****

**Automation of Business Analyst task**

**(Generation of UML diagrams)**

Tony Shereen Shawky -Mai Adel Elsaied- Radwa Ibrahem Abdelrahman- Mariam Nader Neseem- Kirolos Magdy Yassa

Software Engineering department, Faculty of Computer and Information Sciences

Ain Shams University Cairo, Egypt

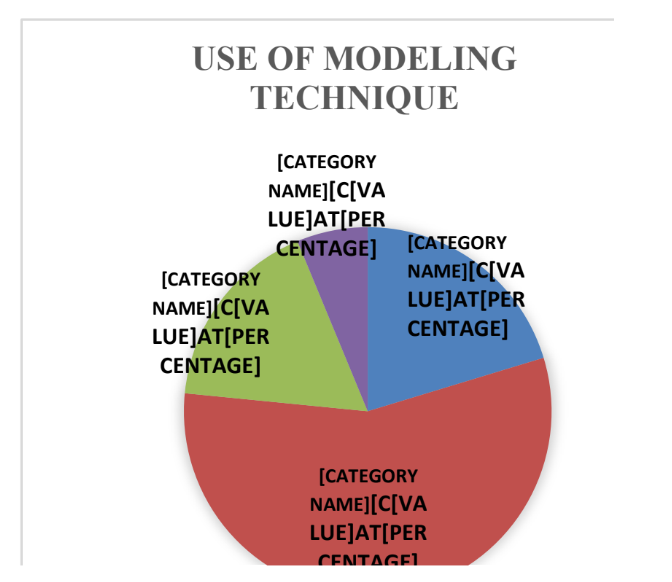
mohamed.mabrouk@cis.asu.edu.eg

*Abstract*— Unified Modeling Language (UML) diagrams are crucial for displaying business requirements and aiding system design from the end user's perspective. However, drawing these diagrams takes time due to the complexity of the business environment and the technical capabilities of UML diagrams. This project thesis aims to develop a software tool for producing UML diagrams from user text files using natural language processing. The system generates use case and class diagrams using NLP techniques, creating a usable, trustworthy web application. The system recognizes important actors, use cases, and relationships for use case diagrams and class diagrams. Keywords— Keywords: UML Diagram, Use case Diagram, Class Diagram, Natural Language Processing, Classification, Plant UML

1. **INTRODUCTION**

The software development life cycle (SDLC) is a crucial process in software engineering, involving 5 core stages: requirements gathering, design, implementation, testing, and maintenance. The design phase, especially, is crucial, as it lays the foundation for the rest of the SDLC. UML Diagrams, such as Use Case and Class diagrams, are generated manually during this phase.

UML is a standardized modeling language used in object-oriented software engineering, combining techniques from data modeling, business modeling, object modeling, and component modeling. This project aims to improve the efficiency and accuracy of the design phase of the SDLC by automating the process of creating UML Diagrams, Use Case, and Class diagrams, replacing manual process. This will speed up requirements specification and accelerate the software design process, ultimately ensuring fast, high-quality software delivery.

 The specific problem that this thesis project addresses is the manual process of creating UML Diagrams from Requirements Specifications, which can be a time-consuming and error-prone process, especially for complex systems. this project aims to solve this problem by automating the process of creating UML Diagrams, especially Use Case and Class diagrams. This will help users to speed up the SDLC process, reduce errors, and improve the accuracy of their diagrams.

This system would be highly beneficial for business analysts, specifically. In Addition, Generally, the system can also be beneficial for anyone developing a Software system, for instance, Product Owner, Student, or Designers, with different job and role titles.

1. **BACKGROUND**
2. *Related Works*

There are many CASE tools for drawing the UML use case and class Diagrams, like Visual paradigm, StarUML and DrawIO are some of the famous software tools currently using in the industry, but all of these software tools are generating UML Diagrams manually and using drag-drop approach. Other Software tools that automating the generation process like:

* SmartDraw Software tool

SmartDraw is a commercial, paid tool for automatically generating UML diagrams, including use case and class diagrams, from text descriptions. The input format varies depending on the version and edition, but typically includes descriptions of actors, use cases, and relationships. For generating use case diagrams, it includes a text descriptions of use cases, and for generating Class diagrams, a descriptions of classes, attributes, methods, and relationships is included.

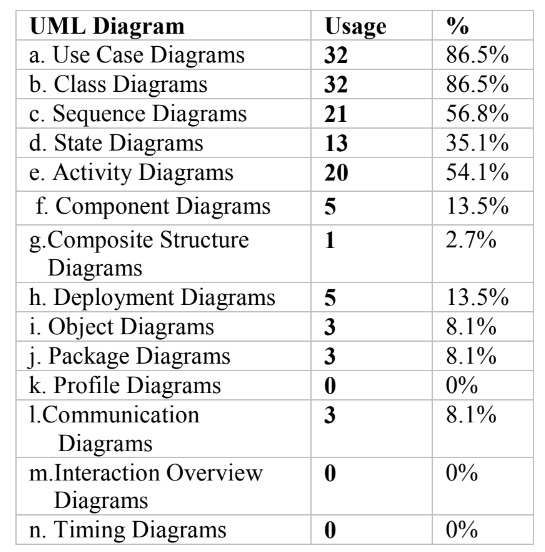
* Rational Rose

## Rational Rose is a commercial, object-oriented software tool for modeling applications using the Unified Modeling Language (UML). It is not open-source and costs vary based on licensing and usage requirements.

1. *Surveys*

Surveys conducted in Sri Lankan, Singaporean, IT industries revealed that some Software designers use both UML and Non-UML modeling for software design. Results (shown in figure 1) showed that the majority (56%) of system designers use UML Diagrams and other modeling techniques for visualizing projects. diagrams for designing.

In addition, results (shown in figure 2) also showed that ,from those who uses UML their systems ,most of them uses Class (86.5%) and Use Diagrams (86.5%) for visualizing their projects.

*Figure 1 Usage of UML Diagrams Based on the Survey*

## Thus, in this thesis project, we have given priority for auto-generating Use Case and Class Diagrams for assissting large fraction of users as much as possible.

III. METHODOLOGY

## Overview

The Objective of this project is to develop an automated UML modeling tool for generating class and use case diagrams during the design phase. The tool uses natural language processing (NLP) techniques to extract relevant information from user input files and generates consistent diagrams, thus improving efficiency and productivity. This developed project is an open-source tool designed to speed up the design phase and save cost and effort. The "Automation of Business Analyst Tasks” enforces consistency and standardization, ensuring accurate and compliant UML models. The tool targets rapid iteration and updates, reflecting system changes, enabling faster adaptation and keeping documentations up-to-date.

## Processing Steps

## In this subsection, we describe the flow of how to use the proposed system from uploading a file of function requirements or user stories to auto-generating use case & class diagrams:

## 1. When registering for an account on the system for the first time the user must first Sign up to the system by filling-out Sign Up form with the required fields.

## 2. Once logged in, user can either add a new project or open an old existing project . If the user chooses to add a new project, he should fill-out a simple interactive form with the required project data , then open the new project.

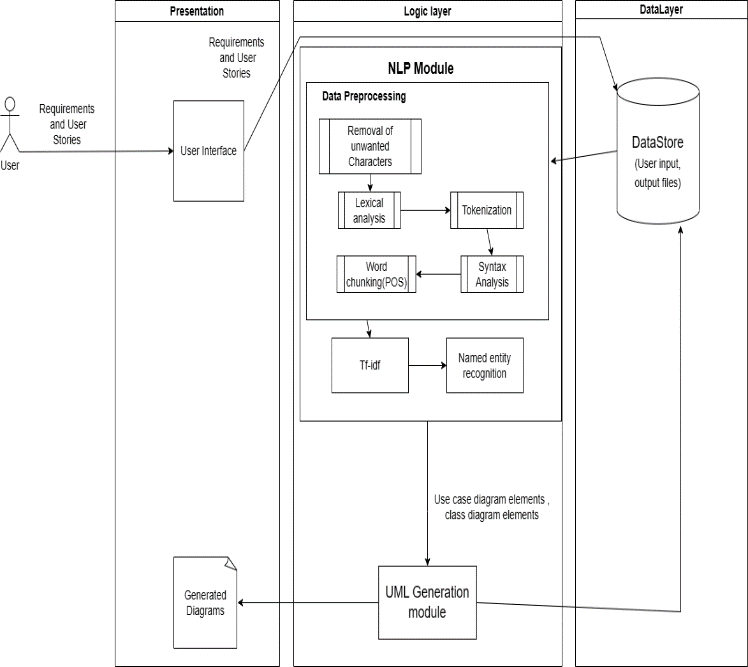
## 3. User upload User Stories and Functional requirement text files while taking into consideration input constraints.

## 4. User chooses which diagram to generate, use case or class diagram.

## 5. System generates diagram and displays the resulted diagram to user.

## 6. The user can also choose to delete file or delete a project.

C. System Architecture



*Figure 3 System Architecture*

## This project follows the Three-tier architecture pattern that is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, the logic tier, and the data tier.

Presentation Tier (frontend): The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect input and information from the user.

Application Tier(backend): The application tier, also known as the logic tier or middle tier, is the heart of the application. In this tier, information collected in the presentation tier is processed. In this layer all data processing and NLP techniques and algorithms are implemented.

Data Tier (Storage): The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed. This is relational database management system NoSQL Database server.

In a three-tier application, all communication goes through the application tier. The presentation tier and the data tier cannot communicate directly with one another but through the logic tier where some preprocessing and data validation is done

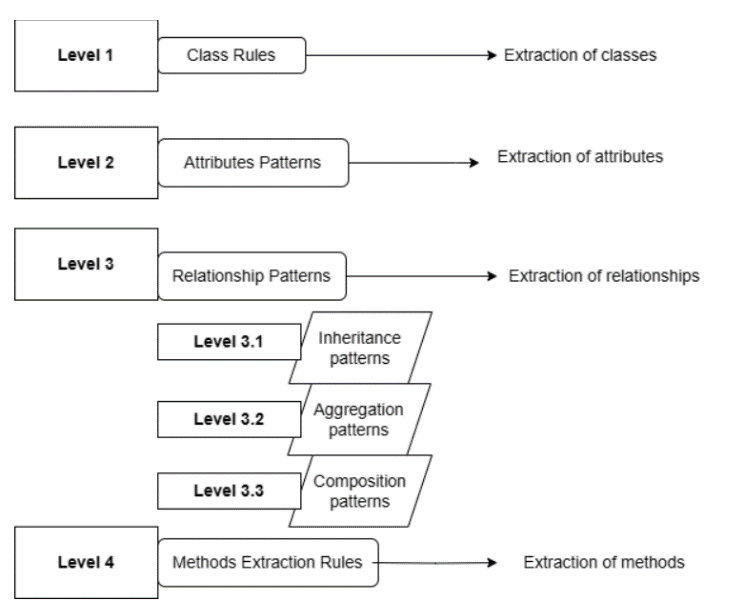
## IV.IMPLEMENTATION

1. *File Preprocessing*: The process involves cleaning user input files, removing unwanted characters, punctuation, numbers, and special symbols. Regular Expressions (“re” module in python) are used for text pattern matching and removal. Stop words are removed to reduce noise and improve efficiency using Spacy library. Spacy library provides predefined stop-word lists. The input text file is identified as sentence boundaries and split into a list of sentences. Tokenization is performed, breaking the sentence into individual words or tokens using the NLP library (SpaCy) in Python.
2. *Generation of Use case diagram:*

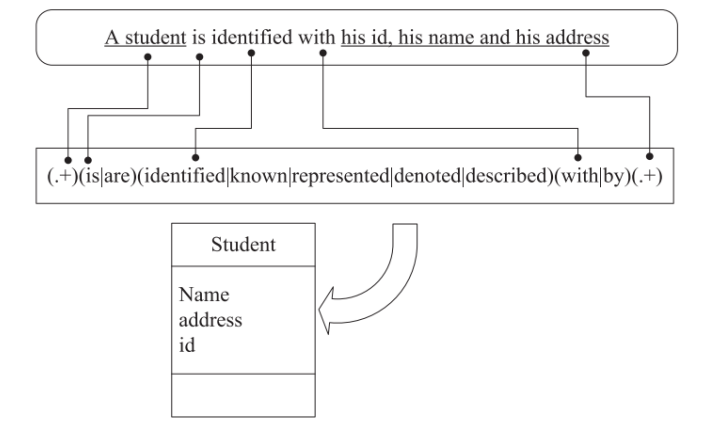
Generation of use case diagram involves 5 main steps:

1. *Actors Extraction :* Actors are extracted from sentences by means of noun chunks of SpaCy library. SpaCy’s noun chunks extraction takes into account grammatical structures and dependencies in the text, allowing it to identify noun phrases accurately.
2. *Use Case Extraction:* The use case is extracted from user stories by using tokens form each sentence, extracting the root verb using POS tags, comparing it to a list of verbs that are not use cases, extracting the object of the root verb using spacy dependency utilities, and extracting object dependents using dependency parsers.
3. *Use Case Extraction Relation Extraction*: The dependency subtree extracts relations between two use cases by comparing tokens of root verbs with words "after, depends on." The subtree compares these tokens with prepositions, extracting what follows as another dependent use case.
4. *Generation of Use case diagram model as text file*: The extracted use case diagram elements are converted into a text file using an implemented use case model class for easy typing.
5. *Generation of Use case diagram image:* After Use case model is written in text file in certain syntax. it passed to PlantUML tool to render it as image file that is displayed to the BA (user) at the end of processing.
6. *Generation of class diagram:*

This module focuses on translating text preprocessing data into useful conceptual modeling data using semantic analysis. A rule-based NLP approach is used to extract class diagram elements, such as classes, attributes, and relations. Pattern matching rules are used to generate structural patterns based on human knowledge competence. These patterns are used to match sentences in the input requirements text, identifying candidate elements of the UML class diagram.[10]



Generation of class diagram involves 6 main steps:

1. *Concepts Extraction:* This module aims to extract concepts from a requirements document using SpaCy parser, stemming algorithm, and WordNet. It involves identifying stop words (saved as {Stopwords\_Found}) , calculating the number of words without stop words, and finding stemming for each word. The module then parses the entire document, extracts Proper Nouns(NN), Noun phases (NP), and verbs (VB) and save results to {Concepts-list}. It then finds semantic relationships between CT and CT2 concepts if {synonyms\_list} contains a synonym (SM) lexically equal to CT. If {synonyms\_list} contains a synonym (SM) lexically equal to CT, CT and CT2 concepts are semantically related. Finally, if {hypernyms\_list} contains a hypernym (HM) lexically equal to CT, CT2 is a kind of CT. The results are saved in a {Generalization-list}.
2. *Classes Extraction:* This module uses various heuristic rules to extract class diagrams, including domain ontology. These rules include Class Identification Rules, which indicate that a concept is a class if it occurs only once in the document and its frequency is less than 2%. Other rules include Classification Rules, which indicate that a concept is general and can be replaced by a specific concept, and Classification Rules, which indicate that a concept is an attribute. If a concept does not satisfy any of the rules, it is most likely a class. Noun phrases with an attribute are considered classes if the second noun is an attribute of that class. The ontology contains information about the concept, such as relationships and attributes. Word Net is used to verify generalization relationships and find semantically similar terms.
3. *.Attributes Extraction:* In this task, extraction of attributes is done . A list of patterns are defined on some key words and expressions with the help of domain experts . Each Sentence in functional requirement file (user input ) after preprocessing is compared to these patterns . The Phrase Matcher of SpaCy allows you to match lists of tokens on a text .

*Figure 5 Illustration Example*

1. *Relationships extraction*: This module involves extraction of Inheritance or generalization, aggregation, and compostion relationships using inheritance, aggregation and composition patterns.
2. *Generation of class diagram model as text file:* After extraction process of class diagram elements , a class diagram model is written in certain syntax in a text file with the help of implemented class model class that assists in typing the class model in a text file.
3. *Rendering UML Diagrams from Plain Text Files:* After Class model is written in text file in certain syntax. it passed to PlantUML tool to render it as image file that is displayed to the BA (user) at the end of processing .Saving Render Resulted images to database .

## V. NLP TECHNIQUES

*Tokenization:*

## Tokenization involves breaking a text into individual units, typically words or sentences. It is a fundamental step in NLP and serves as a basis for further analysis. Tokenization allows for the separation of text into meaningful components, enabling tasks as counting words frequency, analyzing grammar, or creating word embeddings or finding synonyms of specific word. All functional requirements or user stories that are entered by user in form of text files is tokenized (splitted) into sentences then all sentences are tokenized into list of tokens where each sentence contains list of tokens not just words as strings .

*Part-of-Speech Tagging:*

## POS tagging, also known as grammatical tagging , is the process of assigning grammatical tags or labels to words in a sentence, indicating their syntactic category or part of speech. POS tagging is a crucial step in natural language processing tasks and helps in understanding the structure and meaning of text. POS tagging assigns tags to words based on their syntactic roles and categories. Some common POS tags include nouns (NN), verbs (VB), adjectives (JJ), adverbs (RB), pronouns (PRP), prepositions (IN), conjunctions (CC), and determiners (DT). These tags provide information about the word's function and its relationship with other words in a sentence.

## POS is used through the project in extracting root verbs in extracting meaningful use case element. It is also used in pattern matching where The Matcher allows you to define patterns based on linguistic annotations as part-of-speech tags, dependency labels, lemma, and Page | 42 more. For example specifying a pattern that extract an aggregation relation is written as follow :

## Pattern : xClass involve yClass .

## [

## {"POS": "NOUN", "DEP": "nsubj"}, {"POS": "VERB", "DEP": "ROOT", "LOWER": “verbsInvolve”},{"POS": "NOUN"}, {"OP": "\*"}

] [11]

VI. CONCLUSION

##### Design Phase consumes the most amount of time in SDLC, thus developing any assisting tool would highly be beneficial. Thus , this thesis Project develops a Software tool for automation UML Diagrams, use case and class Diagrams, for assisting BAs and designers in Design Phase. In Addition , “Automation of BA Tasks” provides a user-friendly , secure web application for BAs to save , manage , and upload files to the their projects easily. Also, unlike other commercial software tools , “Automation of BA tasks “ is an open- source Website for all users.

##### Our Project makes use of NLP techniques and algorithms in the process of generating use case and class diagram by taking input files from users into the system , user stories for generating use case diagram and functional requirements for generating class diagram , processing them , and providing the output generated Diagrams as a PNG and Txt files for the user.

##### Throughout the Development of “Automation of BA tasks” , our team explored may research papers, meet experts , and reviewed carefully similar Software tools to ensure our project’s value in field of study and to further enhance the consistency and quality of the tool. Our Algorithms and techniques choice to be implemented is merely a starting point to where this thesis could go. “Automation of BA Tasks “ is a promising Research field that we believe could get even bigger resulting in better and more efficient results. The Design and User Interface of the tool were meant to be consistent with BAs to facilitate interaction and user’s experience. The User Interface delivers a simple , easy to navigate and manipulate, and user-friendly Single-page web application.

##### Acknowledgment

##### We express our heartfelt gratitude to ALLAH for granting us the capability to accomplish this task. Then, we would like to extend our sincere appreciation to our supervisors, Dr. Mohamed Mabrouk, for his support, guidance, expertise, and patience throughout this work.

##### References

1. [1] HofferLecture, J. S. (2014). Modern System analysis and Design , Fourth Edition. Pearson
2. Transforming simplified requirement in to a UML use case diagram (n.d.).https://www.researchgate.net/publication/315809182\_Transforming\_Simplifi ed\_Requirement\_in\_to\_a\_UML\_Use\_Case\_Diagram\_Using\_an\_Open\_Source\_To ol
3. Tilakaratna, J.M.P.P., (2016) “Development and Evaluation of an Ontologically Complete and Clear Object-Oriented Conceptual Modelling Grammar”, PhD Thesis, 251-256.
4. Duc Janssens,2019, “Natural Language Processing in Requirements Elicitation and Analysis: A Systematic Literature Review “
5. Prakash M Nadkarni, Lucila Ohno-Machado, Wendy W Chapman, 2011, “Natural Language Processing: an introduction”
6. "UML Distilled: A Brief Guide to the Standard Object Modeling Language" by Martin Fowler .
7. IBM rational rose enterprise 7.0.0.4 IFIX001. (2010, August 1). <https://www.ibm.com/support/pages/ibm-rational-rose-enterprise-7004-ifix001>
8. Open-source tool that uses simple textual descriptions to draw beautiful UML diagrams. (n.d.). PlantUML.com. <https://plantuml.com>
9. [UML]Object Management Group. Unified modeling language specification (2.1.1). Tech-nical report, Frammingam, Mass, 2007. <http://www.uml.org>
10. Naik., A. a. ( (2016).). “Reflecting‎Natural‎Language‎Text‎in‎to‎ UML‎Diagrams.”.
11. Universal POS tags. (2014-2022). Retrieved from Universaldependencies: <https://universaldependencies.org/u/pos>

**IEEE**

**conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being publishe**