NATURAL LANGUAGE PROCESSING FOR AUTOMATED SYSML DIAGRAM GENERATION

A Thesis

by

JOSHUA ANDRE ONTIVEROS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE IN ENGINEERING

Major Subject: Mechanical Engineering

The University of Texas Rio Grande Valley

August 2024

NATURAL LANGUAGE PROCESSING FOR AUTOMATED SYSML DIAGRAM GENERATION

A Thesis
by
JOSHUA ANDRE ONTIVEROS

COMMITTEE MEMBERS

Dr. Satya Aditya Akundi
Co-Chair of Committee

Dr. Constantine Tarawneh
Co-Chair of Committee

Dr. Hiram Moya
Committee Member

Dr. Horacio Vasquez
Committee Member

August 2024

ABSTRACT

Ontiveros, Joshua A., Natural Language Processing for Automated SysML Diagram Generation. Master of Science in Engineering (MSE), May, 2024, 97 pp., 15 tables, 19 figures, references, 14 titles.

This thesis explores the applications of natural language processing (NLP) techniques in model-based system engineering (MBSE) to help generate System Modeling Language (SysML) diagrams. MBSE is a method that aids in enhancing traditional engineering practices by modeling to help improve understanding and communication in systems development. SysML, one of the modeling languages for MBSE, helps represent a system's architecture, behavior, and information flow. Translating systems requirements and specifications into SysML models can be time-consuming and can lead to errors when created manually. Automating the creation of SysML diagrams from textual descriptions with the help of NLP techniques can aid in faster realization and reduce modeling errors. This will simplify the initial stages of system design, ensuring precision and uniformity in models. Despite the promise, generating SysML diagrams using NLP faces challenges against natural language's inherent complexity and the need for significant domain knowledge, leading to challenges in extracting and interpreting system requirements from natural language text. This thesis reviews three commonly used NLP-based approaches to generate system representation from natural language text, i.e., Rule-based, Machine Learning (ML)-based, and Hybrid methods. The rule-based method relies on predefined rules to map text to SysML elements, the Machine Learning method learns from data to identify relationships and patterns, and the Hybrid method aims to combine the strengths of both.

Further, a Rule-based framework is proposed to partially automate the generation of SysML diagrams to address some of the challenges identified. The framework demonstrates its effectiveness in creating a SysML representation by implementing an application for generating Class diagrams. The proposed framework underlines the potential issues related to natural language variability and complexity, paving the way for a more streamlined generation of system architectural representations.

DEDICATION

This thesis is dedicated to my family who have been encouraging me throughout my entire educational process. To my mother and father Deborah L Ontiveros, and Cesar Ontiveros, I am forever thankful for the love and enlightening my beliefs that I would be able to accomplish my academic goal. My brothers Cesar A Ontiveros, Luis A Ontiveros, and Carlos A Ontiveros for being amazing role models and inspiring me to become a better individual. To my beautiful wife Silvia C Ontiveros for the motivation and always supporting me through every trial.

ACKNOWLEDGMENTS

I am grateful for the financial support I received for this research and the support from the University of Texas Rio Grande Valley Mechanical engineering faculty. The National Science Foundation (NSF) Award No. 2235999 and CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS) under NSF Award No. 2112650 supported and funded this research at the University of Texas Rio Grande Valley.

I want to express my gratitude to Dr. Satya Aditya Akundi for introducing me to this field of research and opening my perspective to this area of study. You have been the best of help with your support and guidance throughout my master's degree. Your mentorship has significantly shaped my educational and professional journey, and I am deeply grateful. Thank you for believing in me and inspiring me to strive for excellence, I could not have done it without you.

I am grateful to Dr. Constantine Tarawneh for giving me the opportunity to conduct this research for the CREST program. Your support and availability during my journey have been invaluable and this chapter of my career would not have been possible without your assistance. You have inspired me to become a better individual during my academic excellence and personal development. Thank you for being a great mentor and making sure I excel throughout my career and personal life.

I want to thank my friend Timothy Lyons for encouraging me during my research and master's program. The obstacles we've faced have shown me that nothing is impossible, and I am forever grateful for your friendship and inspiration.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

1.1 Natural Language Processing

Natural language processing (NLP) is a helpful technology in machine learning and for software requirements specifications. It is a branch of Artificial Intelligence (AI) that helps bridge the gap between human natural language and machines so that computers can understand, generate, and interpret human language in a meaningful and useful way [1]. NLP assists in extracting and analyzing text data gathered from documents to help specify assembling, classifying, and recording requirements. Although NLP has many advantages in machine learning and software requirement specification, it can be highly ambiguous and complex when analyzing and extracting requirements from a text. Many tools help reduce errors and issues when extracting information from text documents. Using NLP techniques, one can develop algorithms for machines to process and understand human language that benefit information retrieval, translation, data analysis, and extraction. NLP accomplished many advances in word segmentation, parts-of-speech tagging, and syntactic analysis [2]. NLP can analyze large amounts of textural data, making the process faster and more efficient [3]. This brings us to the importance of information extraction (IE), which can be crucial in helping identify phrases and interesting textual data that can help summarize and find entities necessary for many applications [4]. IE is a powerful method that uses various NLP techniques in extracting useful information from source documents for classification, identification, building databases, and many more that

can be applied to various fields [4]. Integrating automatic identification and IE of vital information into domain-specific areas alters the search experience from an extensive process to an efficient and accurate direct search, which is benefitted with the help of NLP [4].

1.2 Model Based System Engineering

Model-based System engineering (MBSE) is a methodology that uses models to illustrate and support the entire lifecycle of a system [5]. Unlike traditional engineering methods, which rely on lengthy requirement specifications and having to design and analyze the process manually, MBSE designs systems using digital modeling and simulations, which showcases an interactive way to represent a system and how it behaves [5]. These digital models explain the different structures within a system, the behavior each component is experiencing, and how they are interconnected with other parts of the system. MBSE automates the system design process by making understanding and updating information within a complex system easier. It centralizes information, ensuring it is accessible and visible to the stakeholders who need it [6]. The Advantages of using MBSE approaches to design systems are that it adds clarity in design, which improves decision-making and collaboration between engineering teams, it helps comprehend a greater complexity of systems that use resources from other subsystems to show how they are interrelated, and it identifies risks and reduces costs and time before implementing a real-world test, enabling a way to track progress [5].

Various industries and sectors benefit from using MBSE; these industries, such as aerospace, automotive, defense systems, and beyond, create complicated products that are demonstrated through modeling to help support how a specific system is designed and operates. These products include satellite, spacecrafts, weapon systems, surveillance, communication networks, and many more [5][6]. MBSE helps manage these products, which can contain

multiple components that add complexity to the system; this is useful in preventing failures and helps get products delivered in confidence.

1.3 Modeling Languages

Unified Modeling Language (UML) and System Modeling Language (SysML) are visual modeling languages used as tools in MBSE to design, specify, and document system artifacts. Both UML and SysML share the same graphical modeling language. UML is widely used to create models of software systems. SysML, an extension of UML, is usually used for systems engineering applications to support specification, requirements, analysis, design verification, and validation of systems and system-of-systems [7]. SysML reuses a subset of UML diagrams and heavily modifies others. Some modified system diagrams captured are the block definition diagram, like the traditional UML class diagrams, and the internal block diagram, used to describe a block's internals [8]. Two new diagrams are also considered in SysML, a requirement diagram and a parametric diagram, which are variations of UML diagrams [8]. Figure 1 illustrates the diagrams from UML and SysML and their intersection.

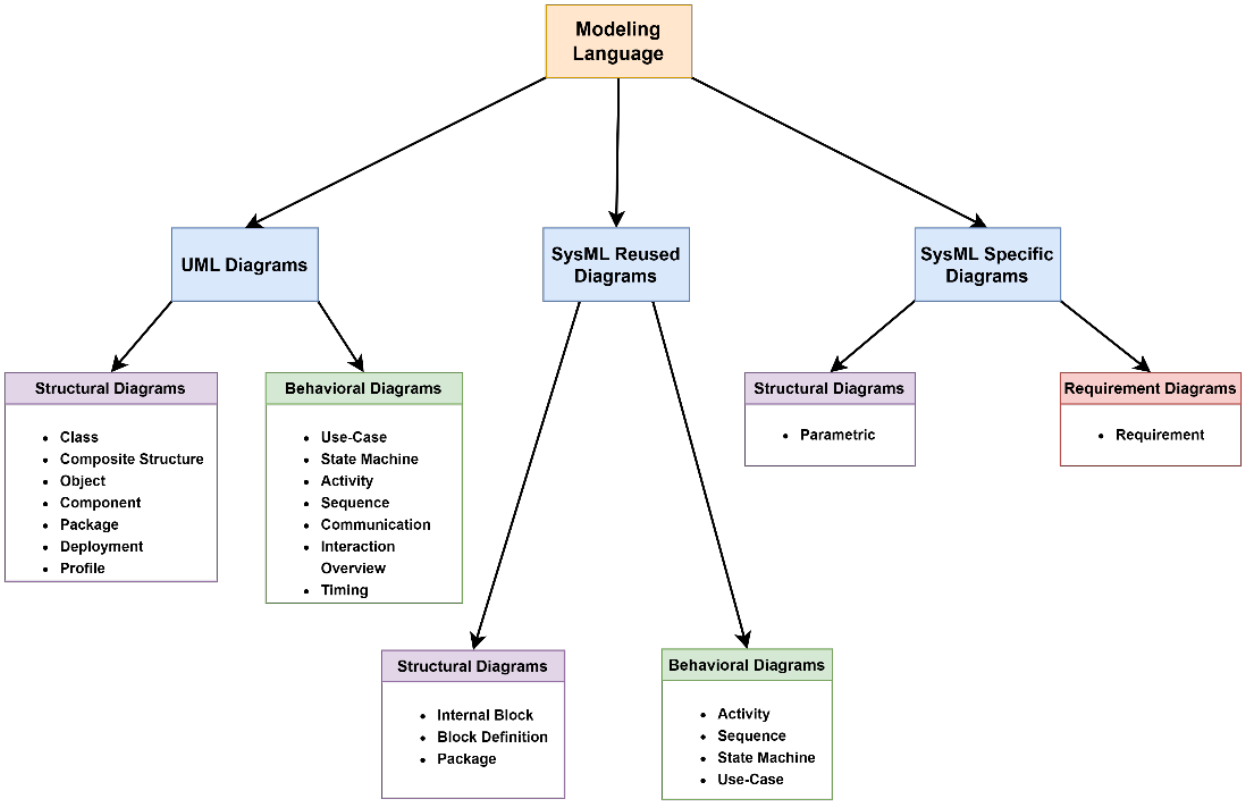

Figure 1. Different types of UML and SysML diagrams [9]

SysML includes diagrams that can specify a system's requirement, behavior, structure, and parametric relationships [8]. These diagrams illustrate how a system is structured, behaves, and interacts with other parts, showcasing the rules it must follow. Behavior diagrams include use-case diagrams that help depict the operation of a system, activity diagram that shows the stages and flow of a system control according to process, sequence diagrams that illustrate different elements and the interactions between them over time, and a state machine diagram that models and tracks an object in response to events [10]. Structural diagrams include block definition diagrams, which contain subsystems and elements illustrating the system's hierarchy; internal block diagrams, which detail components and their connections, illustrating the internal organization of a system; and package diagrams, which organize and manage a system by grouping related components in a package [10]. Parametric diagrams focus on specifying system

constraints, integrating its performance, reliability, and physical characteristics with engineering analysis to ensure the system aligns with design goals.

In contrast, the requirement diagrams capture and facilitate traceability through development by documenting and organizing the system's design [10]. These diagrams help facilitate a good understanding of how a system is designed and works, displaying the essentials of all the components in a system's design. They help highlight what works together in systems engineering to help turn ideas into real-life solutions. Figure 2 illustrates the different diagrams in SysML.

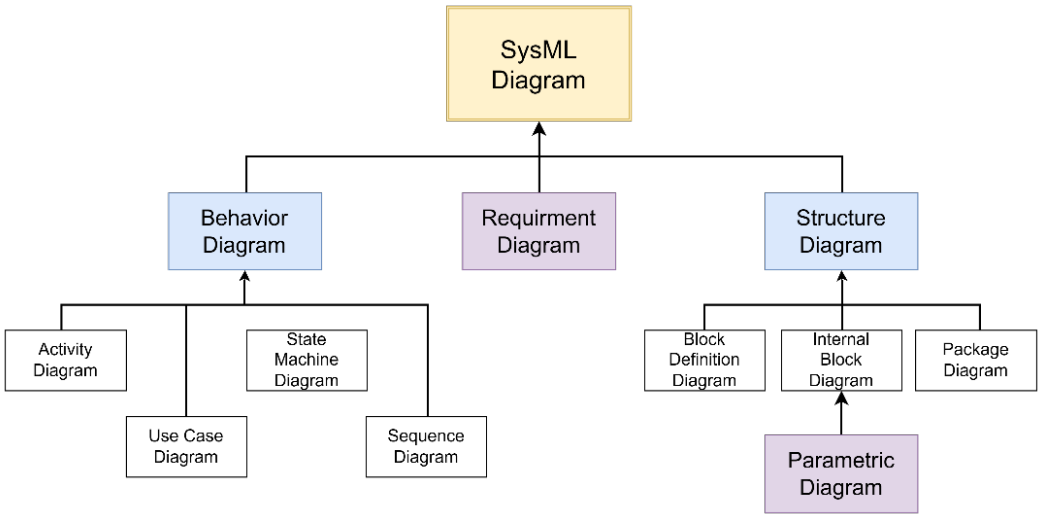

Figure 2. Representation of Different SysML Diagrams

1.4 Integrating NLP with MBSE

Integrating Natural Language Processing (NLP) with SysML enhances engineering systems' design experience. Only some attempts have been observed to include aspects of NLP for requirements elicitation, tracing, and classification, with a substantial need to preprocess and structure input data [11]. The emerging field of research at this intersection is how NLP can be

used to automatically generate SysML diagrams from textual descriptions such as requirement documentation. This process involves using NLP techniques to analyze and interpret text descriptions of a system and then using that information to generate a SysML diagram automatically. Automatic generation of SysML diagrams from textual input such as requirements documents has the potential to mitigate common errors in the system design process, reduce errors when compared to the manual creation of system architectures, and, in addition, make it easier to consider non-technical stakeholders to contribute to the system design process. The methodology includes extracting entities and their relationships using NLP and further mapping them to SysML elements to be refined by engineers [12].

The intersection of NLP and SysML is a groundbreaking advancement in visual architectures, aiming to automate translating textual descriptions into detailed SysML diagrams. The integration of NLP and SysML can be done by extracting a collection of critical phrases and relationships related to the phrases to generate an organized list corresponding to a SysML profile [12]. The automation of creating a visual diagram leverages rule-based and machine-learning algorithms, which are employed to interpret structural connections and hierarchy and utilize drawing algorithms to render the data into visual diagrams according to SysML standards. One of the primary challenges for these algorithms is gathering enough diverse data for training, which is vital for the model's ability to accurately recognize and classify entities for the autonomous generation of SysML elements. Such advancements highlight the promising integration of AI within the MBSE process [13]. Integrating NLP and SysML diagrams not only aids in restructuring system modeling diagrams but also enhances the monitoring of all systems. SysML diagrams are an essential tool, given their role in clarifying an operational understanding of a system. With the help of NLP, they can become powerful in assisting in translating

requirements into understandable models. This underscores the importance of SysML in advanced systems engineering and highlights its importance for integration.

This thesis explores three approaches to implementing NLP in creating system architectures: Rule-based, Machine Learning-based, and Hybrid [14]. A rule-based approach implies a set of predefined rules that analyze and transform information from text-based requirements to create system architectures. The machine learning approach involves training a model on large datasets of natural language text using algorithms to recognize elements and generate a structured text representation to diagram components [15][16]. A hybrid approach combines rule-based and machine learning techniques; it can use a rule-based approach to identify relationships and entities from NL text and use a machine learning model to generate a diagram. Both approaches are used separately or combined to help achieve better performance in creating diagrams more efficiently.

This thesis extensively examines the relevant methods and frameworks within the research area. Chapter II offers a review of the three approaches. Chapter III discusses the challenges and roadblocks observed. Chapter IV examines the methodology used to create system diagrams using NLP. Chapter V showcases a case study example to illustrate how the system operates, leading to the conclusion.

CHAPTER II

LITERATURE REVIEW

2.1 Rule-based Approach for NLP-Driven SysML Diagrams

A rule-based approach uses a set of heuristic rules that perform the transformation process of generating system diagrams. This approach is supported by NLP techniques to aid in processing, interpreting sentences, and extracting elements from given requirement specifications to generate structured architectural diagrams. This method helps write and normalize requirement documents to avoid misunderstanding and leverages automation and human reasoning for enhanced diagram accuracy [17].

Rule-based Natural Language processing techniques have been used to extract elements of a use case diagram from textual description. The process begins with spell-checking, which is crucial for ensuring accuracy for the next steps in the analysis. Next, segmentation is used to help manage data, which divides the text into sentences to help simplify the analysis, followed by tokenization to further break down sentences into individual words known as tokens, which helps the parts-of-speech (POS) process in the following stage. The POS tagging assigns grammatical roles to these tokens, such as verbs, nouns, adjectives, etc., which is crucial for understanding the meaning of the text. These steps are enhanced by chunking, which groups a sequence of words under a single tag to create meaningful clusters based on grammatical patterns. Rule-based techniques are then applied to identify grammar roles from the part-of-speech tags for extracting elements of the use case diagram. This approach has been successful in identifying the elements

of a use case diagram, such as actors (identifying subjects and pronouns of a text), relationships, and use cases (identifying verbs in text) [18]. A similar approach is using a text-to-model framework to improve productivity while creating SysML models. The process includes techniques such as Pre-NLP text cleaning, structural analysis, and Named-Entity Recognition (NER) to identify actors and their responsible actions for a set of machine-readable natural language-based policy documents. NER refers to labeling words of a text with their grammatical category, such as nouns, verbs, etc. This aids in ensuring all the sets of actions required are first captured using NLP and then translated and compiled into a proper SysML model [19]. Another approach to assist in developing system models is to use a set of heuristic rules based on the frequency of unique verbs and actions in a text to help in identifying objects, attributes, relationships, and actors for developing a use case diagram [20,17]. Chen and Zheng use the semantic representation of text through an intermediate graphic language called the recursive object model (ROMA) to generate the use case and class diagrams [21]. This approach depends on the capability of the intermediate ROMA system used, which is majorly used to capture the semantics of the natural language used.

Meziane et al. developed a set of rules for attributes, class, and relationship naming conventions using 45 class diagrams taken from academic textbooks for a syntactic analysis based on the frequency of how often the textbook uses a specific rule in identifying classes and relationships. The association identified in most cases is reported to be composed of a single verb in the third person singular or a verb followed by a preposition. Rules developed were aimed to understand and disambiguate the names given to classes, relationships, attributes, and operations in a UML class diagram [22]. Arumugam and Uma use a similar rule-based approach where a given text input is split into sentences for tagging and marking parts of speech of each word. The

text input is simplified into constructs by using a normalizer for ease of mapping words to object-oriented constituents [23]. Biase et al. propose a high-level model through a semi-automatic approach containing rules that improve the initial models by creating transitions and annotating them with triggers, conditions, and actions. This enables the generation of fragments of SysML state machine diagrams from text requirements [24]. For illustrative purposes, a rule-based technique identified by Salih and Sahraoui for generating class diagrams [25] is provided in Table 1. The table represents a set of rules, each separated by a semicolon that helps identify components of a class diagram from a natural language text document.

Table 1. A Rule-Based Technique to Generate Class Diagrams

Components of a Class Diagram	**Classification Rules**
Classes	NN + NNP + VBP; NN + NNP; NNP + NN + VBZ; NN + VBZ, base form + NN; NNP + NNS, NNP + NN, NNS, NNP + NNS; NNP, NNP + VBZ past tense
Methods	NN + NN + NNP; NNP + NNP + NN + NN; non-3rd person singular present VBP + NN; 3rd person singular present VBZ + NN + NN; non-3rd person singular present VBP + CC + NN + any words; IN + JJ + NN
Attributes	JJ + NN
Relationships	VBP + NN; VBP + NNP; VBP + VBG

Note: Singular Noun (NN), Plural Noun (NNP), Verb (VBP), Plural Noun (NNS), Verb non 3rd person (VBP), Cardinal Number (CC), Preposition (IN), Adjective (JJ), Verb Present Participle (VBP); Verb 3rd person singular present (VBZ); Verb present participle (VBG).

2.2 Machine Learning-based Approach for NLP-Driven SysML Diagrams

The machine learning approach trains a model to study relationships and patterns between NL text and architectural elements. It automates the process of translating NL text of requirements that help generate SysML diagrams, which can sometimes be a tedious task to do

manually. Using machine learning helps reduce time and errors when creating diagrams. Machine learning algorithms can study an extent of meaningful understanding in categorizing and identifying patterns to generate diagrams, depending on the data analyzed. Limited articles exist on machine learning in developing automated system architectures using NLP.

Narawita and Vidanage developed a web-based UML generator that extracts use cases, identified actors, and attributes using a combination of NLP preprocessing techniques and Extensible Markup Language (XML) rules to generate class diagrams. Once identified, a Weka module helps rate the use cases and extract associations [16]. The Weka Module is a popular Java-based machine-learning library that provides users access to visualization tools and algorithms for data analysis and modeling [26]. Kochbati et al. propose a machine-learning model in which pre-processed text is transformed into numerical vectors to compute semantic similarity among the words in a text input and identify clusters to generate use case models [27].

Chami et al. propose a text-to-model framework that labels the uploaded raw text data to identify the actors, use cases, blocks, and associations. Approximately 100 sentences were labeled and fed into an open-source library for advanced NLP to train and customize a machine-learning model. This framework is applied to identify actors textually, use cases, and associations from a text input [13]. Qie et al. propose a deep learning technique that first extracts semantic relationships from input texts to identify relationships such as composition, aggregation, and generalization for developing a block definition diagram. The semantic analysis involves entity recognition and entity relation extraction that gathers domain-specific words and uses word embedding to implement a deep convolutional neural network (CNN). Once the relationships are identified, an Rhapsody API dynamically creates models [2]. The proposed

techniques highlight the potential of ML models to streamline and aid the automatic translation of natural language text to system models - but primarily the Use case diagram.

2.3 Hybrid Approach for NLP-Driven SysML Diagrams

The hybrid approach combines rule-based and machine-learning approaches to help generate UML diagrams from textual requirements and model training. It can be used in parts to process textual data and locate patterns to create diagrams, combining the strengths of manual and automated approaches. Generating diagrams with this approach can benefit the overall design quality; it can accommodate using both automatic and manual methods to resolve complex solutions that are impracticable with one. Hybrid uses NL processing techniques such as tokenizing, POS tagging, sentence splitting, word chunking, and other tasks to extract text information, eventually identifying components such as elements, actors, and relationships to make up SysML diagrams. Machine learning algorithms are trained to identify relationships between the processing data and categorize applicable components of SysML diagrams. The combination of both rule-based and machine-learning approaches allows for a more robust and accurate system.

Using machine learning and NLP techniques, Narawita and Vidanage save time and increase precision in generating use case and class diagrams. A text input of requirements is first analyzed using an NLP module for tokenizing and POS tagging to classify potential actors and classes using POS tag value nouns. An XML rule removes unwanted words from a list of nouns, followed by word chunking to find verbs, nouns to represent actions, and two consecutive nouns where the second noun is a number to define attributes in a use case diagram. A Weka machine learning model evaluates and rates the actions and relationships to determine their validity [16]. Riesener and Dölle propose a structured Model-Based System Engineering (MBSE) requirement

table for processing unstructured text using the spaCy NLP module. The text is tokenized, and POS is tagged to group noun phrases to identify subjects and objects within a sentence by co-referencing pronouns to nouns. NER identifies entities added to the requirement table for training machine learning models to detect requirement properties based on context [28]. Zhong et al. propose removing relationships and critical phrases from a raw text input to identify blocks and relationships based on documents such as specifications, manuals, technical reports, and maintenance reports to generate SysML diagrams, specifically structure and requirement diagrams. Steps involve manual selection of corpus text documents, extracting key nouns, extracting relationships, generating a list of phrases and relations, generating SysML model elements, and manual iterative selecting of the profiles and blocks to be plotted [29].

CHAPTER III

CHALLENGES AND ROADBLOCKS OBSERVED

3.1 Challenges of Implementing a Rule-based Approach

Natural language is inherently complex, and developing a set of rules to aid in the automation of generating system architecture would require rules that address language variability among different actors of a team while generating text-based requirements. This demands the use of contexts and synonyms when considering the development of a set of rules. However, from a domain-specific outlook, as one develops a set of rules, the number of rules to be created can increase exponentially, and maintaining consistency and scale will become increasingly difficult. It is also identified that rules, once defined, are rigid, meaning with changes in technology and terms used across domains, a standard set of terms and keywords are not applicable across domains, and the change to new terms and what they mean in a sentence structure needs to be continually updated with significant manual input. Another challenge could be the propagation of errors if the parts of speech that define a rule must be tagged appropriately, leading to incorrect set assumptions on constituent SyML diagram elements from a natural language text.

3.2 Challenges of Implementing Machine Learning-based Approach

The complexity and variability of natural language make it difficult to accurately extract the information required for general system architectural diagrams from raw text using machine learning models. Further, ensuring a machine learning model always accurately uses the correct notations to generate SysML diagrams can be challenging. To mitigate this, machine learning

models require large amounts of data to learn from. Considering the nature of SysML diagrams and their context-specific models, collecting data to train the models could prove difficult.

3.3 Challenges of Implementing Hybrid Approach

A common challenge involves developers' need to manually integrate heuristic rules into machine learning models to ensure the rules defined are finely tuned and precise. In addition, generating rules can take time, making it challenging to update from implementing changes. While machine learning models can analyze substantial amounts of data, one issue is that these methods demand extensive datasets to tune the model for optimal outcomes to ensure the output is interpretable in the context of SysML. Training machine learning models with large datasets to account for variability in natural language is a challenge, and the key is to ensure a cohesive set of data, such as manually annotated diagrams, is used for training.

Table 2 further dissects these challenges, identifying and categorizing them according to the type of system architectural diagram.

Table 2. Challenges Identified for Generate Individual SysML Diagrams Using NLP.

System Architectural Diagram Type	Associated Challenges
Use Case Diagrams	• Lack of tools for grouping English language text into different bits of meaningful information [18], • Requirements specification issues due to continually emerging technical jargon and frequently observed inconsistencies in large requirements textual documents [30], • Lack of NLP plugins that support complex part-of-speech relationships, such as compound nouns between use cases [31], • Lack of standardized format in which domain requirements are specified [21,16, 32] and • Lack of the ability for the user to modify a diagram once automatically generated as an image [21, 16].
Activity and Sequence Diagrams	• The ambiguity of natural language could generate different versions of the activity diagram based on how a model interprets the natural language text [22]. • The difficulty of NLP tools to understand Variable terminology across domains that may not accurately represent the required transition and the order of actions among different activities [20]
Class Diagram	• Ambiguity and imprecision of natural language may lead to not extracting the relevant information and relationships between concepts [33] • Tools such as CM-Builder can generate UML class diagrams. However, they cannot identify operations for candidate classes, and despite being proficient in analyzing NL requirements, their efficiency in generating UML models from analyzed requirements is questioned [33]. • Using grammatical knowledge patterns by machine learning algorithms while training could lead to assumptions like core classes are always connected, incomplete diagrams, and the inability to identify the multiplicity of relationships [20,17].
Package Diagram	• Poor writing style and document structure, such as implicit headings, can lead to difficulty automatically creating a package diagram [19]. An example includes the accuracy of text-to-model results, which was challenging to evaluate, as differences in writing style between documents could affect the accuracy [19]
Block Definition Diagram	• Lexical-level features cannot capture compound semantics as they only describe word similarities, making it difficult for a model to understand NL text, leading to a failure to capture entity relationships [19,2].
Requirement Diagram	• Factors such as writing style and domain expertise can vary the input text's quality, directly affecting the generated diagram's quality. Developing accurate diagrams requires obtaining a sufficient corpus of text documents, which can be challenging in specific domains or topics [29].
Internal Block Diagram	• Like Block Definition Diagrams, writing style, domain expertise, and document complexity affect the quality and accuracy [19,29].

3.4 Motivation for Research

Several challenges have been reported in transforming natural language descriptions into automated system architectural diagrams. These include the inherent complexity of natural language, the difficulty in maintaining rule-based systems updated and working, the strictness of these rules, and the amount of data needed to make machine learning models more effective. Based on the literature review, approaches have been implemented to overcome these problems by utilizing NLP techniques such as tokenization, spell-checking, parsing, and named-entity recognition to refine the accuracy of rule-based methods. Additionally, machine learning techniques can be implemented with semantic analysis, deep learning, and NLP processing methods to help with the ambiguity and variability of natural languages. Hybrid models with rule-based and machine-learning methods demonstrate specific promises, leveraging each other's strengths to overcome limitations. The aim is to convert text-based descriptions into system models, like SysML and UML diagrams, more accurately and efficiently.

This thesis presents an architectural generation framework that addresses several challenges in automating system architectural generation from natural language descriptions, as observed in the literature. Challenges the proposed framework addresses are the complexity of understanding complicated language [12], the limitations of inflexible rules [34], the need for extensive data for training machine learning models [35], and the interpretation of SysML diagrams [16]. The framework allows the flexibility of processing language change and structure by utilizing spaCy’s NLP techniques, which include tokenization, part-of-speech tagging, lemmatization, and dependency parsing [36]. It improves the accuracy of extracting complex semantics, enhances the quality and relevance of the generated architectural diagrams, and addresses the limitations observed in literature from rule-based and machine-learning techniques.

By utilizing spaCy's capabilities to understand complex syntactic and semantic language, the framework addresses challenges such as ambiguities in the multiple meanings words or phrases have and inaccurately interpreting diverse textual descriptions. This helps generate SysML diagrams more efficiently and accurately, ensuring that the textual description is transformed into structured system models, demonstrating a valuable method for automating architectural diagrams using NLP.

The framework begins by normalizing text through cleaning and segmentation to simplify the inherent complexity of natural language into a uniform text base, improving accuracy. The spaCy NLP library is utilized for its advanced parsing capabilities, including POS tagging, dependency parsing, and noun chunking extraction, to overcome scalability and maintenance challenges, which enables the framework to process large volumes of text with high precision. Additionally, advanced visualization tools like displacy and a custom function tool showcase token relations and characteristics, which helps extract system modeling elements. This process ensures the accurate extraction of critical elements such as potential classes, methods, attributes, and relationships, addressing the nuances of natural language inconsistency and the scalability of rule-based systems. Chapter IV provides a detailed explanation of the methodology.

The challenges observed in the literature are categorized with a brief explanation of how this thesis, with the integration of the spaCy NLP library, addresses them.

3.5 Understanding the Complexities of Natural Language

Traditional NLP methods need help handling the inconsistencies and distinctions in natural language, which can lead to the problem of lexical ambiguity found in the text [12]. This leads to a need for computational approaches that can handle textual inconsistencies [37]. Using techniques such as Tokenization (breaking text into a block), Part-of-speech tagging (tagging words to clarify their functionality in a sentence), Lemmatization (ensuring consistent interpretation), and Dependency parsing (disambiguating sentences with semantically ambiguous subordinators [38]) by the NLP library spaCy, aids in understanding textual complexities and further clarify natural language while reducing ambiguity and further providing flexibility to understand linguistic contexts, a challenge observed typically in rule-based techniques when interpreting natural language.

3.6 Hybrid Model Potential

Creating complex system modeling diagrams using machine learning methods typically uses a significant amount of training data and examples to learn from, which can be challenging to gather. Gathering as much data and different input text as possible is essential to ensure proper coverage for a machine-learning model [13]. The more data the expert labels, the better the results are obtained [39]. The spaCy library in this thesis is explored to use pre-trained machine learning models [34] to organize and understand a text and mitigate the need for linguistic learning from scratch.

3.7 Enhanced Interpretability and User Friendly

The proposed architectural generation framework utilizes visualization tools, such as displacy tool and a custom token classifier function, to help stakeholders better glean information with a visual insight into the linguistic features of natural language text and the critical

information needed to identify system architecture elements. The framework's information extraction and selection process include user-friendly features that allow stakeholders to interact with the system directly, making diagram generation more accessible and adaptable to their needs. By allowing stakeholders to make corrections and adjustments during the generation process, the framework improves its overall satisfaction and user-friendliness in creating diagrams.

CHAPTER IV

METHODOLOGY

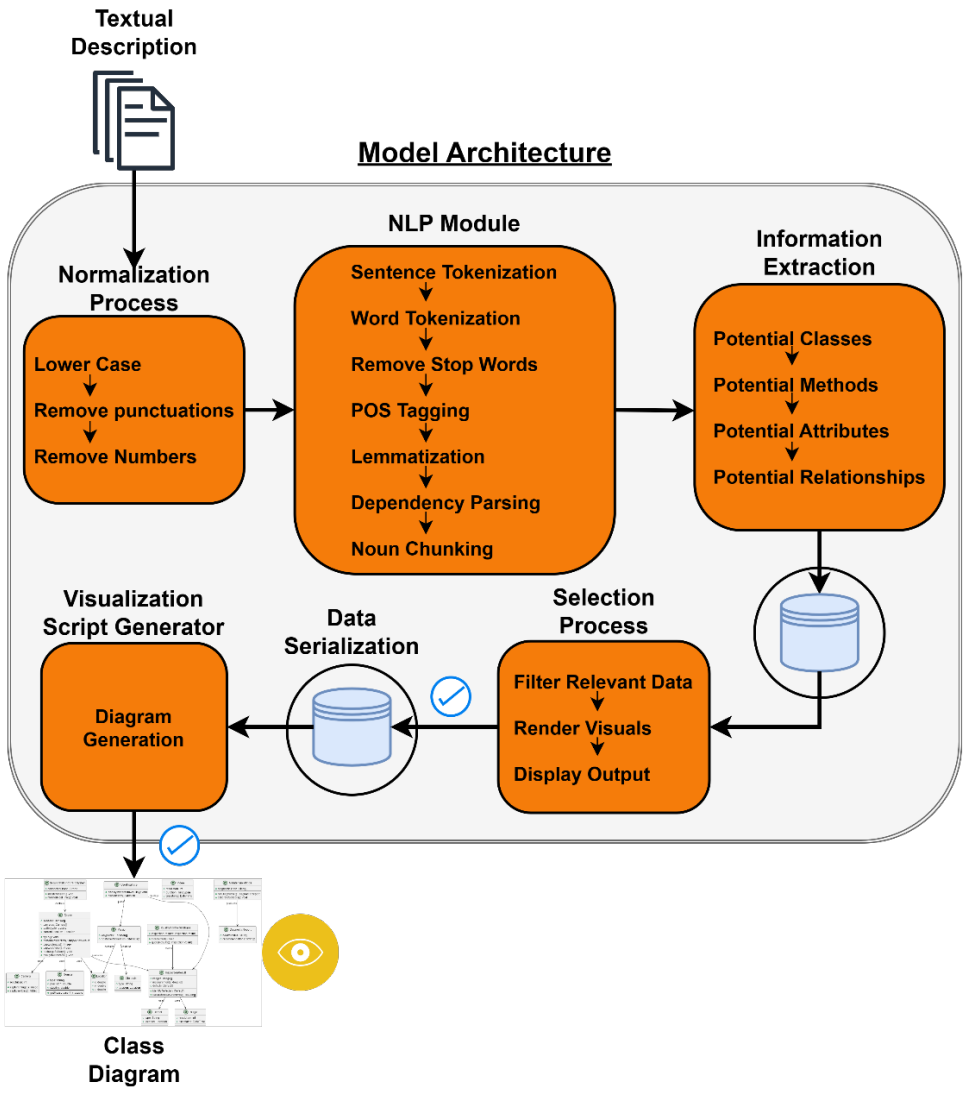

Figure 3. Model Representation of the Implemented Framework

The proposed methodology utilizes Python's programming strengths and spaCy's sophisticated natural language processing features. It specializes in tasks like tokenization, part-of-speech tagging, lemmatization, dependency parsing, and the attributes included with each task to process textual descriptions from .docx files for automatic system architectural representation. Spyder [40] and PyCharm [41] Community Edition IDEs are used for development along with PlantUML [42], a PyCharm add-in that generates system representations, alleviating manual labor in software design documentation and providing an effective transition from textual analysis to visual modeling.

4.1 Normalization Process

The first step involves processing textual documents using the Python Docx library to work with a Word document. This library can read paragraphs as structured objects from any Word document while maintaining the original format [43]. This step is crucial for the normalization as it is used to clean and structure the document for further analysis. The 'docx' library is imported to work specifically with Microsoft Word (.docx) files, allowing textual descriptions to be pulled from input text documents and prepare text for paragraph segmentation [44].

Once the textual requirements are gathered, the next step involves text cleaning and preprocessing to manipulate and prepare text for further analysis. These steps include transforming all words to lowercase [45], using the regular expression library tool for removing numerical data and pattern-matching techniques [46], using the string library tool to remove punctuation symbols [47], and using the spaCy library tool to identify and remove stop words [48]. These techniques help ensure uniformity and simplification within the text during the

normalization process, creating a solid foundation for the advanced parsing tasks that follow into the spaCy's NLP module.

Although the spaCy library can handle various text cleaning and preprocessing tasks on its own [49], establishing a text cleaning stage before NLP analysis ensures a smoother transition for further study, especially since SysML representations of a system architecture diagram typically excludes punctuations, stop words, and symbols.

4.2 Linguistic Features

Examples illustrate before-and-after states of a sample text on key linguistic features focusing on manipulating text, such as transforming lowercase, removing punctuation, removing stop words, and removing numbers for NLP tasks.

4.2.1 Transforming to Lowercase

Transforming text to its lowercase letters facilitates the preprocessing step by successfully enhancing classification accuracy [45, 50]. This technique involves transforming every sentence from text to all lowercase (using the text. lower () method), simplifying data for further analysis.

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Output: "the uav-101 model, designed for high-altitude surveillance, includes a 20mp camera and advanced navigation systems. the uav autonomously executes

missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."

4.2.2 Removing Punctuations

The process removes symbols (such as periods, commas, question marks, etc.) usually found in the text, such as requirement documents. This technique is crucial before the NLP process so that tokenization can focus only on alphabetic characters [50]. To achieve this, the string attribute utilizes its ‘string.punctuation’, removing any punctuation from text [47].

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Output: "The UAV101 model designed for high-altitude surveillance includes a 20MP camera and advanced navigation systems The UAV autonomously executes missions, adapting swiftly to environmental changes and submits data back to the central monitoring station"

4.2.3 Removing Numericals

This technique removes numbers (such as numerical lists, dates, time, ordinal numbers, etc.) that are found to have substantial syntactic properties in the text during preprocessing, which helps the NLP model focus on linguistic analysis when extracting relationships and behaviors. This method uses the regular expression function ‘re.sub’ to replace any numbers with an empty string, leaving only alphabetic characters [46]. Architectural diagrams usually exclude numerals from their designs. However, when specific numerical values are necessary to show

how a system functions, the regular expression function can remove and retain specific numerical for representation.

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Output: "The UAV- model, designed for high-altitude surveillance, includes a MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."

4.3 NLP Module

After the text is cleaned and prepared, the next step is to extract meaningful information. For this step, spaCy library was chosen over NLTK because of the user-friendly quality the library offers. The library provides an object-oriented approach rather than serving as a tool [49]. It's well known for its speed and efficiency when processing large amounts of textual data, making it an ideal choice for system requirements [51].

The main objective of information extraction is to establish a pipeline that analyzes text through different NLP tasks such as tagging, parsing, lemmatization, and named entity recognition. These pipelines can be tailored by preference regarding specific languages, capability features, the type of text it's trained on, and the package size [51]. This research will be using the "en_core_web_sm" trained model, which uses an English web text pipeline for the NLP features and because of its optimal balance between speed execution and accuracy the NLP

tasks have to offer [36] [51]. These tasks, such as parts of speech tagging and identifying dependencies from text, are instrumental in identifying potential classes, methods, attributes, and relationships within a text input.

The first step is to examine the text and split it into sentences to determine the sentence boundaries and define the word structuring. Each sentence then undergoes word tokenization, where every word is broken down into building blocks [17]. These two steps are crucial in creating a linguistic and semantic structure for both sentence and word levels. The next step is to analyze each token through several attributes: its part of speech (POS), its role in sentence structure (DEP), its base form of the word (LEMMA), and the token itself with each sentence. This process involves iterating over each token to capture and record these essential details and organize each token by successfully storing them in a structured format. This analysis provides a clear overview of how linguistic structure can be found within input text, which aids in extracting elements for system architectural representation. Figure 4 shown below illustrates the NLP pipeline and the tasks being used to produce a document object.

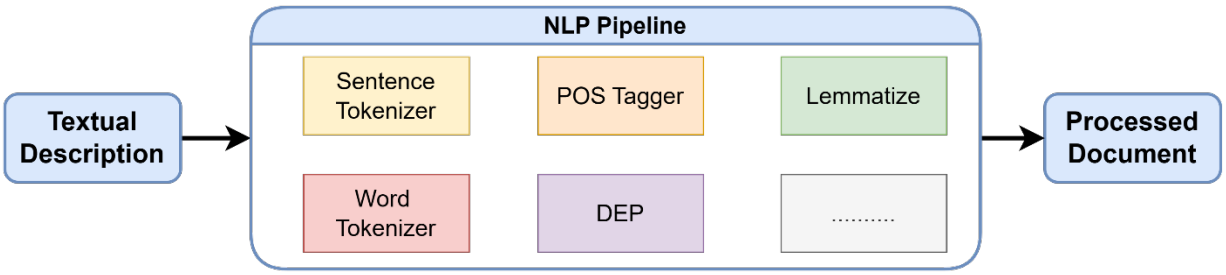

Figure 4. SpaCy's NLP Pipeline.

4.3.1 Sentence Tokenization

Sentence tokenization allows for examining each sentence as a distinct unit, which is essential for further analysis of input text [17]. It ensures a more focused and accurate analysis in identifying specific elements and relationships that assist in creating accurate architectural

diagrams. After the paragraphs are cleaned, the command ‘for sentence in doc.sents:’ is used to iterate over each sentence to tokenize the text [52].

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Output: {‘The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems.', 'The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station.'}

4.3.2 Word Tokenization

Sentence tokenization breaks down the text into fundamental building blocks known as tokens [49]. Word tokenization involves segmenting the text into words, punctuation, numbers, etc., found within sentences throughout an input text document. This breakdown is necessary for understanding the grouping of tokens to recognize relationships, a crucial step in extracting meaningful elements necessary to create architectural diagrams. Using the command ‘doc = nlp (cleaned paragraph),’ the cleaned text undergoes tokenization, resulting in a ‘doc’ object [36]. Subsequently, the tokens can be filtered (by utilizing the ‘for token in doc:’ command), which iterates and identifies specific tokens depending on the rules applied [52]. Organizing the text through tokenization lays the foundation for further analysis, paving the way toward insightful architectural generation.

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions,

adapting swiftly to environmental changes, and submits data back to the central monitoring station."

- Output: {'The', 'UAV-101', 'model', ',', 'designed', 'for', 'high', '-', 'altitude', 'surveillance', ',', 'includes', 'a', '20MP', 'camera', 'and', 'advanced', 'navigation', 'systems', '.', 'The', 'UAV', 'autonomously', 'executes', 'missions', ',', 'adapting', 'swiftly', 'to', 'environmental', 'changes', ',', 'and', 'submits', 'data', 'back', 'to', 'the', 'central', 'monitoring', 'station', '.'}

4.3.3 Removing Stop Words

Stop word removal implies removing commonly used words (such as the, in, a, an, etc.) not helpful in extracting knowledge [20]. This is crucial for transforming natural language text into a readable format so machine learning models can focus on meaningful text components [27]. In this thesis, spaCy's attribute 'token.is stop' filters stop words [48].

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."

- Output: {"UAV-101 model, designed high-altitude surveillance, includes 20MP camera advanced navigation systems. UAV autonomously executes missions, adapting swiftly environmental changes, submits data central monitoring station."}

4.3.4 Part of Speech (POS) Tagging

Speech tagging is necessary to identify the grammatical role of each token within the sentence and determine whether it's a noun, verb, adjective, etc. [53]. It is essential to understand the structure and meaning of a sentence. The syntactic structure of these tokens plays a crucial

role in determining what words can be considered for potential class names, actions, or methods and identifying relationships and attributes. By distinguishing the syntactic structure, POS tagging can precisely map textual descriptions to determine which tokens can be assigned to the architectural diagrams. POS tagging is performed (using the attribute ‘token.pos’) to access each token’s grammatical structure based on applied rules [34].

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Output: {"The" (DET), "UAV-101" (PROPN), "model" (NOUN), "designed" (VERB), "for" (ADP), "high" (ADJ), "altitude" (NOUN), "surveillance" (NOUN), "includes" (VERB), "a" (DET), "20MP" (PROPN), "camera" (NOUN), "and" (CCONJ), "advanced" (ADJ), "navigation" (NOUN), "systems" (NOUN), "The" (DET), "UAV" (PROPN), "autonomously" (ADV), "executes" (VERB), "missions" (NOUN), "adapting" (VERB), "swiftly" (ADV), "to" (ADP), "environmental" (ADJ), "changes" (NOUN), "and" (CCONJ), "submits" (VERB), "data" (NOUN), "back" (ADV), "to" (ADP), "the" (DET), "central" (ADJ), "monitoring" (NOUN), "station" (NOUN)}

These POS tag descriptions distinguish lexical and grammatical properties of words [51]. The descriptions are as follows adjective (ADJ), Determiner (DET), adverb (ADV), auxiliary (AUX), coordinating conjuncture (CCONJ), adposition (ADP), interjection (INTJ), noun (NOUN), numerical (NUM), particle (PART), pronoun (PRON), proper noun (PROPN), punctuation (PUNCT), symbol (SYM), subordination conjunction (SCONJ), and verb (VERB) [54].

4.3.5 Lemmatization

Lemmatizing transforms words based on their tagged token to their root or dictionary form based on their POS tag [17]. This step normalizes words such as changing “installed” to “install” or “engines” to “engine,” ensuring consistency and uniformity in the terminology used across diagram elements. It is performed by iterating over tokens in a sentence (using the command ‘token.lemma_’) to access each token’s base form [34]. This method helps refine the extraction process, permitting a more accurate depiction of classes, methods, and other architectural elements.

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Output: {"the" "UAV-101" "model" "," "design" "for" "high" "-" "altitude" "surveillance" "," "include" "a" "20MP" "camera" "and" "advanced" "navigation" "system" "." "the" "UAV" "autonomously" "execute" "mission" "," "adapt" "swiftly" "to" "environmental" "change" "," "and" "submit" "datum" "back" "to" "the" "central" "monitoring" "station" "."}

4.3.6 Dependency Parsing

Dependency parsing builds a categorized tree of relationships between tokens, illustrating their grammatical structure and clarifying how words interact [34]. This tree analogy helps determine and visualize connections like trailing family ties, which helps identify relationships between entities within the text. It analyzes which tokens function as modifiers or are associated with others to confirm potential attributes or relationships in architectural diagrams. In general,

dependency parsing focuses on how child tokens are associated with their parent tokens, revealing their contribution to the overall meaning of a sentence. This method (using the command ‘token.dep_’) helps distinguish syntactic roles, including subjects, objects, and modifiers, essential for mapping characteristics and relationships within system diagrams [34].

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Output: {"The" (det)}, {"UAV-101" (compound)}, {"model" (nsubj)}, {"," (punct)}, {"designed" (acl)}, {"for" (prep)}, {"high" (amod)}, {"-" (punct)}, {"altitude" (compound)}, {"surveillance" (pobj)}, {"," (punct)}, {"includes" (ROOT)}, {"a" (det)}, {"20MP" (amod)}, {"camera" (nmod)}, {"and" (cc)}, {"advanced" (conj)}, {"navigation" (compound)}, {"systems" (dobj)}, {"." (punct)}, {"The" (det)}, {"UAV" (nsubj)}, {"autonomously" (advmod)}, {"executes" (ROOT)}, {"missions" (dobj)}, {"," (punct)}, {"adapting" (advcl)}, {"swiftly" (advmod)}, {"to" (prep)}, {"environmental" (amod)}, {"changes" (pobj)}, {"," (punct)}, {"and" (cc)}, {"submits" (conj)}, {"data" (dobj)}, {"back" (advmod)}, {"to" (prep)}, {"the" (det)}, {"central" (amod)}, {"monitoring" (compound)}, {"station" (pobj)}, {"." (punct)}

These syntactic dependency descriptions help correlate the relation between tokens [51]. The descriptions are as follows: determiner (DET), adjectival modifier (AMOD), the object of a preposition (POBJ), adverbial modifier (ADVMOD), clausal complement (CCOMP), direct object (DOBJ), prepositional modifier (PREP), compound modifier (COMPOUND), clausal

modifier of a noun (ACL), nominal subject (NSUBJ), open clausal complement (XCOMP), coordinating conjunction (CC), conjunct (CONJ), etc. [54].

4.3.7 Noun Chunking

Noun chunking splits sentences into noun phrases tagged with nouns from the POS tagger, typically comprising a head noun and its modifiers [34]. This method highlights critical entities and their descriptions, helping extract information and providing a deeper understanding of the text's structure. Noun chunking can be performed by using the 'for chunk in sentence.noun_chunks:' command after the text has been processed [52].

- Input: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Output: {"The UAV-101 model", "high-altitude surveillance", "a 20MP camera and advanced navigation systems", "The UAV", "missions", "environmental changes", "data", "the central monitoring station"}

4.4 Information Extraction

After the text is cleaned and processed, the next step is extracting information. Two powerful tools help in this process: 'displacy' and a custom function named 'show tokens' to reveal the text structure, guiding the extraction of essential information for architectural representation diagrams.

The 'displacy' tool is a feature within the spaCy library that connects word relationships within a sentence, known as a dependency tree [36]. It visualizes how tokens are related, using

arrows to connect child tokens to their parents, illustrating a sentence's grammatical structure. This visualization helps understand how words work together, which can be used to define attributes and identify relationships between potential classes in the system.

The 'show tokens' function acts as a guide showing insights into each token's characteristics. It is used to shed light on a token's part of speech, its base form (or lemma), and how it connects to other tokens (its dependencies), which all three are displayed side by side for a better understanding of the extraction. This function offers insights that help navigate the text more effectively using an in-depth detail token analysis.

Both tools do more than help dissect sentences; they help chunk nouns representing potential classes in a system and identify verb-noun pairs that capture potential methods more broadly than isolated tokens. These tools' approach makes the syntactic structure of a sentence clear and helps us pull vital information from text, paving the way for creating accurate and insightful diagrams. The figures below illustrate a textual description example, showing how both tools are utilized.

Textual Description: "The engine drives the conveyor belt."

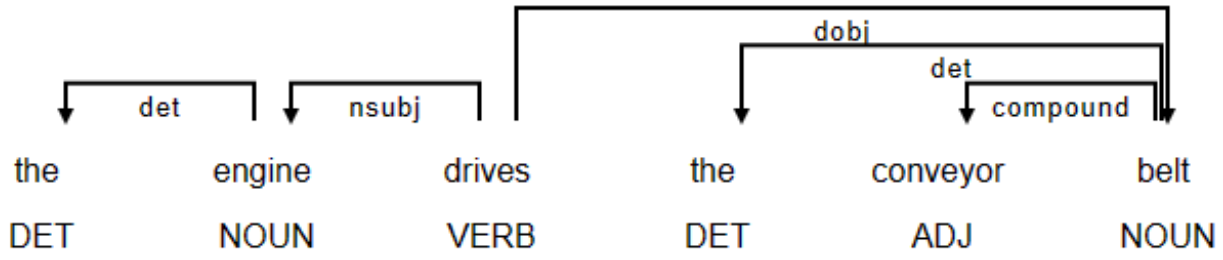

Figure 5: Displacy Visualization

Table 3. Show Tokens Tool.

Token	Lemma	POS	Dep
the	the	DET	det
engine	engine	NOUN	nsubj
drives	drive	VERB	ccomp
the	the	DET	det
conveyor	conveyor	ADJ	compound
belt	belt	NOUN	dobj

Figure 5 and Table 3 illustrate the linguistic analysis results from the textual description. Figure 5 shows a graphical representation of the sentence syntactic structure; it illustrates each word part-of-speech tag (such as NOUN, VERB, DET, etc.) and their grammatical relationships, which are connected by lines to represent the dependency relation (such as nominal subject 'nsubj', direct object 'dobj', etc.), crucial for understanding the sentences meaning and grammar. Table 3 showcases the descriptions list of tokens, lemmatized tokens, POS tags, and the dependencies representing the sentence's detailed linguistic breakdown, which is essential for identifying elements for extraction.

The element extraction framework begins by creating sets (using the 'set()' function) to store collections of potential classes, methods, attributes, and relationships. This ensures that

each collected element avoids duplication after extraction. These sets can be unordered collections of elements, which is critical for analyzing extracted elements from text [55].

4.4.1 Extracting Potential Classes

After cleaning, the text data is analyzed to extract potential classes from textual descriptions. It filters out any digits and the stop words and takes each token's lemmatized form in each sentence to refine the necessary tokens relevant to diagram modeling. The process of identifying potential classes is found by observing nouns that are singular, multiword, and chunking nouns which represent a mixture of simple and complex entities within a sentence [56]. The rules shown below illustrate how class elements can be found within textual descriptions alongside an example description.

- ***Class-Rule 1***: If the POS tag contains [NOUNS] or [PROPN], singular nouns are extracted for potential classes, capturing the lemmatized noun. These singular nouns form the base of potential class identification.
- ***Class-Rule 2***: Multiword nouns adjacent to each other, which represent class names such as [NOUN] [NOUN] or [NOUN] [PROPN] are extracted.
- ***Class-Rule 3***: Noun chunks that contain [NOUNS] and their modifiers are extracted to capture a class description. These chunks include a combination of [NOUNS] with their modifiers, which can be [VERBS], [ADJ], or both their combination. Noun chunking helps capture the full description of class attributes from simple to complex forms.
- ***Class-Rule 4***: If the DEP tag contains [nsubj] and [nsubjpass], their tokens are extracted for potential classes, capturing the lemmatized token. If modifiers such as [compounds] or [adj] precede the subjects, extract both subject and modifier to create a descriptive

entity. These subjects can be considered as potential entities since they are assigned to the syntactic heads of nouns or pro nouns.

The singular nouns, multiword nouns, and chunking nouns are then added to a list of potential classes, identifying the entities to be considered for the system.

- Example sentence: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Potential Classes Extracted: {'highaltitude', 'the central monitoring station', 'surveillance', 'datum', 'station', 'change', 'navigation system', 'navigation', 'the uav', 'mission', 'monitoring', 'monitoring station', 'environmental change', 'model', 'uav', 'a mp camera and advanced navigation system', 'highaltitude surveillance', 'system', 'the uav model', 'camera'}

4.4.2 Extracting Potential Methods

Potential methods are identified by examining verbs within sentences [53]. The process of identifying potential methods begins by analyzing the structure of a sentence, and looking for verbs that indicate actions alongside nouns or proper nouns as possible objects of these actions. A set list is created before forming verb-noun pairs to avoid word overlapping, helping capture unique pairing for each sentence being analyzed. This step helps examine the interactions within each sentence to find possible elements that are appropriate for potential methods. In each sentence, it filters out digits and stops words and takes the tokens lemmatized form. The rules shown below illustrate a description of how potential methods are found and extracted alongside an example description.

- ***Method-Rule 1***: Begin by creating a new set at the start of each sentence to capture verb-noun pairs to ensure the sentence is analyzed independently. This step helps prevent duplications in potential method identification and pairs actions with their objects.
- ***Method-Rule 2***: Singular [VERBS] are identified after iterating over each sentence token and added to the potential method set considering their lemmatized word. The [VERB] is also stored for a possible pairing with the following noun in the next step.
- ***Method-Rule 3***: Verb noun pairs are extracted by identifying a [NOUN] or [PROPN] that's found after a [VERB] from the pairing list to capture a potential method by combining an action and their object. This step takes both lemmatized words for the pair.
- ***Method-Rule 4***: The pairs are then added to a list of potential methods, identifying the actions and their respective objects within the system.
- ***Method-Rule 5***: To provide a deeper insight, this analysis can be extended to capture verb noun triples which find a [VERB] its direct [NOUN] and an additional [NOUN] to expand the context of understanding an action and extracting additional elements for a system being designed.

To illustrate an example

  - Example sentence: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
  - Potential Methods Extracted: {'design', 'include camera', 'submit station', 'design highaltitude', 'include systems', 'include uav', 'submit data', 'adapt', 'include

navigation', 'submit', 'adapt changes', 'include', 'execute missions', 'execute', 'submit monitoring', 'design surveillance'}

4.4.3 Extracting Potential Attributes

Potential attributes are found by examining words describing a class's characteristic or properties [57]. The process of identifying these attributes is by locating nouns modified by adjectives or, less commonly, by a compound relationship. This step analysis child tokens to identify class characteristics by examining the adjective or compound child tokens related to the noun, modifying them to present an accurate representation of attributes in a diagram. Using child tokens in the analysis helps identify a wide range of attributes that provide enhanced detailed descriptions of class characteristics. The rule shown below illustrates how potential attributes are identified and extracted, along with a description example.

- ***Attribute-Rule 1***: POS tagging is used to identify [NOUNS] after iterating over each token in a sentence. These nouns are used for the next step to help capture a representing attribute.
- ***Attribute-Rule 2***: For each [NOUN] captured, children tokens connected to it labeled [ADJ] or [compound] are identified with it. These modifiers provide a descriptive context for the noun.
- ***Attribute-Rule 3***: The modifiers collected with the noun create a complete attribute phrase. This phrase captures a more sophisticated description of the attribute.
- ***Attribute-Rule 4:*** These attributes are added to a set to ensure each is unique and not duplicated, creating a list of potential attributes.

- Example sentence: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
- Potential Attributes Extracted: {'missions', 'highaltitude', 'navigation systems', 'uav model', 'environmental changes', 'highaltitude surveillance', 'monitoring', 'data', 'camera', 'central monitoring station', 'navigation', 'monitoring station'}

4.4.4 Extracting Potential Relationships

Potential relationships are identified by connecting entities that associate with one another [53]. Patterns of verb-noun pairs are examined to extract potential relationships, suggesting a relationship between classes. Each verb identified is examined in connection with its dependency children, to determine if the token matches any nouns or proper nouns that could be used as subjects related to potential classes. Relationships are recorded between the verb in its lemmatized form and the child token's text when a match is found. This method can be further altered by identifying verb-noun pairs between classes, where the two nouns bridging the pair are considered potential classes from the first extraction step. This method can identify any relationship between classes stated in the text document, illustrating the system's structure. The rules below illustrate how potential relationships are identified and extracted, along with a description of the example.

- ***Relationship-Rule 1***: The text is scanned for [VERBS] using the POS tagger, identifying actions within sentences for relationship identification.

- ***Relationships-Rule 2***: For each [VERB] found, its child tokens are examined to see if it is associated with potential subjects. These subjects are considered among previously identified potential classes, such as [NOUNs] or [PROPN].
- ***Relationships-Rule 3***: The [VERB] is matched with any potential subjects to see if it acts upon any entity recognized as a class from the previous analysis.
- ***Relationships-Rule 4***: A relationships entry is created if a match is found, indicating that the subject is indeed recognized as a potential class. This entry pairs the [VERB] in its lemma form with the subject's text, capturing the action-entity relationship within the relationship set.
- ***Relationships-Rule 5***: Relationships can also be extracted by examining verb-direct object connections, capturing another action-entity within the text.
- ***Relationships-Rule 6***: For each [VERB] found, its [dobj] is identified among the verb's child tokens. If the direct object is found in relation to the verb, they are both paired in their lemma form, creating a relationship entry.
- ***Relationship-Rule 7***: Another relationship can be formed by identifying a [VERB] and a DEP tag [prep] that follows it. If the [prep] is present, both [VERB] and [prep] are paired creating an association within the text.
    - Example sentence: "The UAV-101 model, designed for high-altitude surveillance, includes a 20MP camera and advanced navigation systems. The UAV autonomously executes missions, adapting swiftly to environmental changes, and submits data back to the central monitoring station."
    - Potential Relationships Extracted: {'include systems'}, {'execute uav'}, {'submit data'}, {'include model'}, {'execute missions'}

4.5 Data Serialization

The data serialization step occurs before and after the selection process, which stores analysis in a JSON file for further use following the selection and PlantUML syntax generation. After applying the extraction method, all potential elements are verified through the JSON file and are pending to be selected in the selection process. JSON files were used in this research due to their simplicity and handling of large amounts of data to help confirm the extraction of essential elements [58]. This format offers the advantage of re-executing the extraction method with modified rules to find elements that were missed or not correctly captured, ensuring the data serialization remains current without storing any previous markers. After the selection process, the elements selected are stored in an object-oriented format to showcase which elements will be used through diagram generation for the system. The serialization data is then converted into PlantUML syntax through the visualization script generator for further diagram visualizations.

4.6 Selection Process

The selection process is an interactive interface that allows users to select suitable elements for diagram visualization. This process loads the JSON file from the extraction analysis and prompts users to create classes with their matching characteristics and operations. Users are instructed to select a new class and select methods and attributes belonging to that class. This step is repeated until all necessary class entities are created and move on to selecting relationships. Users are prompted to choose relationships and pair them with classes that have already been previously selected to create an association among the classes within the system. After all required elements are chosen and assigned, they are saved into a JSON file in object-oriented format. The file is then used in the visualization script generator, which successfully transforms the JSON data into a structured representation of a system.

4.7 Visualization Script Generator

The final step is the visualization stage, where the selected elements are generated into a system architecture. The visualization script generator operates by translating the selected elements stored from the selection process into PlantUML syntax required to create system architectures. PlantUML was chosen in this research because of its straightforward and user-friendly syntax, making it easy to create, share, and modify diagrams. It can handle JSON formats as they both use object-oriented representations, which can be transformed into visual diagrams to help understand and visualize data structures. PlantUML can also be integrated through various IDE's such as spyder or PyCharm to create and modify diagrams within the coding environment, making it an easy, productive process [59].

First, the selected elements chosen by the user are imported from the JSON file. The visualization script generator begins with an @startuml to indicate the beginning of the diagram definition. Multiword class names are formatted by replacing spaces with an underscore to ensure they are suitable identifiers in PlantUML code. Each class's methods and attributes that contain multiword spaces are also replaced with underscores to comply with PlantUML principle. Methods are joined immediately with parenthesis declaring a field name to signify their function nature, and all elements are then organized into an object-oriented blueprint under the corresponding class [59]. Relationships are directly included as strings in the PlantUML code after being formatted during the selection process to fit appropriate syntax. Once all classes and relationships have been added to the PlantUML code, the process concludes with an @enduml statement indicating the end of the diagram definition. Finally, the visual script generator outputs a printed PlantUML code for viewing. It can be manually copied to a PlantUML renderer using either software or a plugin used in PyCharm IDE to automate system architectural generation.

An example of PlantUML syntax below showcases how a hypothetical Class 1 is connected to Class 2, with both classes having methods and attributes.

```
@startuml

' Define Class1 with an attribute and a method
class Class1 {
    -attribute1 : DataType
    +method1() : ReturnType
}

' Define Class2 with an attribute and a method
class Class2 {
    -attribute2 : DataType
    +method2() : ReturnType
}

' Relation between Class1 and Class2
Class1 --> Class2 : relationLabel

@enduml
```

Figure 6: PlantUML Syntax According to Principle.

CHAPTER V

CASE STUDY AND EVALUATION

The case study in this section demonstrates the process and extraction of a requirement specification to generate a class diagram representation. A detailed case was generated to describe a UAV and its integration of advanced technology and operational needs, as shown in Figure 7. The requirement was examined and tested for validation through the approach in this chapter.

> The system integrates a sophisticated UAV network, streamlining both surveillance and logistical deliveries. Each UAV, at the core of the network, is outfitted with leading-edge navigational tech and cameras for comprehensive monitoring tasks, alongside cargo bays designed for precise delivery missions. Centralized control is managed through sophisticated software, enabling intricate data processing, UAV tracking, and task allocation in real-time. Surveillance operations utilize the UAVs' cameras to secure live feed analysis, while delivery missions are executed with pinpoint accuracy thanks to advanced GPS guidance. Data integrity and operational security are upheld through robust communication protocols, ensuring data protection and system reliability. Moreover, the system is adept at assigning UAVs dynamically, catering to emergent needs or enhancing operational efficacy. This UAV network epitomizes the fusion of aerial technology and data analytics, delivering unparalleled capabilities for contemporary surveillance and logistical needs.

Figure 7: UAV Requirement Specification.

In the processing stage, the visualization tools examine the tokens' grammatical structure and relationships, illustrating the POS and dependencies tags. Potential elements can be extracted from the token's examination to generate a class diagram.

```
Potential Classes or Entities: {'unparalleled capability', 'feed', 'delivery mission', 'tech', 'analytic', 'leadingedge navigational
tech', 'reliability', 'data protection', 'contemporary surveillance', 'leadingedge', 'tracking', 'surveillance', 'pinpoint accuracy
thank', 'need', 'task', 'datum', 'live feed analysis', 'cargo bay', 'efficacy', 'sophisticated software', 'intricate datum', 'emergent
need', 'core', 'task allocation', 'advanced gps guidance datum integrity', 'protocol', 'analysis', 'integrity', 'operational security',
'gps', 'guidance', 'system reliability', 'accuracy', 'delivery', 'comprehensive monitoring task', 'security', 'bay', 'control', 'thank',
'precise delivery mission centralized control', 'robust communication protocol', 'operational efficacy', 'network', 'logistical need',
'monitoring', 'uav tracking', 'cargo', 'mission', 'operation', 'logistical delivery', 'capability', 'communication', 'realtime
surveillance operation', 'uav', 'allocation', 'pinpoint', 'protection', 'system', 'technology', 'data', 'camera', 'software', 'fusion'}


Potential Methods or Behaviors: {'ensure system', 'cater', 'uphold communication', 'process tracking', 'outfit cameras', 'execute
pinpoint', 'process operations', 'design control', 'uphold', 'utilize cameras', 'epitomize', 'enable', 'process task', 'epitomize data',
'execute guidance', 'adept', 'outfit', 'ensure reliability', 'outfit tech', 'streamline', 'design', 'process surveillance', 'execute
security', 'deliver capabilities', 'deliver needs', 'cater needs', 'epitomize analytics', 'design delivery', 'streamline core', 'outfit
leadingedge', 'secure analysis', 'streamline network', 'process', 'secure missions', 'execute thanks', 'epitomize technology', 'manage
software', 'ensure data', 'integrate network', 'outfit monitoring', 'streamline deliveries', 'assign', 'streamline uav', 'streamline
surveillance', 'manage', 'execute accuracy', 'execute gps', 'outfit cargo', 'secure', 'enhance network', 'execute', 'utilize', 'enable
data', 'ensure protection', 'design missions', 'integrate', 'secure feed', 'enhance efficacy', 'process allocation', 'execute data',
'outfit bays', 'ensure', 'execute integrity', 'epitomize fusion', 'uphold protocols', 'outfit tasks', 'enhance', 'deliver surveillance',
'deliver', 'secure delivery'}


Potential Attributes: {'feed', 'delivery missions', 'uav network', 'aerial technology', 'data protection', 'contemporary surveillance',
'sophisticated uav network', 'surveillance', 'precise delivery missions', 'task', 'live feed analysis', 'sophisticated software',
'guidance data', 'core', 'emergent needs', 'logistical needs', 'task allocation', 'unparalleled capabilities', 'accuracy thanks', 'data
analytics', 'navigational tech', 'cargo bays', 'monitoring tasks', 'pinpoint accuracy', 'communication protocols', 'gps', 'operational
security', 'comprehensive monitoring tasks', 'gps guidance', 'intricate data', 'logistical deliveries', 'delivery', 'robust communication
protocols', 'advanced data integrity', 'system reliability', 'realtime surveillance operations', 'network', 'operational efficacy',
'monitoring', 'cargo', 'uav tracking', 'surveillance operations', 'uavs cameras', 'communication', 'centralized control', 'uav',
'pinpoint', 'system', 'data integrity', 'feed analysis', 'data', 'cameras', 'fusion'}


Simple Relationships:
[epitomize fusion] [streamline uav] [streamline surveillance] [integrate system] [process tracking] [secure analysis] [enable data]
[ensure protection] [utilize cameras] [deliver capabilities] [integrate network] [manage control] [adept system] [enhance efficacy]
[epitomize network]
```

Figure 8. Potential Elements Extracted

Figure 8 illustrates the text's potential classes, attributes, methods, and relationships using the aforementioned rules. These elements are saved to a JSON file for the next step, where the user can select possible elements for the selection process to generate a class diagram.

The user is prompted to create the first entity in the selection process. The process goes in order of creating a class, its methods, and its attributes, and then the user is asked if another class creation is necessary. Once all the classes and their elements are selected, the last step of the selection process is to select associations and link them with the chosen entities. Figure 9 illustrates the available elements that can be chosen for class selection. It depicts a list of nouns, noun chunks, and multiple nouns tagged from the textual requirement. Class selection is the only process that can choose more than one option for a single entity to distinguish a multiple-word entity that might not have been recognized from the extraction process.

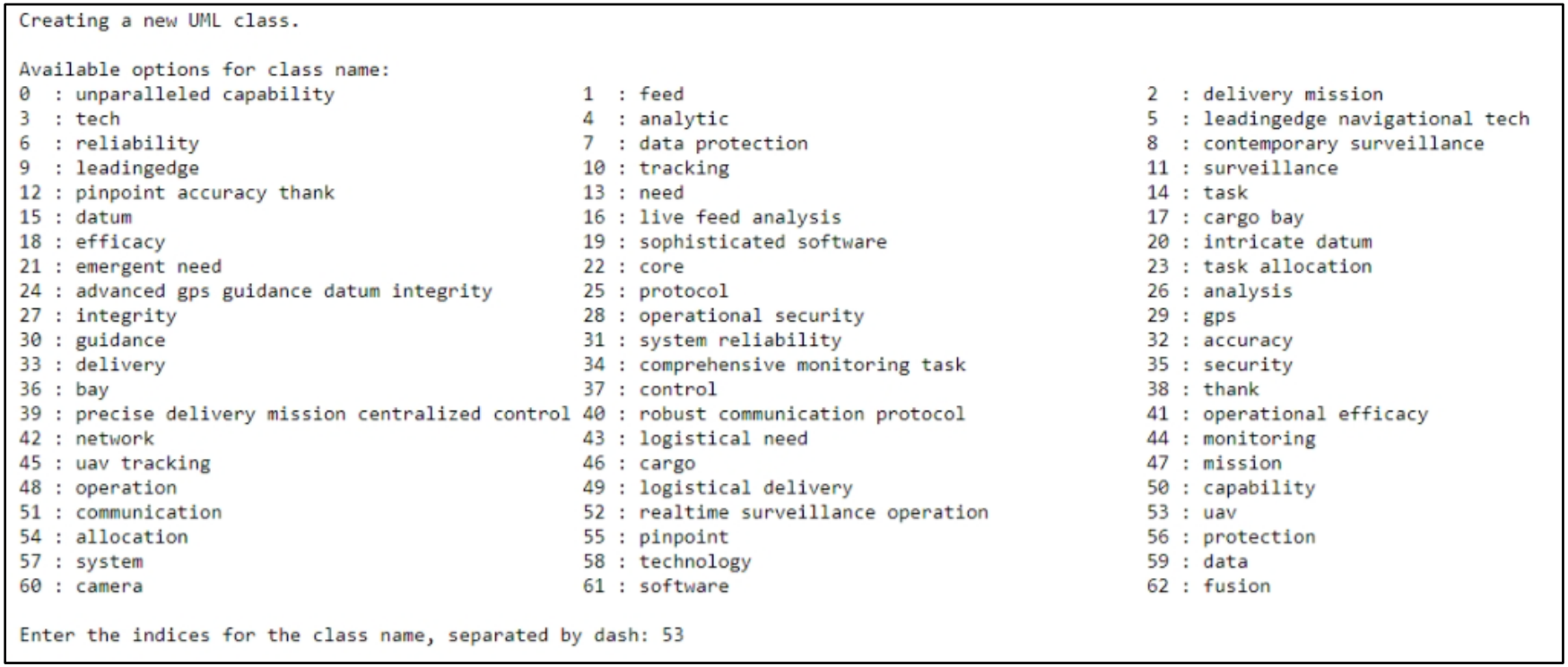

Figure 9: Class Selection.

Figure 10 shows the potential methods list. These lists of words were tagged by verbs that denote nouns to characterize an object's action. After choosing a class, the user can pick relevant methods, separated by dashes, to incorporate into the class. For instance, methods that pertain to the entity UAV based on the textual description can be that it executes a ‘pinpoint destination’ or ‘utilizes a camera.’

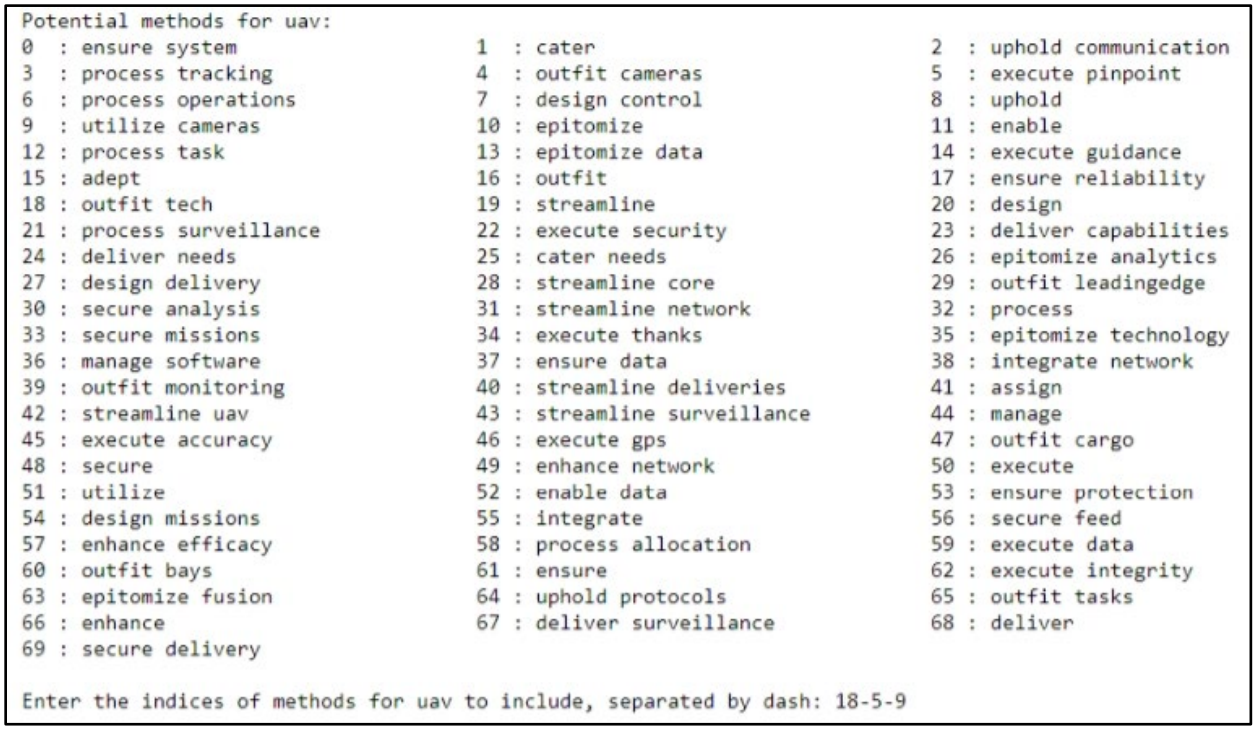

Figure 10: Methods Selection.

Figure 11 illustrates a list of potential attributes that can be selected from the textual description. Nouns tagged the list of words and their modifiers using children tokens to identify class characteristics representing properties that pertain to entities. After selecting the methods for a particular class, the user can pick as many attributes as possible, separated by dashes, to incorporate them into the specific class.

```
Potential attributes for uav:
0  : feed                          1  : delivery missions           2  : uav network
3  : aerial technology             4  : data protection            5  : contemporary surveillance
6  : sophisticated uav network     7  : surveillance               8  : precise delivery missions
9  : task                          10 : live feed analysis         11 : sophisticated software
12 : guidance data                 13 : core                       14 : emergent needs
15 : logistical needs              16 : task allocation            17 : unparalleled capabilities
18 : accuracy thanks               19 : data analytics             20 : navigational tech
21 : cargo bays                    22 : monitoring tasks           23 : pinpoint accuracy
24 : communication protocols       25 : gps                        26 : operational security
27 : comprehensive monitoring tasks 28 : gps guidance              29 : intricate data
30 : logistical deliveries         31 : delivery                   32 : robust communication protocols
33 : advanced data integrity       34 : system reliability         35 : realtime surveillance operations
36 : network                       37 : operational efficacy       38 : monitoring
39 : cargo                         40 : uav tracking               41 : surveillance operations
42 : uavs cameras                  43 : communication              44 : centralized control
45 : uav                           46 : pinpoint                   47 : system
48 : data integrity                49 : feed analysis              50 : data
51 : cameras                       52 : fusion

Enter the indices of attributes for uav to include, separated by dash: 20-51-21
```

Figure 11. Attributes Selection

```
Select relationships for the UML diagram:

Potential Relationships:
0  : ['epitomize', 'fusion']       1  : ['streamline', 'uav']          2  : ['streamline', 'surveillance']
3  : ['integrate', 'system']       4  : ['process', 'tracking']       5  : ['secure', 'analysis']
6  : ['enable', 'data']            7  : ['ensure', 'protection']      8  : ['utilize', 'cameras']
9  : ['deliver', 'capabilities']   10 : ['integrate', 'network']      11 : ['manage', 'control']
12 : ['adept', 'system']           13 : ['enhance', 'efficacy']       14 : ['epitomize', 'network']

Enter the indices of Relationships to include, separated by dash: 8
```

Figure 12: Relationship Selection

After collecting each entity's methods and attributes, the user can select and associate potential relationships with the previously chosen classes. These lists of relationships were tagged by verbs paired with nouns or direct objects using the dependency children to suggest possible subjects being associated between entities. Figure 12 illustrates the potential relationships found within the textual description. The complete set of relationships discovered in the requirement specification can be selected together, separated by dashes for the pairing step. After all the necessary relationships are selected, the user can combine each collection with two previously picked classes, creating an association between the entities.

After the user finishes selecting and pairing potential relationships between classes, the selections are saved into a JSON file as an object-oriented UML class representation. This file saves the selection data for the visualization script generator, which is considered the most critical phase, to generate the selections of PlantUML syntax for diagram generation. Figure 13 illustrates an example of the selection representation saved into the JSON data serialization.


```
{
    "classes": {
        "uav": {
            "methods": [
                "utilize cameras",
                "execute pinpoint",
                "outfit tech"
            ],
            "attributes": [
                "navigational tech",
                "cameras",
                "cargo bays"
            ]
        },
        "realtime surveillance operation": {
            "methods": [
                "process surveillance",
                "secure feed"
            ],
            "attributes": [
                "live feed analysis"
            ]
        }
    },
    "relationships": [
        "uav --> realtime_surveillance_operation : utilize cameras"
    ]
}
```


Figure 13. Class Representation Data Serialization.

Once the class representation data is saved, the visualization script generator converts it into appropriate PlantUML syntax and pastes its output. The syntax can then be copied and used in the same Python environment using the PlantUML plugin or in software for diagram generation. Figure 14 below illustrates the syntax output after using the script generator.

```
@startuml
class uav {
    navigational_tech
    cameras
    cargo_bays
    outfit_tech()
    execute_pinpoint()
    utilize_cameras()
}
class realtime_surveillance_operation {
    live_feed_analysis
    process_surveillance()
    secure_feed()
}
class delivery_mission {
    gps_guidance
    pinpoint_accuracy
    execute_pinpoint()
    streamline()
    deliver_needs()
}
class sophisticated_software {
    task_allocation
    uav_tracking
    manage_software()
    streamline_uav()
    process_operations()
    integrate_network()
    ensure_data()
    ensure_protection()
}
class robust_communication_protocol {
    data_protection
    operational_security
    uphold_protocols()
    uphold_communication()
    secure_missions()
}
class network_system {
    system_reliability
    operational_efficacy
    manage_software()
    enhance_network()
}
sophisticated_software --> uav : manage control
sophisticated_software --> uav : enable data
uav --> realtime_surveillance_operation : utilize cameras
uav --> delivery_mission : deliver capabilities
sophisticated_software --> robust_communication_protocol : ensure protection
uav --> network_system : epitomize network
delivery_mission --> uav : process tracking
sophisticated_software --> realtime_surveillance_operation : streamline surveillance
@enduml
```

Figure 14: PlantUML Syntax Generation.

After the extraction process, selection process, and the PlantUML syntax are finalized, the class diagram is generated for visualization. Figure 15 illustrates the class diagram representation of the UAV requirement specification. Each class shows the named entity located on the top, the methods towards the bottom of the class, and the attributes in between them. The

lines with an arrowhead going from one class to another represent the relationship between the two entities, labeled with an association description specifying their connection's nature.

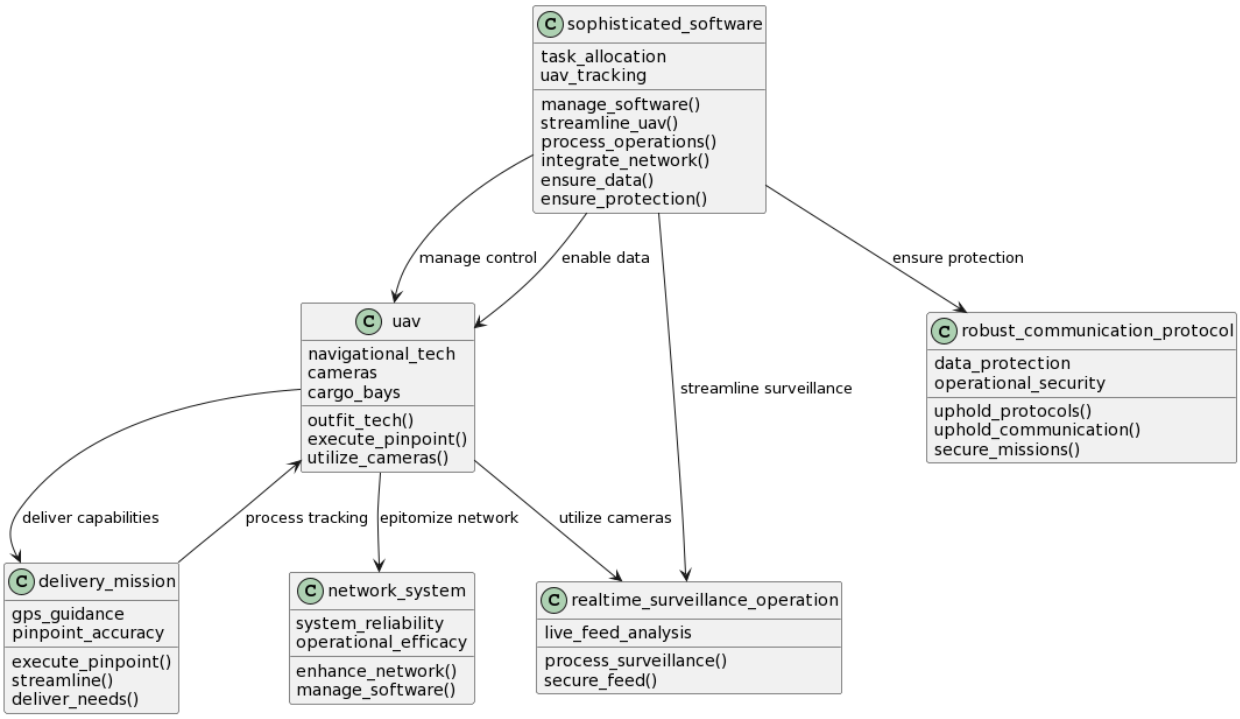

Figure 15. Class Diagram of Requirement Specification.

The elements extracted from the requirement specification are evaluated on their effectiveness in creating a class diagram. The essential elements utilized for the classes, methods, attributes, and relationships are measured to show the system's ability to identify them from the requirement text. The process aids in assessing the system's accuracy in classifying these elements, which offers insight into the precision of capturing the appropriate details needed for creating an accurate and comprehensive diagram.

This approach of extracting information in developing a class diagram doesn't focus only on word-for-word matches. It also employs an adaptable approach that accounts for semantic similarities. The approach recognizes phrases and terms in the textual description that can be

used as equivalent selections for the diagram with similar meanings. This strategy improves the system's precision in element extraction by recognizing relevant information that might be implied and explicitly mentioned in the description, which ensures an accurate and comprehensive representation even when the wording varies. This flexible interpretation helps with the system's performance and accuracy in capturing the essence of the described elements, providing an understanding of modeling complex textual data.

Precision and recall are vital measures to help understand how well the system identifies and selects appropriate elements [60, 61, 62]. These two metrics' flexibility will be used to evaluate the case study description and will only consider exact and semantically aligned information with the intended concept; the measurement equations to calculate the extracted analysis are shown below.

- Precision measures the ratio of correctly identified elements out of all elements, which helps identify the system's prediction by checking how many elements are relevant.

$$Precision = \frac{Number\ of\ correct\ items\ identified}{Total\ numbers\ of\ correct\ and\ incorrect\ items\ identified}$$

- Recall – This evaluates the system's ability to identify all relevant elements belonging to a particular class out of all the items that belong to that class. This metric checks how many of the correct items the system identified.

$$Recall = \frac{Number\ of\ correct\ items\ identified}{Total\ numbers\ of\ correct\ items}$$

The tables below show the identified elements found within the requirement text, along with their precision and metric ratings.

Table 4: Class Extraction Metrics

Sentences	Correct Classes	Extracted Classes	Precision	Recall
The system integrates a sophisticated UAV network, streamlining both surveillance and logistical deliveries.	“System” “Sophisticated uav Network” “Surveillance” “Logistical deliveries”	‘delivery’, ‘network’, ‘system’, ‘sophisticated uav network’, ‘surveillance’, 'logistical delivery'	0.66	1.0
Each UAV, at the core of the network, is outfitted with leading-edge navigational tech and cameras for comprehensive monitoring tasks, alongside cargo bays designed for precise delivery missions.	“UAV” “Network” “Navigational tech” “Camera” “Monitoring tasks” “Cargo bays” “Delivery missions”	delivery mission’, ‘tech’, ‘uav’, ‘delivery’, ‘comprehensive monitoring task’, ‘leadingedge navigational tech’, ‘core’, ‘monitoring task’, ‘leadingedge’, ‘task’, ‘network’, ‘bay’, ‘monitoring’, ‘cargo’, ‘mission’, ‘camera’, ‘precise delivery mission’, ‘cargo bay’	0.38	1.0
Centralized control is managed through sophisticated software, enabling intricate data processing, UAV tracking, and task allocation in real-time.	“Centralized control” “Sophisticated software” “Intricate data” “UAV tracking” “Task allocations”	‘sophisticated software’, ‘intricate datum’, ‘task’, ‘control’, ‘allocation’, ‘tracking’, ‘task allocation’, ‘uav tracking’, ‘centralize control’, ‘datum’, ‘software’	0.36	0.8
Surveillance operations utilize the UAVs’ cameras to secure live feed analysis, while delivery missions are executed with pinpoint accuracy thanks to advanced GPS guidance.	“Surveillance operations” “UAVs’ cameras” “camera” “Live feed analysis” “Delivery missions” “Advanced GPS guidance” “GPS guidance”	‘feed’, ‘delivery mission’, ‘surveillance’, ‘pinpoint accuracy thank’, ‘live feed analysis’, ‘surveillance operation’, ‘pinpoint accuracy’, ‘analysis’, ‘gps’, ‘guidance’, ‘accuracy’, ‘gps guidance’, ‘delivery’, ‘thank’, ‘mission’, ‘operation’, ‘accuracy thank’, ‘pinpoint’, ‘advanced gps guidance’, ‘uavs camera’, ‘feed analysis’, ‘camera’	0.22	1.0
Data integrity and operational security are upheld through robust communication protocols, ensuring data protection and system reliability.	“Data integrity” “Operational security” “Communication protocol” “Data protection” “System reliability”	‘communication’, ‘operational security’, ‘datum integrity’, ‘system reliability’, ‘security’, ‘reliability’, ‘data protection’, ‘robust communication protocol’, ‘communication protocol’, ‘system’, ‘protocol’, ‘data’, ‘datum’, ‘protection’, ‘integrity’	0.31	1.0
Moreover, the system is adept at assigning UAVs dynamically, catering to emergent needs or enhancing operational efficacy.	“System” “UAVs” “Emergent needs” “Operational efficacy”	'emergent need', 'system', 'operational efficacy', 'need', 'efficacy', 'uavs'	0.66	1.0
This UAV network epitomizes the fusion of aerial technology and data analytics, delivering unparalleled capabilities for contemporary surveillance and logistical needs.	“UAV network” “Fusion” “Ariel technology” “Data analytics” “Contemporary surveillance” “Logistical needs”	'unparalleled capability', 'analytic', 'data analytic', 'uav network', 'contemporary surveillance', 'network', 'logistical need', 'technology', 'data', 'capabilities', 'surveillance', 'need', 'aerial technology data analytic', 'fusion'	0.35	0.83

Table 4 shows the evaluation results of the two metric measurements used to identify correct classes from the textual description. The correct classes can be considered nouns or proper nouns found in the textual requirement. All classes were correctly identified for each sentence based on the recall measurement, and good results were generated for both metrics.

Table 5 showcases the metric results from the identified methods in the description. These methods are typically verbs paired with nouns to represent an action happening to an object. The table shows that the precision is relatively high apart from sentence three, which happened to extract more methods than needed for the diagram generation. Recall measurements generated good results as all methods were correctly identified from the description.

Table 6 shows the metric results for the identified attributes in the description. These are typically nouns modified by words that help describe a class's characteristics. The table shows that precision and recall have good results for all sentences, correctly extracting all identified attributes. Sentence six has the lowest ratings for both metrics as it did not capture one of the correct words in the description.

Table 7 illustrates the metric results for the identified relationships in the description. These are typically verb-noun pairs found between two classes to represent an association. The table shows high results for each sentence, showcasing that all relationships were correctly extracted from the description.

Table 5: Methods Extraction Metrics

Sentences	Correct Methods	Extracted Methods	Pre-cision	Re-call
The system integrates a sophisticated UAV network, streamlining both surveillance and logistical deliveries.	“Integrates network” “Streamlining surveillance” “Streamlining deliveries”	'streamline', 'streamline surveillance', 'integrate network', 'integrate', 'streamline deliveries'	0.6	1.0
Each UAV, at the core of the network, is outfitted with leading-edge navigational tech and cameras for comprehensive monitoring tasks, alongside cargo bays designed for precise delivery missions.	“outfitted tech” “outfitted cameras” “outfitted monitoring” “outfitted task” “design delivery”	'design', 'outfit', 'design delivery', 'design missions', 'outfit cameras', 'outfit tech', 'outfit cargo', 'outfit monitoring', 'outfit bays', 'outfit leadingedge', 'outfit tasks'	0.45	1.0
Centralized control is managed through sophisticated software, enabling intricate data processing, UAV tracking, and task allocation in real-time.	“enable” “enable data” “process”	'process tracking', 'manage', 'centralize', 'process', 'enable', 'enable datum', 'process allocation', 'process task', 'centralize control', 'manage software'	0.3	1.0
Surveillance operations utilize the UAVs’ cameras to secure live feed analysis, while delivery missions are executed with pinpoint accuracy thanks to advanced GPS guidance.	“utilize” “utilize camera” “secure” “secure feed” “executed” “execute GPS”	'utilize camera', 'secure analysis', 'secure', 'execute', 'utilize', 'secure feed', 'secure delivery', 'secure mission', 'execute pinpoint', 'execute guidance', 'execute accuracy', 'execute gps',	0.5	1.0
Data integrity and operational security are upheld through robust communication protocols, ensuring data protection and system reliability.	“uphold” “ensure” “ensure protection” “ensure reliability”	'ensure', 'ensure protection', 'uphold', 'uphold communication', 'uphold protocol', 'ensure system', 'ensure reliability', 'ensure data'	0.5	1.0
Moreover, the system is adept at assigning UAVs dynamically, catering to emergent needs or enhancing operational efficacy.	“Adept” “Assign” “Catering” “Catering needs” “Enhance” “Enhancing efficacy”	'catering', 'enhance efficacy', 'assign', 'adept', 'enhance', 'cater need'	1.0	1.0
This UAV network epitomizes the fusion of aerial technology and data analytics, delivering unparalleled capabilities for contemporary surveillance and logistical needs.	“Epitomize fusion” “Epitomize” “Deliver capabilities” “Deliver”	'epitomize fusion', 'deliver capability', 'deliver', 'epitomize'	1.0	1.0

Table 6: Attribute Extraction Metrics

Sentences	Correct Attributes	Extracted Attributes	Precision	Recall
The system integrates a sophisticated UAV network, streamlining both surveillance and logistical deliveries.	“Sophisticated UAV network” “Logistical deliveries” “Surveillance”	'logistical deliveries', 'system', 'sophisticated uav network', 'surveillance'	0.75	1.0
Each UAV, at the core of the network, is outfitted with leading-edge navigational tech and cameras for comprehensive monitoring tasks, alongside cargo bays designed for precise delivery missions.	“navigational tech” “comprehensive monitoring tasks” “precise delivery missions” “cargo bays” “cameras	'comprehensive monitoring tasks', 'uav', 'delivery', 'core', 'network', 'monitoring', 'cargo', 'navigational tech', 'cargo bays', 'precise delivery missions', 'cameras'	0.45	1.0
Centralized control is managed through sophisticated software, enabling intricate data processing, UAV tracking, and task allocation in real-time.	“sophisticated software” “intricate data” “UAV tracking” “task allocation” “real-time”	'sophisticated software', 'centralized control', 'intricate data', 'task allocation', 'uav tracking', 'realtime', 'task'	0.71	1.0
Surveillance operations utilize the UAVs’ cameras to secure live feed analysis, while delivery missions are executed with pinpoint accuracy thanks to advanced GPS guidance.	“UAVs’ cameras” “live feed analysis” “pinpoint accuracy” “advanced GPS guidance” “delivery” “surveillance” “feed”	'feed', 'delivery missions', 'gps', 'delivery', 'pinpoint', 'advanced gps guidance', 'pinpoint accuracy', 'accuracy thanks', 'surveillance', 'surveillance operations', 'live feed analysis', 'uavs cameras'	0.58	1.0
Data integrity and operational security are upheld through robust communication protocols, ensuring data protection and system reliability.	“data integrity” “operational security” “robust communication protocols” “data protection” “system reliability”	'communication', 'operational security', 'robust communication protocols', 'system reliability', 'data protection', 'system', 'data integrity', 'data'	0.63	1.0
Moreover, the system is adept at assigning UAVs dynamically, catering to emergent needs or enhancing operational efficacy.	“assigning UAV’s” “dynamically” “emergent needs” “operational efficacy”	'operational efficacy', 'emergent needs'	1.0	0.5
This UAV network epitomizes the fusion of aerial technology and data analytics, delivering unparalleled capabilities for contemporary surveillance and logistical needs.	“aerial technology” “data analytics” “unparalleled capabilities” “contemporary surveillance” “logistical needs”	'uav network', 'aerial technology', 'contemporary surveillance', 'logistical needs', 'unparalleled capabilities', 'data analytics', 'data', 'fusion'	0.63	1.0

Table 7: Relationship Extraction Metrics

Sentences	Correct relationships	Extracted relationships	Precision	Recall
The system integrates a sophisticated UAV network, streamlining both surveillance and logistical deliveries.	“System integrates” “integrates network” “integrates” “streamlining”	'streamline surveillance', 'streamlining', 'integrates', 'integrate system', 'integrate network'	0.8	1.0
Each UAV, at the core of the network, is outfitted with leading-edge navigational tech and cameras for comprehensive monitoring tasks, alongside cargo bays designed for precise delivery missions.	“outfitted with” “designed for” “outfitted”	'outfitted with', 'outfit uav', 'design for', 'outfitted'	0.75	1.0
Centralized control is managed through sophisticated software, enabling intricate data processing, UAV tracking, and task allocation in real-time.	“managed through” “enabling” “enabling data” “manage”	'process in', 'manage through', 'process tracking', 'enabling data', 'manage control', 'manage', 'enabling'	0.57	1.0
Surveillance operations utilize the UAVs’ cameras to secure live feed analysis, while delivery missions are executed with pinpoint accuracy thanks to advanced GPS guidance.	“utilize camera” “utilize” “execute with” “execute” “secure analysis” “secure”	'execute with', 'utilize cameras', 'secure analysis', 'execute', 'execute missions', 'utilize', 'secure'	0.85	1.0
Data integrity and operational security are upheld through robust communication protocols, ensuring data protection and system reliability.	“upheld though” “upheld” “ensuring” “ensure protection” “ensure reliability”	'uphold integrity', 'uphold', 'uphold through', 'ensure protection', 'ensuring'	0.8	0.8
Moreover, the system is adept at assigning UAVs dynamically, catering to emergent needs or enhancing operational efficacy.	“adept at” “adept” “catering to” “enhancing efficacy”	'adept at', 'cater to', 'adept system', 'adept', 'enhance efficacy'	0.8	1.0
This UAV network epitomizes the fusion of aerial technology and data analytics, delivering unparalleled capabilities for contemporary surveillance and logistical needs.	“epitomizes fusion” “epitomize” “delivering capabilities”	'epitomize', 'epitomize network', 'epitomize fusion', 'deliver capabilities'	0.75	1.0

Table 8. Case Study 2 Extraction

Case Study 2: "Each product has a description, a price and a supplier. Suppliers have addresses, phone numbers, and names. Each address is made up of a street address, a city, and a postcode. Each product has a single supplier. There is nothing to stop a supplier supplying many products."		
Potential Classes	Potential Attributes	Potential Relationships
Correct: Product Supplier Address	**Correct:** Description Price Address Phone Number Name Street Address City Postcode Single supplier	**Correct:** Has a Has description Have Made of Stop Stop supplier Supply Supply product Is a
Extracted: 'products', 'numbers', 'phone number', 'phone', 'name', 'address', 'supplier', 'number', 'description', 'product', 'postcode', 'names', 'street', 'city', 'supplier supplier', 'addresses', 'street address', 'address phone', 'price', 'address phone number', 'suppliers'	**Extracted:** 'many product', 'product', 'address', 'postcode', 'price', 'street address', 'phone number', 'supplier', 'street', 'city', 'description', 'phone', 'supplier supplier', 'single supplier', 'name'	**Extracted:** 'have a' 'has description' 'has supplier' 'has a' 'is a' 'made a' 'supply products' 'stop supplier' 'stop a' 'supplying products' 'have numbers' 'has of' 'stop' 'made of' 'made each' 'has each' 'supply'

Table 9: Case Study 2 Metrics

Potential Elements	Precision	Recall
Classes/Entities	0.14	1.0
Attributes	0.6	1.0
Relationships	0.53	1.0

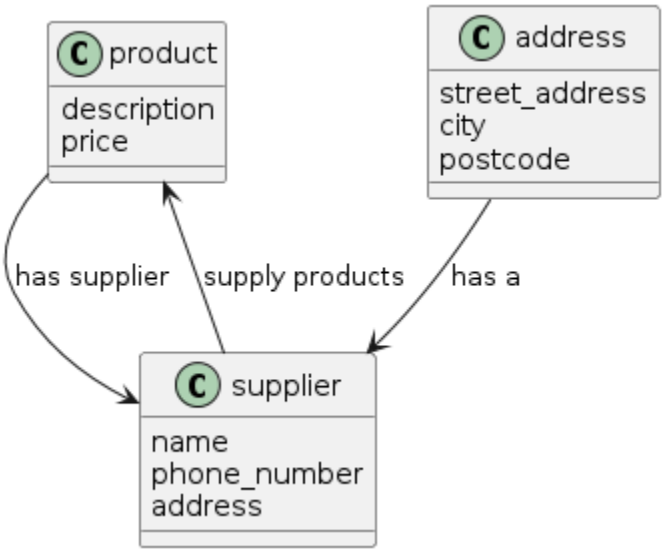

Figure 16. Class Diagram of Case Study 2.

Table 8 illustrates a new case study example along with its correct and extracted potential elements. The textual description was taken from example 1 used in Baginski thesis [62]. Table 9 showcases the metric evaluation using the rules applied to extract these potential elements, followed by the class diagram construction of case study 2, as shown in Figure 16. The precision and recall values compared to Baginski's findings for this example showed to get a precision of (1.0, 0.88, 1.0) and a recall of (1.0, 1.0, 1.0) for the classes, attributes, and relationships respectively [62]. The metric results shown in Table 9 display that the system correctly identified all correct elements by having a recall of 100% and having low values for the precision, which indicates the system captured a broad list of potential elements.

Table 10: Case Study 3 Extraction

Case Study 3:

"Participants at the summer school are either students or teachers. Each student registers for the NEMO Summer School providing, amongst others, their level of study (Bachelor, Master or PhD) and their field of study. Additionally each student provides her/his first name, last name, their country of provenience and e-mail address. Students attend courses during the summer school. Courses can be a lecture, a fundamentals exercise or application exercises. [The fundamental exercise is considered as one unit as it covers one topic, although it takes place in several sessions.] Each course has a title, is being given by one or more lecturers and takes places in a room. Every room has a name, a seating capacity, and technical equipment. Lectures and application exercises take place in a lecture hall, while fundamental exercises are conducted in PC-labs. Within the fundamentals exercise students are split in groups. Each group has a group number, a room (i.e. PC-lab) and a tutor. Teachers can be either lecturers or tutors. Each teacher has a first name, last name, host institution, and country."

Potential Classes	Potential Attributes	Potential Relationships
Correct: 'Participant', 'Student', 'Teacher', 'Course', 'Room', 'Group', 'Session', 'Application Exercises', 'Fundamental Exercise', 'Lecturer', 'PCLab', 'Tutor'	**Correct:** 'First name', 'Last name', 'Country', 'Email address', 'Level of study', 'Field', 'Country of provenience', 'Host institution', 'Title', 'Lecture', 'Room', 'Name', 'Seating capacity', 'Technical equipment', 'Lecture hall', 'PClab', 'Group number', 'Student', 'Tutor', 'Phd', 'Bachelors', 'Masters', 'Seating'	**Correct:** 'Attend', 'Attends courses', 'Registers', 'Given in', 'Given a', 'Takes place', 'Split students', 'Conduct exercises', 'Has a', 'Conducted in', 'Covers topic', 'Includes', 'Provide level', 'considered as'
Extracted: 'fundamentals', 'lecture', 'email address student', 'email', 'lectures', 'fundamental', 'bachelor', 'session', 'equipment lecture', 'exercises', 'bachelor master', 'courses', 'email address', 'tutors', 'school course', 'nemo', 'register', 'lecture hall', 'application exercise', 'course', 'address', 'students', 'tutor', 'participants', 'nemo summer', 'provenience', 'teachers', 'number', 'study bachelor', 'title', 'fundamental exercise', 'level', 'equipment', 'hall', 'unit', 'exercise student', 'address student', 'host institution', 'group', 'country', 'summer school', 'lecturers', 'student', 'places', 'it', 'pclab', 'lecturer', 'tutor teacher', 'participant', 'host', 'study bachelor master', 'technical equipment lecture', 'pclabs', 'phd', 'groups', 'seating capacity', 'teacher', 'institution', 'application', 'room', 'capacity', 'place', 'student register', 'seating', 'exercise', 'last name', 'registers', 'summer', 'study', 'topic', 'master', 'field', 'sessions', 'group number', 'school'	**Extracted:** 'lecture', 'email', 'equipment lecture', 'fundamental', 'bachelor', 'bachelor master', 'email address', 'school course', 'lecture hall', 'name', 'first name', 'application exercise', 'course', 'tutor', 'provenience', 'title', 'fundamental exercise', 'level', 'several session', 'unit', 'address student', 'exercise student', 'host institution', 'group', 'country', 'summer school', 'student', 'lecturer', 'pclab', 'tutor teacher', 'participant', 'host', 'phd', 'seating capacity', 'teacher', 'student register', 'other', 'application', 'place', 'room', 'seating', 'exercise', 'last name', 'summer', 'study', 'topic', 'technical equipment', 'field', 'group number'	**Extracted:** 'cover topic' 'provide' 'takes places' 'split a' 'given in' 'consider exercise' 'provides the' 'provides amongst' 'attend courses' 'split' 'provides name' 'take a' 'take place' 'provides herhis' 'attend during' 'name each' 'considered the' 'has every' 'cover' 'takes a' 'name for' 'conduct' 'split in' 'provides for' 'give title' 'take in' 'split each' 'attend students' 'has each' 'attend the' 'providing level' 'provide student' 'conduct exercises' 'attend' 'takes in' 'take within' 'provide level' 'name the' 'name amongst' 'name country' 'has a' 'covers in' 'take' 'name herhis' 'providing amongst' 'name of' 'consider' 'considered as' 'provides of' 'considered in' 'has room' 'name during' 'providing of' 'provides each' 'conducted in' 'conducted the' 'split students' 'has in' 'takes place' 'give' 'covers topic' 'has name' 'given a' 'take the' 'name at' 'take places' 'takes every' 'conducted within' 'has number' 'provides at' 'given every'

Table 11. Case Study 3 Metrics

Potential Elements	Precision	Recall
Classes/Entities	0.17	1.0
Attributes	0.45	0.96
Relationships	0.18	0.93

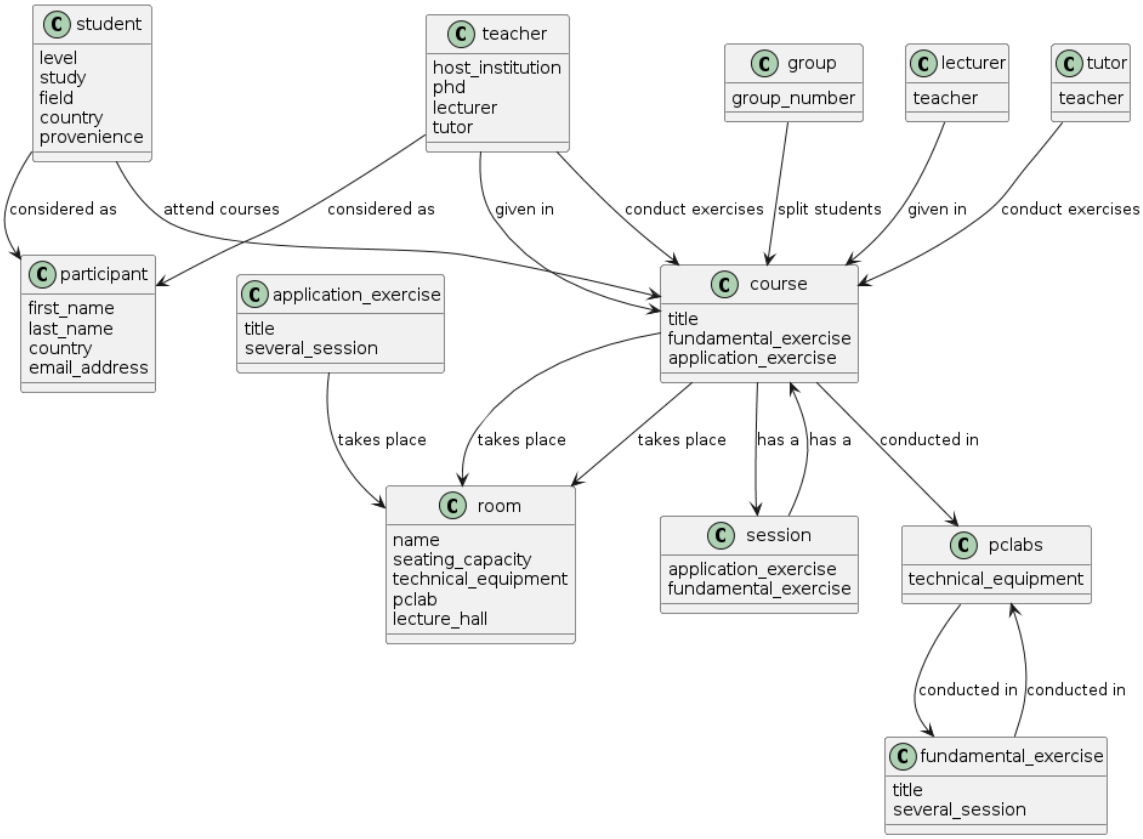

Figure 17. Class Diagram of Case Study 3.

Table 10 illustrates a third case study example taken from example 8 used in Baginski thesis [62], along with its correct and extracted potential elements. Table 11 showcases the metric evaluation using the rules applied to extract the potential elements followed by the class

diagram construction of case study 3 shown in figure 17. Baginski's precision and recall values for this example showed to get a precision of (0.71, 0.42, 0.63) and a recall of (0.77, 0.53, 0.42) for the classes, attributes, and relationships respectively [62]. The metric results shown in table 11 display that the system correctly identified more correct elements compared to Baginski's recall but still captured a broad list of potential elements as the values of the system's precision are low.

Table 12. Case Study 4 Extraction

Case Study 4:		
"A database for storing a private collection of books is required (yours or somebody else's). It should function both as an inventory of books as well as a help to find books to read. This database should focu on books and provide information surrounding those. Consider that books come in different editions (1s 2nd etc.) and each edition is owned once at most. Some books are written by one author, others by man and others can also have editors. The existence of series (Dark Tower, Discworld etc.) being a collectio several books should also be considered in the data structure. Additionally the data structure should sup finding books for a specific mood. It should also be possible to enter books that are not yet obtained, bu are planned to be added to the collection, or in other words a wish list. The wish list should contain som extra data (e.g. date added, a price limit etc.) in addition to the normal information about books."		
Potential Classes	Potential Attributes	Potential Relationships
Correct: 'Book', 'Author', 'Editor', 'Series', 'Mood', 'Wish List', 'data structure', 'inventory', 'edition', 'datum', 'database', 'price', 'help'	**Correct:** 'Date Added', 'Price Limit', 'different edition', 'specific mood', 'price', 'several books', 'wish', 'eg date', 'normal information', 'private collection', 'inventory', 'data', 'database', 'edition'	**Correct:** 'Written by', 'Have', 'Contain', 'have editors', 'provide information', 'own edition', 'focus on', 'enter books', 'find books', 'finding books', 'support', 'function as', 'added to', 'add', 'find books'
Extracted: 'collection', 'wish', 'data structure', 'existence', 'wish list', 'series dark', 'date', 'nd', 'author', 'dark', 'help', 'datum', 'edition st', 'st nd', 'inventory', 'addition', 'information', 'eg', 'limit', 'datum eg', 'series', 'dark tower', 'somebody', 'tower discworld', 'st', 'structure', 'it', 'most book', 'eg date', 'list', 'price limit', 'database', 'word', 'book', 'different edition', 'edition', 'mood', 'series dark tower discworld', 'price', 'editor', 'data', 'discworld', 'that', 'tower'	**Extracted:** 'collection', 'data structure', 'wish', 'existence', 'wish list', 'author', 'help', 'datum', 'normal information', 'specific mood', 'inventory', 'addition', 'information', 'datum eg', 'most book', 'eg date', 'price limit', 'database', 'book', 'different edition', 'edition', 'other', 'price', 'private collection', 'editor', 'several book', 'data', 'other word', 'author other'	**Extracted:** 'provide' 'add' 'require database' 'added limit' 'focus on' 'support list' 'provide information' 'focus etc' 'contain' 'enter books' 'support structure' 'store collection' 'read database' 'have editors' 'read' 'surrounding those' 'considered in' 'contain list' 'require' 'come in' 'finding for' 'own' 'support' 'added to' 'function as' 'added in' 'own edition' 'have existence' 'contain date' 'storing collection' 'finding books' 'store' 'add limit' 'find books'

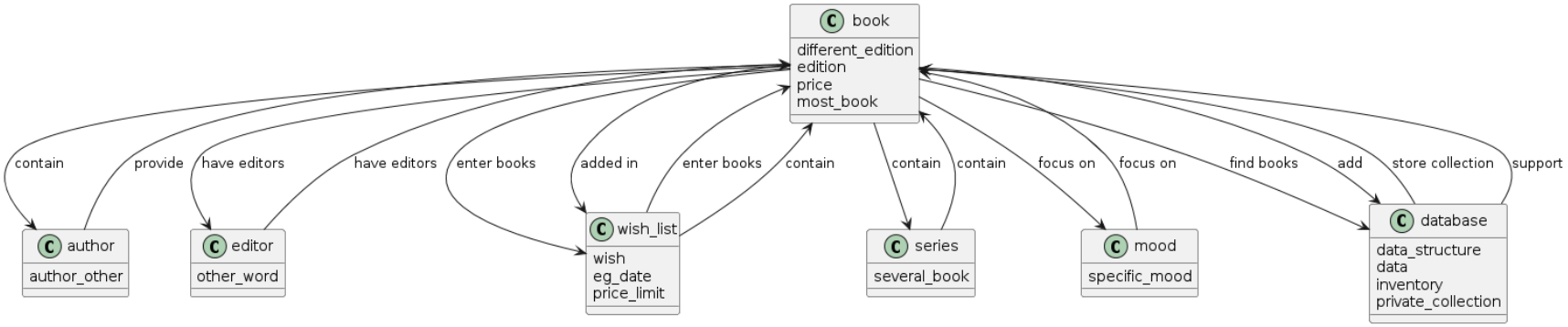

Figure 18: Class Diagram of Case Study 4.

Table 13: Case Study 4 Metrics

Potential Elements	Precision	Recall
Classes/Entities	0.3	1.0
Attributes	0.45	0.93
Relationships	0.38	0.93

Table 12 illustrates a fourth case study example taken from example 7 used in Baginski thesis [62], along with its correct and extracted potential elements. Table 13 showcases the metric evaluation using the rules applied to extract the potential elements followed by the class diagram construction of case study 4 shown in Figure 18. Baginski's results conducted for this example showed a precision of (0.38, 0.07, 0.20) and a recall of (0.75, 0.08, 0.25) respectively [62]. The metrics shown in Table 8 shows that the system's precision and recall have a better accuracy and exact extraction rate as the systems recall was a perfect score for the entities and almost perfect for the attributes and relationships. The system's precision did show to identify less incorrect items, since the values were higher for each elements precision compared to Babinski's results.

Table 14: Case Study 5 Extraction

Case Study 5: "A university consists of a number of departments. Each department offers several courses. A number of modules make up each course. Students enroll in a particular course and take modules towards the completion of that course. Each module is taught by a lecturer from the appropriate department, and each lecturer tutors a group of students."		
Potential Classes	Potential Attributes	Potential Relationships
Correct: 'university', 'department', 'course', 'module', 'student', 'lecturer'	**Correct:** 'department', 'appropriate department', 'course', 'several courses', 'particular course', 'modules', 'students', 'lecturer'	**Correct:** 'consists of', 'offers', 'enrolls in', 'takes module', 'teach', 'teach module', 'offers courses'
Extracted: 'student', 'course', 'module', 'completion', 'lecturer', 'course student', 'tutor', 'number', 'department', 'lecturer tutor', 'university', 'group'	**Extracted:** 'course', 'students', 'modules', 'particular course', 'completion', 'lecturer', 'appropriate department', 'number', 'department', 'lecturer tutor', 'several courses', 'university', 'group'	**Extracted:** 'offer' 'consists of' 'takes module' 'teach' 'consist university' 'enrolls in' 'make course' 'offers courses' 'take towards' 'consist' 'offer department' 'teach module'

Table 15: Case Study 5 Metrics

Potential Elements	Precision	Recall
Classes/Entities	0.5	1.0
Attributes	0.57	1.0
Relationships	0.58	1.0

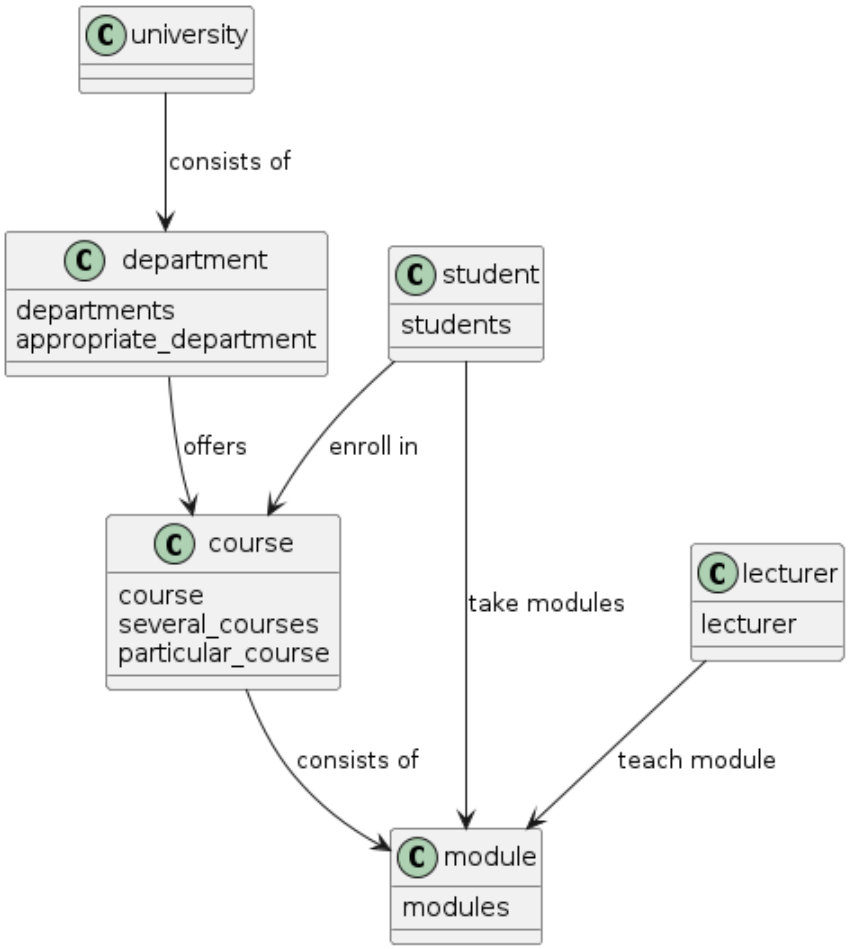

Figure 19: Class Diagram of Case Study 5.

Table 14 illustrates a fifth case study example taken from example 5 used in Baginski thesis [62], along with its correct and extracted potential elements. Table 15 showcases the metric evaluation using the rules applied to extract the potential elements followed by the class diagram construction of case study 5 shown in figure 19. Baginski's results conducted for example 5 showed a precision of (1.0, n/a, 1.0) and a recall of (1.0, n/a, 0.71) respectively [62]. Although this sentence is shown to be simple, the text only primarily outlines the entities and the associations between them which is why n/a for attributes is used in its evaluation. The metric results shown in table 15 display that the system correctly identified all correct elements by

having a recall of 100% for both classes and relationships compared to Baginski's methods. However, the system's precision was relatively low, which indicates that the system captured a broader list of potential elements than the other method.

It is shown through Baginski's method that the smaller and simpler the text description is, the easier it is to capture exact correct words and have a possible higher precision simultaneously. Examples 1 and 5 show the text to have a simple sentence structure compared to the others, which is why the system's recall performed better in extracting the correct elements for both examples. Example 7, the system experienced the same problem as Baginski's methods for mistakenly identifying "e.g." as an attribute, highlighting a challenge for refinement. The system identified numerous elements, for example, 7 and 8, because of the length and structure of the description and the rules being applied for the extraction process, which is ultimately why a low precision score is received. The rules used for extraction ensure that no possible elements have been overlooked, which shows that a broad list of elements can inspire more flexible design decisions. The low precision serves as a strategy by providing a more extensive set of elements that cater to a variety of user needs and scenarios in the development of a system diagram.

CHAPTER VI

CONCLUSION

This thesis systematically reviews and analyzes natural language processing techniques for the automated generation of SysML diagrams. It demonstrates common challenges with rule-base, machine learning, and hybrid methods when utilizing NLP to extract system requirements from text for generating modeling language diagrams. A rule-based method is used in this research to showcase the importance of information extraction in extracting vital system architectural elements. This thesis tackles the challenges against natural language complexity by simplifying the text through normalization and using a robust NLP library for its parsing capabilities to understand the structure of ambiguous linguistics better. It also demonstrates a potential hybrid method using pre-trained models and a list of rules that accurately extract class elements for diagram generation. The extraction and selection process aims to be simple and easy to understand so that the user can adjust it before processing the required elements into comprehensive PlantUML syntax for rendering.

Based on the case studies shown in Chapter V, the proposed method has demonstrated tremendous success in utilizing NLP to extract elements from complex descriptions. These elements are generated into system modeling diagrams according to subjectivity, allowing the creation of models with a range of diverse options. Additionally, the method can be explored further to reduce the number of rules used during the extraction process and serve as a fully automated SysML diagram generator. The versatility of rules used to extract

elements from textual descriptions of various sizes can help in the development process of a product through its entire lifecycle for multiple projects. It can be used as a vital tool for system engineers.

The rule-based approach provides consistency in mapping textual patterns to SysML diagram elements by evaluating their precision and recall measurements. The measurements underscore the accuracy of extracting correct semantic similarity elements from system requirements for modeling. Future work would address the rules used to extract potential elements to improve precision while maintaining recall, enhancing the systems extracting method, which can be used for a full-potential machine learning or hybrid approach.

REFERENCES

[1.] Rayhan, A., Kinzler, R., & Rayhan, R. (2023). Natural language processing: Transforming how machines understand human language. https://doi.org/10.13140/RG.2.2.34900.99200

[2.] Qie, Y., Zhu, W., Liu, A., Zhang, Y., Wang, J., Li, T., ... & Wang, Y. (2018, August). A Deep Learning Based Framework for Textual Requirement Analysis and Model Generation. In 2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC) (pp. 1-6). IEEE.

[3.] Johri, P., Khatri, S. K., Al-Taani, A. T., Sabharwal, M., Suvanov, S., & Kumar, A. (2021). Natural language processing: History, evolution, application, and future work. In *Proceedings of 3rd International Conference on Computing Informatics and Networks: ICCIN 2020* (pp. 365-375). Springer Singapore.

[4.] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, *82*(3), 3713-3744.

[5.] McGrath, A., & Jonker, A. (2023, December 2). What is model-based systems engineering (MBSE)? IBM. Retrieved June 2024, from https://www.ibm.com/topics/model-based-systems-engineering

[6.] Kaelble, S. (2022). MBSE For Dummies® (Siemens Special Edition). Hoboken, NJ: John Wiley & Sons, Inc.

[7.] Theobald, M., & Tatibouet, J. (2019, February). Using fUML Combined with a DSML: An Implementation using Papyrus UML/SysML Modeler. In MODELSWARD (pp. 248-255).

[8.] Hause, M. (2006, September). The SysML modelling language. In *Fifteenth European systems engineering conference* (Vol. 9, pp. 1-12).

[9.] Kulvatunyou, B., Ivezic, N., & Srinivasan, V. (2016). On architecting and composing engineering information services to enable smart manufacturing. *Journal of computing and information science in engineering*, *16*(3), 031002.

[10.] Alenazi, M., Niu, N., & Savolainen, J. (2019, September). SysML modeling mistakes and their impacts on requirements. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)* (pp. 14-23). IEEE.

[11.] Zhao, Liping, et al. "Natural language processing for requirements engineering: A systematic mapping study." ACM Computing Surveys (CSUR) 54.3 (2021): 1-41.

[12.] Zhong, Shaohong, Andrea Scarinci, and Alice Cicirello. "Natural Language Processing for systems engineering: Automatic generation of Systems Modelling Language diagrams." Knowledge-Based Systems 259 (2023): 110071.

[13.] Chami, M., Zoghbi, C., & Bruel, J. M. (2019). A First Step towards AI for MBSE: Generating a Part of SysML Models from Text Using AI. A First Step towards AI.

[14.] Ahmed, S., Ahmed, A., & Eisty, N. U. (2022, May). Automatic Transformation of Natural to Unified Modeling Language: A Systematic Review. In 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA) (pp. 112-119). IEEE.

[15.] Petrotta, M., Sterling Heights, M. I., & Peterson, T. (2019, July). Implementing Augmented Intelligence in Systems Engineering. In INCOSE International Symposium (Vol. 29, No. 1, pp. 543-543).

[16.] Narawita, C.R., & Vidanage, K. (2018). UML generator – use case and class diagram generation from text requirements. International Journal on Advances in Ict for Emerging Regions (icter), 10, 1.

[17.] Abdelnabi, E.A., Maatuk, A.M., Abdelaziz, T.M., & Elakeili, S.M. (2020). Generating UML Class Diagram using NLP Techniques and Heuristic Rules. 2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), 277-282.

[18.] Z. A. Hamza and M. Hammad, “Generating UML Use Case Models from Software Requirements Using Natural Language Processing,” 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO), Manama, Bahrain, 2019, pp. 1-6, doi: 10.1109/ICMSAO.2019.8880431.

[19.] Chen, Max & Bhada, Shamsnaz. (2022). Converting natural language policy article into MBSE model. INCOSE International Symposium. 32. 73-81. 10.1002/iis2.12897.

[20.] Shinde, S.K., Bhojane, V., & Mahajan, P. (2012). NLP based Object Oriented Analysis and Design from Requirement Specification. International Journal of Computer Applications, 47, 30-34.

[21.] Chen, Lei & Zeng, Yong. (2009). Automatic Generation of UML Diagrams From Product Requirements Described by Natural Language. 2. 10.1115/DETC2009-86514.

[22.] Bajwa, I.S., & Choudhary, M.A. (2006). Natural language processing based automated system for UML diagrams generation.

[23.] Arumugam, Chandrasekar & Uma, G.. (2006). Automatic Construction of Object Oriented Design Models [UML Diagrams] from Natural Language Requirements Specification. 4099. 1155-1159. 10.1007/11801603_152.

[24.] de Biase, Maria Stella & Marrone, Stefano & Palladino, Angelo. (2022). Towards Automatic Model Completion: from Requirements to SysML State Machines. 10.48550/arXiv.2210.03388.

[25.] Dawood, O. S. (2018). Toward requirements and design traceability using natural language processing. European Journal of Engineering and Technology Research, 3(7), 42-49.

[26.] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[27.] Kochbati, T., Li, S., Gérard, S., & Mraidha, C. (2021). From User Stories to Models: A Machine Learning Empowered Automation. MODELSWARD, 10, 0010197800280040.

[28.] Riesener, M., Dölle, C., Becker, A., Gorbatcheva, S., Rebentisch, E., & Schuh, G. (2021, July). Application of natural language processing for systematic requirement management in model-based systems engineering. In INCOSE International Symposium (Vol. 31, No. 1, pp. 806-815).

[29.] Zhong, S., Scarinci, A., & Cicirello, A. (2023). Natural Language Processing for systems engineering: Automatic generation of Systems Modelling Language diagrams. Knowledge-Based Systems, 259, 110071.

[30.] Seresht, S. M., & Ormandjieva, O. (2008). Automated assistance for use cases elicitation from user requirements text. In Proceedings of the 11th Workshop on Requirements Engineering (WER 2008) (Vol. 16, pp. 128-139).

[31.] Elallaoui, M., Nafil, K., & Touahni, R. (2018). Automatic transformation of user stories into UML use case diagrams using NLP techniques. Procedia computer science, 130, 42-49.

[32.] Osman, M. S., Alabwaini, N. Z., Jaber, T. B., & Alrawashdeh, T. (2019, April). Generate use case from the requirements written in a natural language using machine learning. In 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT) (pp. 748-751). IEEE.

[33.] Joshi, S. D., & Deshpande, D. (2012). Textual requirement analysis for UML diagram extraction by using NLP. International journal of computer applications, 50(8), 42-46.

[34.] Chantrapornchai, C., & Tunsakul, A. (2021). Information extraction on tourism domain using SpaCy and BERT. *ECTI Transactions on Computer and Information Technology*, *15*(1), 108-122.

[35.] Shah, U. S., Patel, S. J., & Jinwala, D. (2016). Specification of Non-Functional Requirements: A Hybrid Approach. In *REFSQ Workshops*.

[36.] Fantechi, A., Gnesi, S., Livi, S., & Semini, L. (2021, September). A spaCy-based tool for extracting variability from NL requirements. In *Proceedings of the 25th ACM International Systems and Software Product Line Conference-Volume B* (pp. 32-35).

[37.] Joseph, S. R., Hlomani, H., Letsholo, K., Kaniwa, F., & Sedimo, K. (2016). Natural language processing: A review. *International Journal of Research in Engineering and Applied Sciences*, *6*(3), 207-210.

[38.] Yamamoto, Y., Matsumoto, Y., & Watanabe, T. (2022). Dependency patterns of complex sentences and semantic disambiguation for abstract meaning representation parsing. *Journal of Natural Language Processing*, *29*(2), 515-541.

[39.] Chami, M., Abdoun, N., & Bruel, J. M. (2022, July). Artificial Intelligence Capabilities for Effective Model-Based Systems Engineering: A Vision Paper. In *INCOSE International Symposium* (Vol. 32, No. 1, pp. 1160-1174).

[40.] Spyder IDE Contributors. (2023). Spyder (Version 5.4.1) [Software]. Available from https://www.spyder-ide.org/

[41.] JetBrains. (2023). PyCharm 2023.2.1 (Community Edition) [Software]. Build #PC-232.9559.58, built on August 22, 2023. Retrieved from https://www.jetbrains.com/pycharm/

[42.] PlantUML Integration. (2023). PlantUML integration (Version 7.0.0-IJ2023.2) for PyCharm [Software plugin]. Available from JetBrains Marketplace.

[43.] Claghorn, R., & Shubayli, H. (2021, July). Requirement Patterns in the Construction Industry. In *INCOSE International Symposium* (Vol. 31, No. 1, pp. 391-408).

[44.] Kulkarni, A., & Shivananda, A. (2019). Natural language processing recipes. Apress.

[45.] Octavially, R. P., Priyadi, Y., & Widowati, S. (2022, November). Extraction of Activity Diagrams Based on Steps Performed in Use Case Description Using Text Mining (Case Study: SRS Myoffice Application). In *2022 2nd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)* (pp. 225-230). IEEE.

[46.] Mande, R., Yelavarti, K. C., & JayaLakshmi, G. (2018, December). Regular Expression Rule-Based Algorithm for Multiple Documents Key Information Extraction. In *2018 International Conference on Smart Systems and Inventive Technology (ICSSIT)* (pp. 262-265). IEEE.

[47.] Ismukanova, A. N., Lavrov, D. N., Keldybekova, L. M., & Mukumova, M. Z. (2018). USING THE PYTHON LIBRARY WHEN CLASSIFYING SCIENTIFIC TEXTS. In *EUROPEAN RESEARCH: INNOVATION IN SCIENCE, EDUCATION AND TECHNOLOGY* (pp. 9-13).

[48.] Srinivasa-Desikan, B. (2018). *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd.

[49.] Jugran, S., Kumar, A., Tyagi, B. S., & Anand, V. (2021, March). Extractive automatic text summarization using SpaCy in Python & NLP. In *2021 International conference on advance computing and innovative technologies in engineering (ICACITE)* (pp. 582-585). IEEE.

[50.] Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information processing & management*, *50*(1), 104-112.

[51.] Explosion AI. (n.d.). *spaCy: Industrial-strength Natural Language Processing in Python*. Retrieved April 4, 2024, from https://spaCy.io

[52.] Vasiliev, Y. (2020). *Natural language processing with Python and spaCy: A practical introduction*. No Starch Press.

[53.] Bashir, N., Bilal, M., Liaqat, M., Marjani, M., Malik, N., & Ali, M. (2021, March). Modeling class diagram using nlp in object-oriented designing. In *2021 National Computing Colleges Conference (NCCC)* (pp. 1-6). IEEE.

[54.] Universal Dependencies. (n.d.). *Dependency Relations Index*. Retrieved [03/20/24], from https://universaldependencies.org/u/dep/index.html

[55.] Sarkar, D. (2016). *Text analytics with python* (Vol. 2). New York, NY, USA:: Apress.

[56.] Herchi, H., & Abdessalem, W. B. (2012). From user requirements to UML class diagram. *arXiv preprint arXiv:1211.0713*.

[57.] Almazroi, A. A., Abualigah, L., Alqarni, M. A., Houssein, E. H., AlHamad, A. Q. M., & Elaziz, M. A. (2021). Class diagram generation from text requirements: An application of natural language processing. *Deep Learning Approaches for Spoken and Natural Language Processing*, 55-79.

[58.] Arachchi, K. D. (2022). *AI Based UML Diagrams Generator* (Doctoral dissertation).

[59.] PlantUML. (2009). Retrieved from https://plantuml.com/

[60.] Arachchi, K. D. (2022). *AI Based UML Diagrams Generator* (Doctoral dissertation).

[61.] Bozyiğit, F. (2019). Object oriented analysis and source code validation using natural language processing.

[62.] Baginski, J. (2018). Text analytics for conceptual modelling

APPENDIX

APPENDIX

The tables below show the algorithms used to extract potential classes, methods, attributes and relationships for system modeling language diagram generation.

ALGORITHM 1: ALGORITHM TO EXTRACT POTENTIAL CLASSES	
	***Input**: Document*
	***Output:** set of potential Classes*
1	*Create empty set for a potential class*
2	*#Classes from Noun Chunks*
3	***for** each sentence in the document:*
4	***for** each noun chunk in the sentence (chunk):*
5	***if** chunk doesn't contain digits or stop words:*
6	*Add its lemma of the chunk to the set of potential classes*
7	
8	*#Classes from Singular Nouns*
9	***for** each token in document:*
10	***if** token is noun or proper noun (POS):*
11	*Add its lemma of the token to the set of potential classes*
12	
13	*#Classes from Noun Pairs*
14	***for i** from 0 → 2nd to last token in document:*
15	***if** token **i** is noun or proper noun (POS):*
16	***if** token at **i+1** is also noun or proper noun (POS):*
17	*Combine lemmas of tokens **i** and **i+1** to form noun pairs*
18	*Add noun pairs to the set of potential classes*
19	
20	*#Classes from Subjects and Their Modifiers*
21	***for** each token in document:*
22	***if** token dependency: 'nominal' or 'passive subject':*
23	*Get dependency: 'compounds' to the left of token*
24	*Get dependency: 'amods' to the left of token*
25	*Combine lemmas of compounds and amods with lemma of token*
26	*Add the combined string to the set of potential classes*
27	
28	*Return the set of potential classes*

ALGORITHM 2: ALGORITHM TO EXTRACT POTENTIAL METHODS	
	***Input**: Document*
	***Output:** set of potential Methods*
1	*Create empty set for a potential Methods*
2	*Initialize Verb_Noun pair empty set*
3	*Initialize Last_Verb as None*
4	*#Methods from Verb Noun Pairs*
5	***for** each token in sentence:*
6	***if** token is Verb (POS):*
7	*Add token lemma to set of potential methods*
8	*Update Last_Verb to lemma token*
9	***elif** token is noun or proper noun and Last_Verb is not None*
10	*Add Last_Verb and lemma token pair to Verb_Noun pair*
11	*#Update Potential Methods with Verb_Noun pairs*
12	***for** each pair in Verb_Noun pair:*
13	*Combine pair into string*
14	*Add string to set of potential Methods*
15	
16	*Return the set of potential methods*

ALGORITHM 3: ALGORITHM TO EXTRACT POTENTIAL ATTRIBUTES	
	***Input**: Document*
	***Output:** set of potential Attributes*
1	*Create empty set for potential attributes*
2	*#Attributes from Noun Dependency Pairs*
3	***for** each token in document:*
4	***if** token is Noun (POS):*
5	*Get dependency: 'compound' left of token*
6	*Get dependency: 'amod' left of token*
7	***if** dependencies found:*
8	*Add both tokens to set of potential attributes*
9	
10	*#Attributes from Noun Child Pairs*
11	***for** each token in document:*
12	***if** token is Noun (POS):*
13	*Get child dependency: 'compound'*
14	*Get child dependency: 'amod'*
15	*Add child tokens to a list and include noun itself*
16	*Combine parts to form attribute phrase*
17	
18	*Return the set of potential attributes*

ALGORITHM 4: ALGORITHM TO EXTRACT POTENTIAL RELATIONSHIPS	
	***Input**: Document*
	***Output:** set of potential Relationships*
1	*Create empty set for potential relationships*
2	*#Relationships from Verbs and Nouns*
3	***for** each token in document:*
4	***if** token is Verb (POS):*
5	***for** each child of the verb:*
6	***if** child is Noun or Proper Noun (POS) and appears in potential classes:*
7	*Add (verb, child) in lemmas to relationships*
8	*Add (verb) to relationships*
9	
10	*#Relationships from Direct Objects*
11	***for** each token in document:*
12	***if** token is Verb (POS):*
13	***for** each child of the verb:*
14	***if** child dependency: 'direct object':*
15	*Add (verb, child) to relationships*
16	
17	*#Relationships from Determiners and Prepositions*
18	***for** each token in document:*
19	***if** token is Verb (POS):*
20	***for** each child of the verb:*
21	***if** child dependency: 'preposition' or 'determiner'*
22	*Add (verb, child) to relationships*
23	
24	*Return the set of potential relationships*

BIOGRAPHICAL SKETCH

Joshua Andre Ontiveros attended South Texas College and pursued his educational career as an undergraduate at the University of Texas Rio Grande Valley, where he graduated Magna Cum Laude with a Bachelor's degree in mechanical engineering in May 2022. During his undergraduate studies, Joshua helped design a three-stage refrigeration system to store COVID-19 vaccines for his senior design project, displaying his skills in advanced system designs. Joshua continued for his Master's degree at the University of Texas Rio Grande Valley in Fall 2022 joining the National Science Foundation CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS) as a research assistant. His research was funded by the National Science Foundation and focused on how AI can bridge the gap between human language to create system modeling diagrams for system engineers. Joshua completed his Master of science in Engineering degree in Mechanical Engineering in May 2024. Joshua Andre Ontiveros can be reached at Josh.ontiveros77@gmail.com.