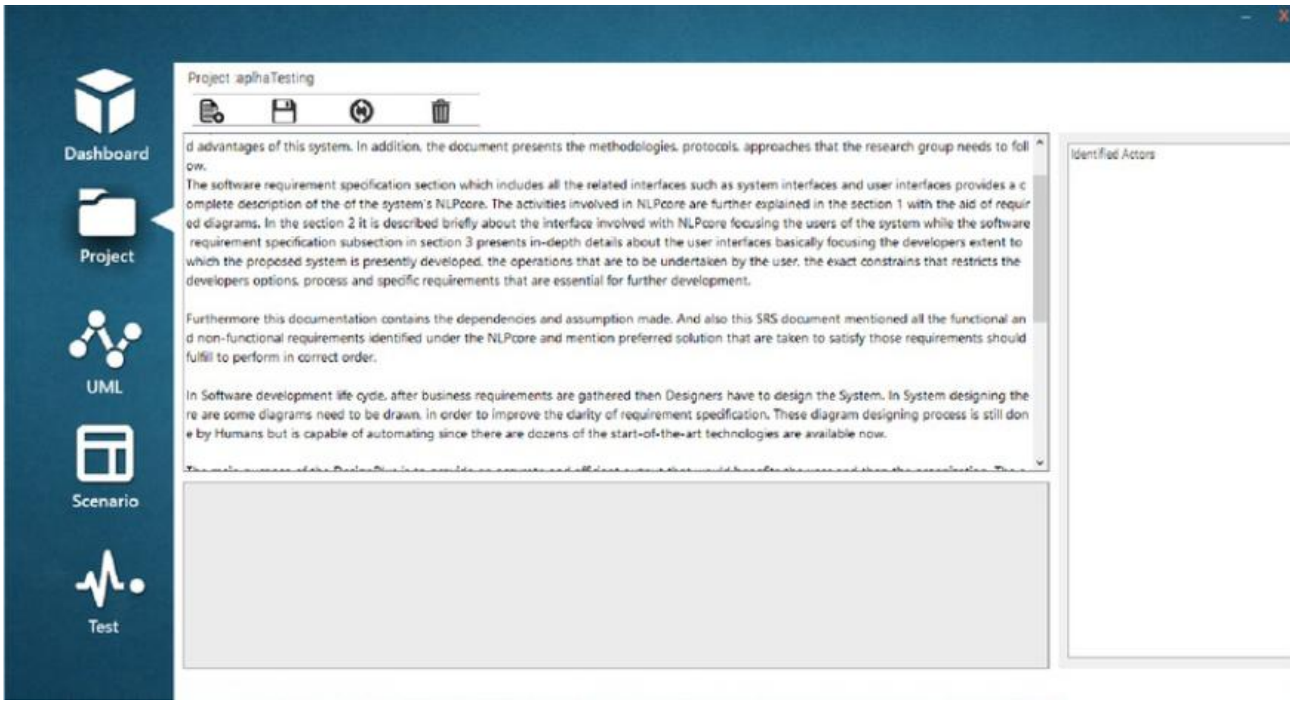


*Figure 2-8 : Interface for Browse*

A successful import will parse the file and the text content will be loaded into the screen for analyzing or further modifications.



Project :aplhaTesting

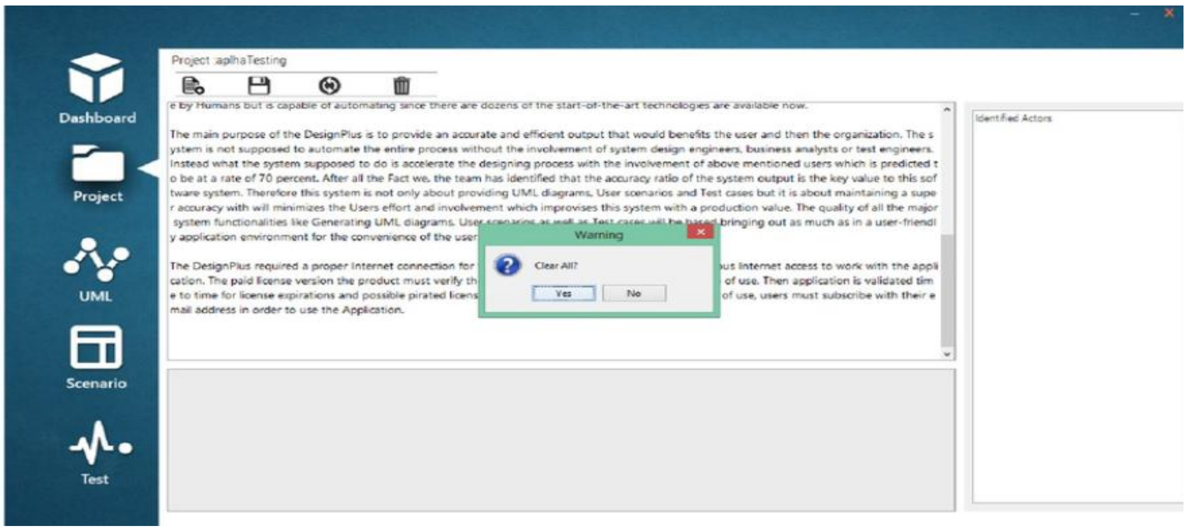
d advantages of this system. In addition, the document presents the methodologies, protocols, approaches that the research group needs to foll ow.
The software requirement specification section which includes all the related interfaces such as system interfaces and user interfaces provides a c omplete description of the of the system's NLPcore. The activities involved in NLPcore are further explained in the section 1 with the aid of requir ed diagrams. In the section 2 it is described briefly about the interface involved with NLPcore focusing the users of the system while the software requirement specification subsection in section 3 presents in-depth details about the user interfaces basically focusing the developers extent to which the proposed system is presently developed, the operations that are to be undertaken by the user, the exact constrains that restricts the developers options, process and specific requirements that are essential for further development.

Furthermore this documentation contains the dependencies and assumption made. And also this SRS document mentioned all the functional an d non-functional requirements identified under the NLPcore and mention preferred solution that are taken to satisfy those requirements should fulfill to perform in correct order.

In Software development life cycle, after business requirements are gathered then Designers have to design the System. In System designing the re are some diagrams need to be drawn, in order to improve the clarity of requirement specification. These diagram designing process is still don e by Humans but is capable of automating since there are dozens of the start-of-the-art technologies are available now.

*Figure 2-9 Content Loaded Interface*

User can clear out the scenario from the system after the confirmation of the action form user. When successfully removed, notification will be prompted in the bottom line of the screen as shown in the second figure. User can still recover the content by simply pressing undo keys or redo keys.



Project :aplhaTesting

e by Humans but is capable of automating since there are dozens of the start-of-the-art technologies are available now.

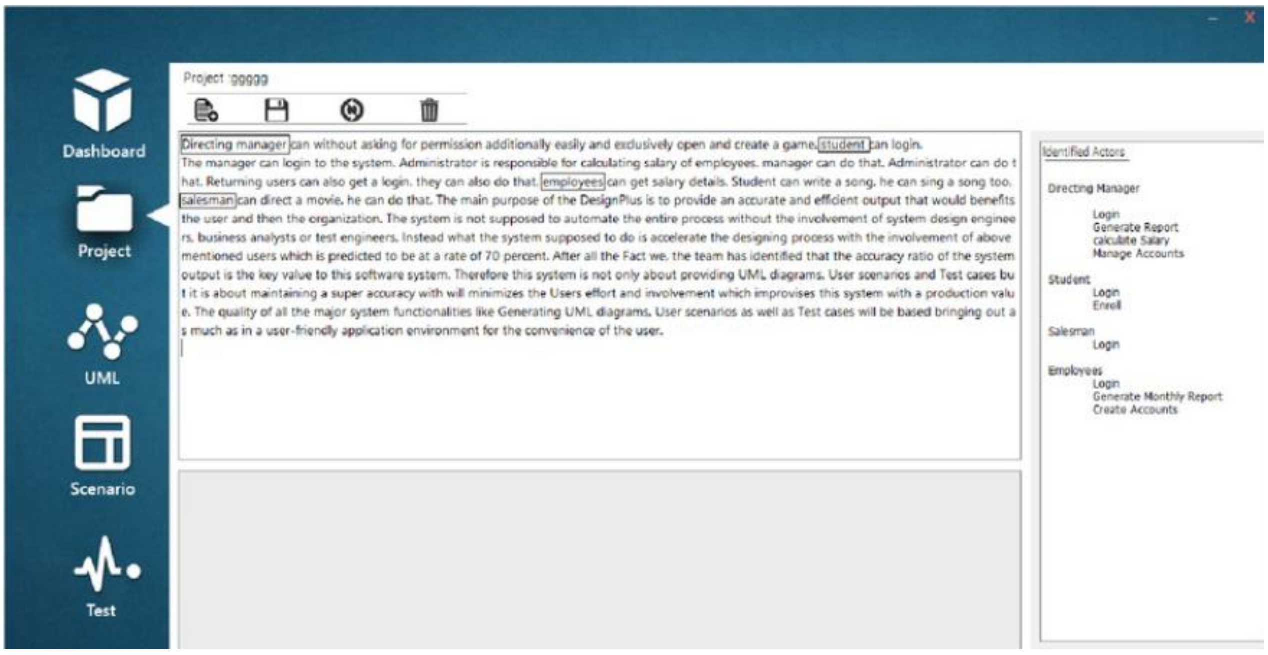
The main purpose of the DesignPlus is to provide an accurate and efficient output that would benefits the user and then the organization. The s ystem is not supposed to automate the entire process without the involvement of system design engineers, business analysts or test engineers. Instead what the system supposed to do is accelerate the designing process with the involvement of above mentioned users which is predicted t o be at a rate of 70 percent. After all the Fact we, the team has identified that the accuracy ratio of the system output is the key value to this sof tware system. Therefore this system is not only about providing UML diagrams, User scenarios and Test cases but it is about maintaining a supe r accuracy with will minimizes the Users effort and involvement which improvises this system with a production value. The quality of all the major system functionalities like Generating UML diagrams, User scenarios as well as Test cases will be based bringing out as much as in a user-friendl y application environment for the convenience of the user

The DesignPlus required a proper Internet connection for [illegible] us Internet access to work with the appli cation. The paid license version the product must verify th [illegible] of use. Then application is validated tim e to time for license expirations and possible pirated licens [illegible] of use, users must subscribe with their e mail address in order to use the Application.

Clear All?

*Figure 2-10 Clear all Prompt*

After user has finished writing, after the last key press of a '.' if not disturbed for few seconds, the System will automatically initiate application of NLPCore functions for the text given. Or the user can manually perform the task by a pressing the generate button. The result will be then displayed in the right compartment of the screen.



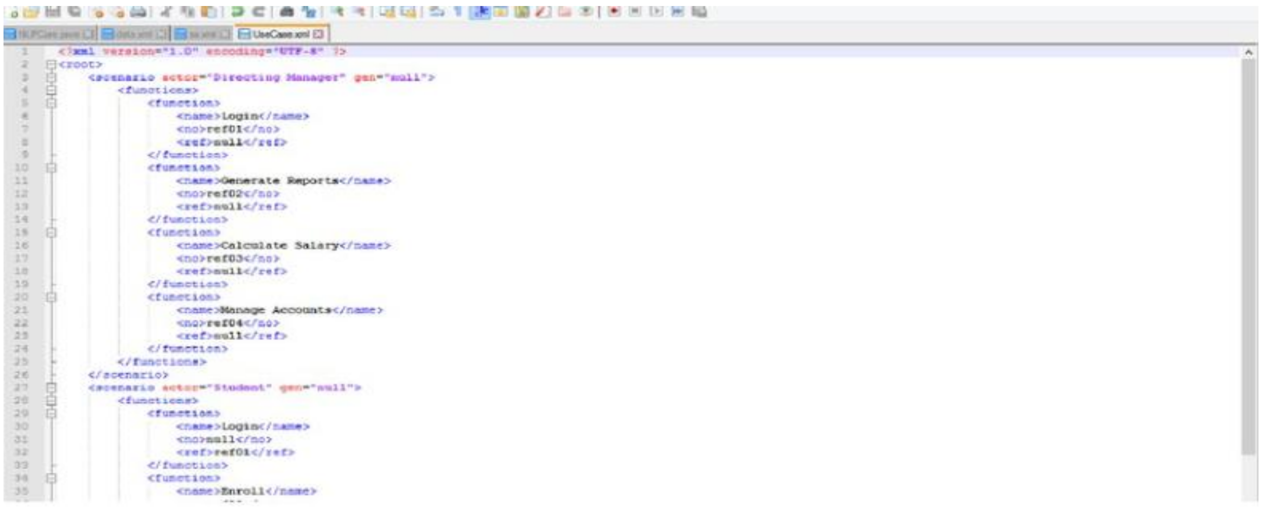
Directing manager can without asking for permission additionally easily and exclusively open and create a game. student can login.
The manager can login to the system. Administrator is responsible for calculating salary of employees. manager can do that. Administrator can do t
hat. Returning users can also get a login. they can also do that. employees can get salary details. Student can write a song. he can sing a song too.
salesman can direct a movie. he can do that. The main purpose of the DesignPlus is to provide an accurate and efficient output that would benefits
the user and then the organization. The system is not supposed to automate the entire process without the involvement of system design enginee
rs. business analysts or test engineers. Instead what the system supposed to do is accelerate the designing process with the involvement of above
mentioned users which is predicted to be at a rate of 70 percent. After all the Fact we. the team has identified that the accuracy ratio of the system
output is the key value to this software system. Therefore this system is not only about providing UML diagrams. User scenarios and Test cases bu
t it is about maintaining a super accuracy with will minimizes the Users effort and involvement which improvises this system with a production valu
e. The quality of all the major system functionalities like Generating UML diagrams. User scenarios as well as Test cases will be based bringing out a
s much as in a user-friendly application environment for the convenience of the user.

Identified Actors

```
Directing Manager
        Login
        Generate Report
        calculate Salary
        Manage Accounts

Student
        Login
        Enroll

Salesman
        Login

Employees
        Login
        Generate Monthly Report
        Create Accounts
```

Figure 2-11 : Results in the Right Corner

The XML file with all the finding will be generated in the project folder. Each actor's & function's relationship will mapped during the process.



```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
    <scenario actor="Directing Manager" gen="null">
        <functions>
            <function>
                <name>Login</name>
                <no>ref01</no>
                <ref>null</ref>
            </function>
            <function>
                <name>Generate Reports</name>
                <no>ref02</no>
                <ref>null</ref>
            </function>
            <function>
                <name>Calculate Salary</name>
                <no>ref03</no>
                <ref>null</ref>
            </function>
            <function>
                <name>Manage Accounts</name>
                <no>ref04</no>
                <ref>null</ref>
            </function>
        </functions>
    </scenario>
    <scenario actor="Student" gen="null">
        <functions>
            <function>
                <name>Login</name>
                <no>null</no>
                <ref>ref01</ref>
            </function>
            <function>
                <name>Enroll</name>
```

Figure 2-12 XML file

Use case generating part is one of major function. It has one main frame under FrameHome class. Main input for the diagram is XML file. Xml file is read using SAX Parser. Then identified all attributes and tags. After that catch all the values that are necessary to draw usecase diagram. Simplify actors and functions get through the Array List to DP_UML class. All the data get in this format.

*Attribute name"+"/"+Attribute value"*

When the attribute name is equal to "Actor" split it from "/" and get Attribute value to array. And also get the number of actors count. According to that decide how many actors need to draw on top of Jpanel. When this attribute name is equal to "Function name" get those values as previous mention. Count the number of functions for one actor and get those functions names to the array up until next "Attribute name=Actor".

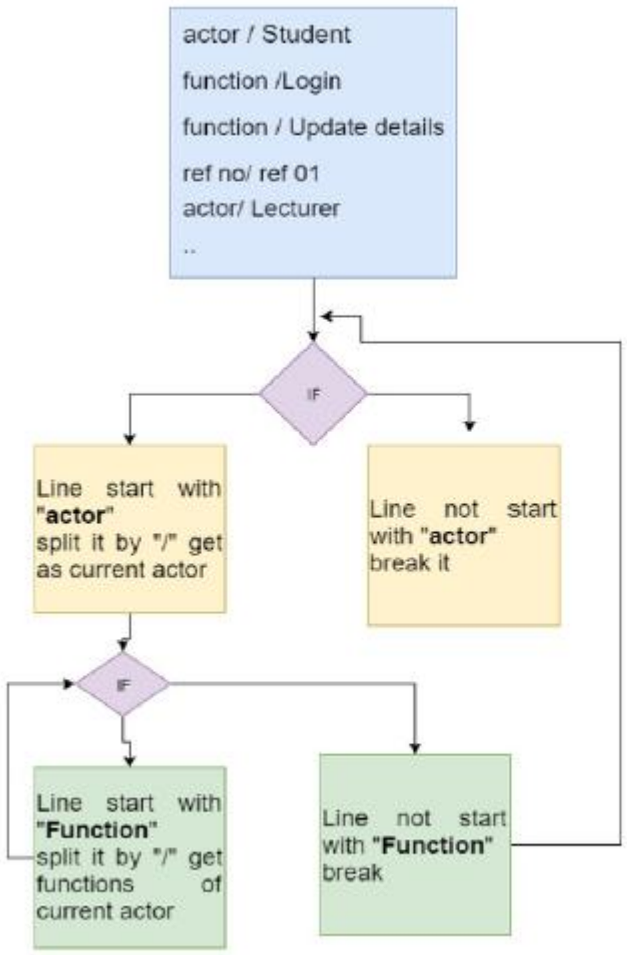

actor / Student
function /Login
function / Update details
ref no/ ref 01
actor/ Lecturer
..
IF
Line start with "actor" split it by "/" get as current actor
Line not start with "actor" break it
IF
Line start with "Function" split it by "/" get functions of current actor
Line not start with "Function" break

*Figure 2-13 Actor Identification*

Next main problem is moving those actors and functions with proper connection in between an actor and functions. To do this rich algorithm is developed. At the very first step all the values are initialized like currentActorX, currentActorY, currentFunctionX, currentFunctionY and windowsHight etc. draw first actor in initialized location and get the number of functions width, height and differences. Using summation of those values algorithm will decide next Y axis position. Once the window size is exist more than Y location of the last function it will automatically swapped the side.

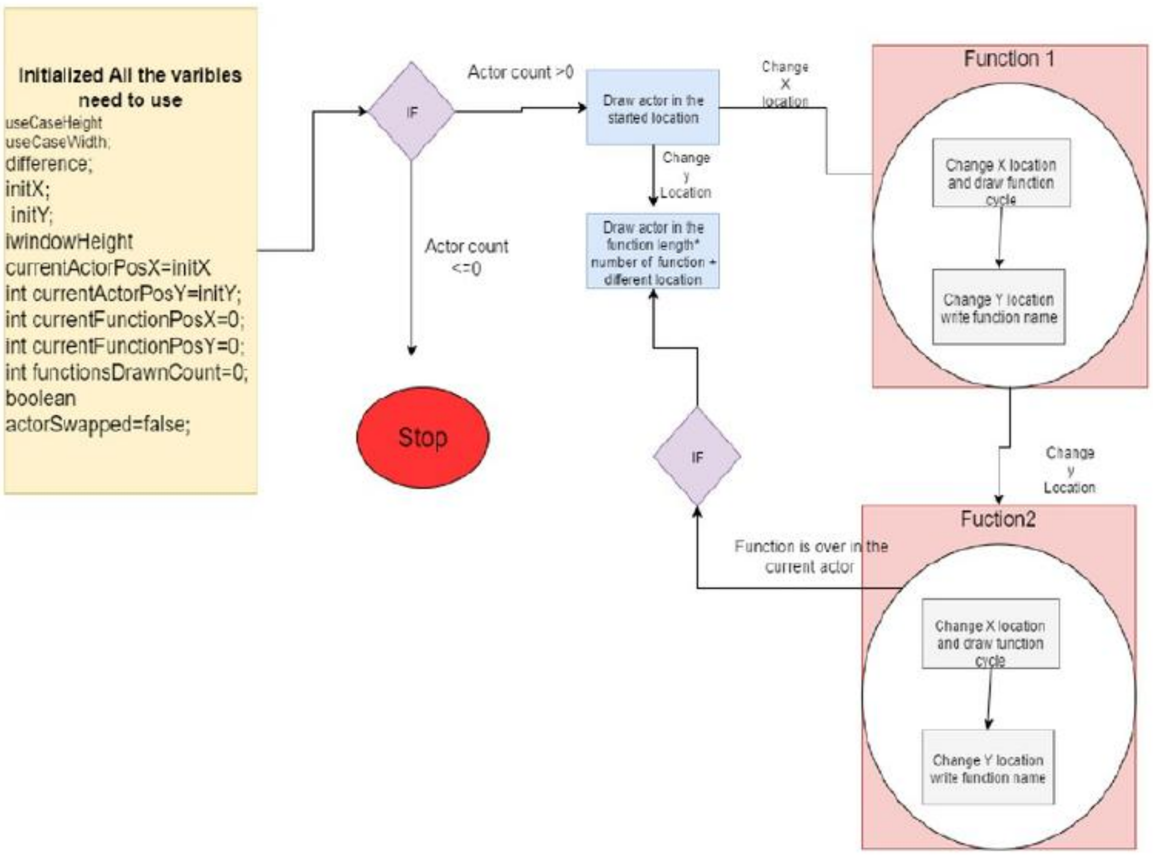

Initialized All the varibles need to use
useCaseHeight
useCaseWidth;
difference;
initX;
initY;
iwindowHeight
currentActorPosX=initX
int currentActorPosY=initY;
int currentFunctionPosX=0;
int currentFunctionPosY=0;
int functionsDrawnCount=0;
boolean
actorSwapped=false;
IF
Actor count >0
Actor count <=0
Stop
Draw actor in the started location
Change y Location
Draw actor in the function length* number of function + diferent location
Change X location
Function 1
Change X location and draw function cycle
Change Y location write function name
Change y Location
Fuction2
Change X location and draw function cycle
Change Y location write function name
IF
Function is over in the current actor

*Figure 2-14 Actors and function drawing algorithm*

Next main component is creating user scenarios according to requirements. Apache OpenNLP 1.5.3 software is used to detect sentences. The Apache OpenNLP library is machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and core reference resolution. These tasks are usually required to build more advanced text processing services. OpenNLP also includes maximum entropy and perceptron based machine learning.

## 2.3 Research Findings

There are lots of new things I had to learn. For the information extraction first I need to identify extract information from which part of the document. For evaluation the NLPCore functionality, I asked few colleagues to use the software and let me know what they think of the software meanwhile I can gather the gather user experience.

The accuracy of the resulting output is mentioned to be very accurate by viewers. There I figure out that the NLPCore needs to have much more grammars included because each new grammar includes into the grammar section also increases the accuracy of the application result. When I increase grammar rules, that covers up the distinguish way of telling of an idea. Since in human language humans can do spell out a single idea in thousand s of ways and there is not limit in that. It's a mere miracle for me to cover up all that possible combination. But what i can do is including a very new grammar and the accuracy will increase gradually & eventually. One other finding is that the OpenNLP lacks some functionality in some ways where in English same one word could stand up as a Noun; and in another place the same word could stand up as verb. But the OpenNLP doesn't always catch the idea correctly. Avoiding this situation is technical unfeasible for the moment, but reducing is always possible.

This is a very challenging part. Then that extracted information should separate sentence by sentence, for that i use some libraries in the NLP. Then that information should store in xml document. Again it is a challenging part of this system. Then using those requirements, system should generate use case diagram, user scenario and test cases generation. Another main problem is identifying relationships in between two or more function when they have reuse functionality or optional functionality.

# 3 CONCLUSIONS AND RECOMMENDATIONS

Content extracted from the given description is a non- trivial through an important task for this application since everything there onwards will be done will based on that data, I have introduced an intelligent algorithm to match the grammar of many different forms. The algorithm is smart enough to notify the user the certainty of the particular piece of findings.

# 4 REFERENCES

[1] Part-Of-Speech tags [Online] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[2] Stuart J. Russell and Peter Norving "Artificial Intelligence A Modern Approach" 1995

[3] "Natural Language Processing ",Wikipedia,16 March 2016, Available:http://en.wikipedia.org/wiki/Natural_language_processig, [Accessed: 2016/02/30]

[4] Debasish Kundu and Debasis Samanta, "A Novel Approach to Generate Test Cases from UMLActivityDiagrams", may–june 2009,Available: http://www.academia.edu/download/31228256/article1.pdf [Accessed: 2018/03/24].

[5] Magda G. Ilieva, Harold Boley, "REPRESENTING TEXTUAL REQUIREMENTS AS GRAPHICAL NATURAL LANGUAGE For UML DIAGRAM GENERATION"

[6] Imran Sarwar Bajwa, M. Abbas Choudhary, "Natural language processing based automated system for UML diagrams generation", 18th National Computer Conference 2006[Accessed: 2018/07/01]

[7] Test Cases from Software Requirements Research Available http://ieeexplore.ieee.org/Xplore/home.jsp [Online]

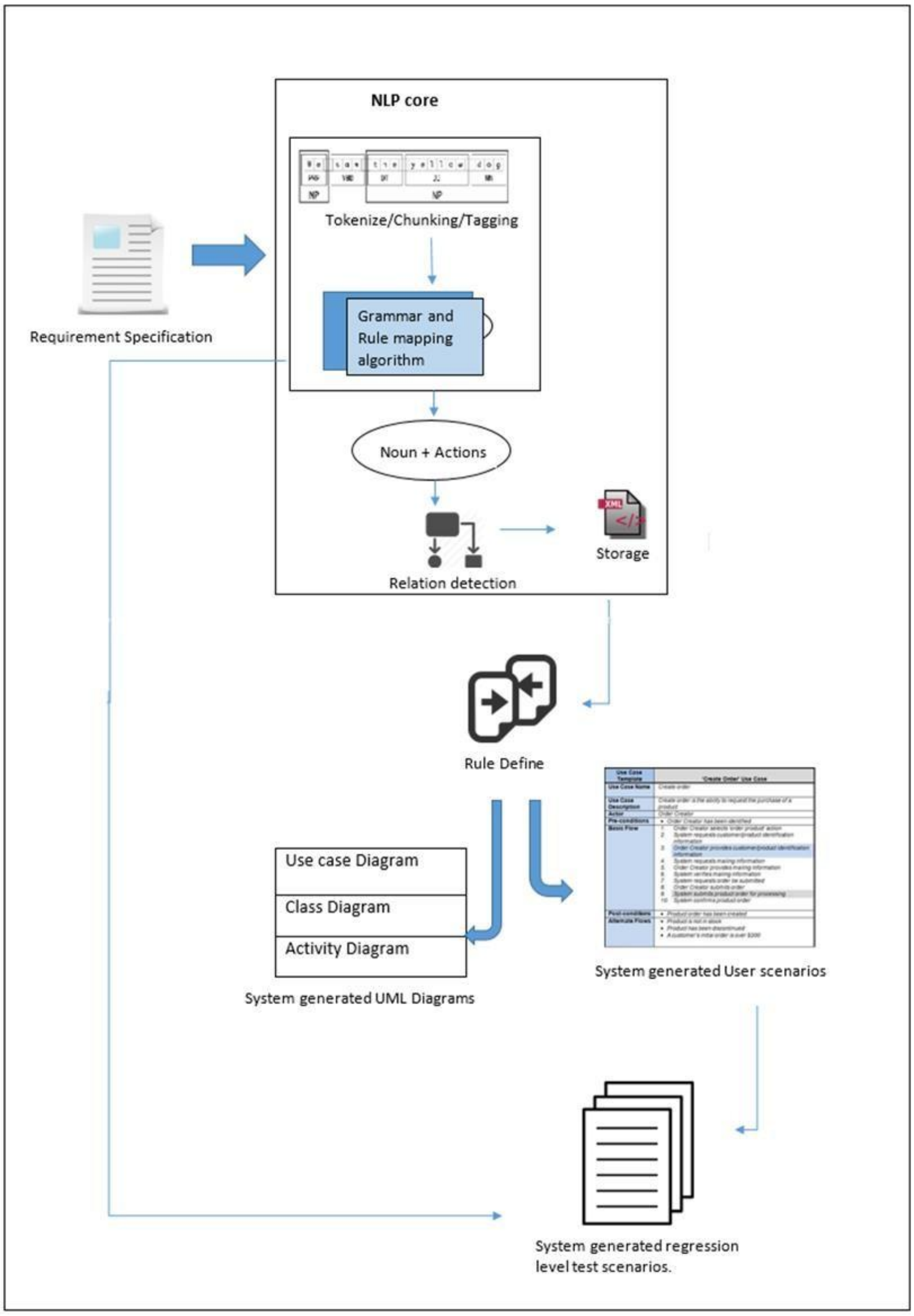
[8] David Nadeau, Satoshi Sekine: A survey of named entity recognition and classification – 2006

[9] Undo & Redo in JAVA [Onine] Available: https://web.archive.org/web/20100114122417/http://exampledepot.com/egs/javax.swing. undo/UndoText.html

[10] XML encryption and decryption [Online] Available: http://stackoverflow.com/questions/8395877/encrypting-and-decrypting-xml

[11] Covington, M., Bates, M. and Weischedel, R. (1995). Challenges in Natural Language Processing. *Language*, 71(2), p.402.

## APPENDIX A: High level system diagram


NLP core
Tokenize/Chunking/Tagging
Grammar and Rule mapping algorithm
Noun + Actions
Relation detection
XML
Storage
Requirement Specification
Rule Define
Use case Diagram
Class Diagram
Activity Diagram
System generated UML Diagrams
System generated User scenarios
System generated regression level test scenarios.

| Use Case Template | 'Create Order' Use Case |
|---|---|
| Use Case Name | Create order |
| Use Case Description | Create order is the ability to request the purchase of a product |
| Actor | Order Creator |
| Pre-conditions | • Order Creator has been identified |
| Basic Flow | 1. Order Creator selects 'order product' action<br>2. System requests customer/product identification information<br>3. Order Creator provides customer/product identification information<br>4. System requests mailing information<br>5. Order Creator provides mailing information<br>6. System verifies mailing information<br>7. System requests order be submitted<br>8. Order Creator submits order<br>9. System submits product order for processing<br>10. System confirms product order |
| Post-conditions | • Product order has been created |
| Alternate Flows | • Product is not in stock<br>• Product has been discontinued<br>• A customer's initial order is over $200 |