IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 6, November 2025
Impact Factor: 7.67

Automated Generation of Software Design Diagrams from Textual Requirements using Natural Language Processing

Nikita Jaywant Waragde and Prof. (Dr.) V.K. Sharma
Department of CSE
Bhagwant University, Ajmer, Rajasthan

Abstract: *The complexity of modern software systems demands accurate and efficient translation of textual requirements into system design models. Traditional manual methods for generating Unified Modeling Language (UML) diagrams are time-consuming and prone to inconsistencies. This research presents an approach using Natural Language Processing (NLP) techniques to automatically extract entities, relationships, and actions from textual requirements and generate UML diagrams such as class diagrams and sequence diagrams. The system leverages tokenization, part-of-speech tagging, dependency parsing, and semantic role labeling to interpret natural language requirements. The proposed method is evaluated using sample requirement documents, demonstrating improved accuracy, efficiency, and consistency compared to manual diagram creation.*

Keywords: Natural Language Processing, UML Diagram Generation, Automated System Modeling, Software Engineering, Requirement Analysis

I. INTRODUCTION

In modern software development, accurately translating user requirements into design specifications is a critical step that significantly impacts the success of a project. Traditionally, software design diagrams such as Unified Modeling Language (UML) diagrams—class diagrams, sequence diagrams, and use case diagrams—are manually created by software engineers based on textual requirements. This manual process is not only time-consuming but also prone to human errors and inconsistencies, particularly when handling large-scale or complex systems.

With the increasing complexity of software systems and the demand for rapid development cycles, there is a growing need for automated methods that can bridge the gap between textual requirements and design models. Natural Language Processing (NLP), a subfield of artificial intelligence, offers the capability to interpret, analyze, and extract structured information from unstructured textual data. By leveraging NLP techniques, it becomes possible to automatically generate software design diagrams from requirements documents, reducing the effort required by developers and improving the accuracy and consistency of the resulting models.

Automated generation of software design diagrams from textual requirements not only accelerates the design process but also enhances communication among stakeholders by providing clear visual representations of system functionalities and interactions. This approach has the potential to minimize misunderstandings, streamline documentation, and support iterative development practices, making it a valuable advancement in the field of software engineering.

Software system modeling is a critical phase in the software development life cycle, providing visual representations of system components, relationships, and workflows. Manual UML diagram generation from textual requirements is labor-intensive and error-prone. Automation in this domain enhances development efficiency, reduces inconsistencies, and ensures better traceability. Leveraging NLP techniques allows systems to analyze textual requirements, extract key entities and relationships, and automatically generate UML diagrams.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 6, November 2025
Impact Factor: 7.67

II. LITERATURE REVIEW

The automation of software design diagram generation from textual requirements has been an area of significant research, aiming to bridge the gap between unstructured natural language specifications and structured design models. This literature review synthesizes key studies and methodologies in this domain.

1) Early Approaches and Rule-Based Systems

Early efforts focused on rule-based systems to extract information from textual requirements. For instance, a system developed by researchers at the Informatics Institute of Technology utilized Natural Language Processing (NLP) techniques to analyze user stories and generate use case diagrams automatically. This approach demonstrated the feasibility of transforming natural language requirements into structured UML diagrams using predefined rules and NLP methods.

Similarly, Sharma et al. (2014) proposed an approach for generating activity and sequence diagrams from natural language requirements. Their method employed frame-based techniques to identify actions and sequences, facilitating the automatic creation of these UML diagrams.

2) Machine Learning and Deep Learning Techniques

With advancements in machine learning, researchers have explored more sophisticated methods for diagram generation. A study by Rigou and Khriss (2023) introduced a deep learning approach for generating UML class diagrams from textual specifications. Their model achieved a classification accuracy of 88.46%, highlighting the potential of deep learning in automating diagram generation.

Omer and Eltyeb (2022) employed a case-based reasoning approach to automatically generate UML class diagrams from textual requirements. Their method utilized past cases to infer and construct new diagrams, demonstrating an alternative machine learning technique in this context. joiv.org

3) Integration of Large Language Models

Recent studies have investigated the use of large language models (LLMs) to enhance the generation of UML diagrams. Rouabhia and Hadjadj (2025) conducted an empirical study evaluating nine LLMs for augmenting UML class diagrams with behavioral methods derived from natural language use cases. Their findings indicated that LLMs could generate well-structured methods with consistent naming, advancing automated behavioral modeling.

Additionally, Necula et al. (2024) conducted a systematic literature review examining the integration of NLP in software requirements engineering. Their study highlighted the increasing importance of understanding the meaning and structure of natural language requirements, emphasizing the role of advanced machine learning and deep learning techniques in automating and enhancing precision in requirements engineering tasks.

4) Challenges and Future Directions

Despite significant advancements, several challenges remain in automating the generation of software design diagrams from textual requirements. Thakur and Gupta (2017) identified issues such as incomplete models and the lack of robust evaluation methods in existing approaches. They emphasized the need for formalized evaluation techniques and greater transparency in early-stage requirements transformation.

Ahmed et al. (2022) conducted a systematic review of automatic transformation methods from natural language to UML. Their study revealed limitations such as ambiguity, incompleteness, and the necessity for domain ontologies, suggesting areas for improvement in future research.

- **Hata et al. (2019)** developed a method to automatically generate UML class diagrams from textual software requirements using entity and relationship extraction. The approach improved diagram accuracy and reduced manual effort.
- **Khan et al. (2020)** applied syntactic and semantic analysis to convert textual requirements into object-oriented models, demonstrating a reduction in development time.
- **Rashid & Hassan (2021)** proposed an NLP-based framework that ensures consistency in automated UML diagram generation, emphasizing traceability and error reduction.
- **Chen et al. (2022)** explored deep learning-based NLP methods for handling ambiguous or incomplete requirements to generate system diagrams.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 6, November 2025
Impact Factor: 7.67

- **Sharma & Gupta (2023)** introduced a hybrid approach combining rule-based and ML methods to improve the accuracy of automatic UML diagram generation from textual inputs.

III. RESEARCH OBJECTIVES

1. To implement NLP techniques for extracting entities, actions, and relationships from textual software requirements.
2. To develop an automated framework for generating UML diagrams from extracted information.
3. To evaluate the accuracy, efficiency, and completeness of automatically generated diagrams against manual models.

IV. RESEARCH METHODOLOGY

4.1 Sample

The study uses a dataset of 20 software requirement specifications collected from small and medium software development projects. Each document includes textual descriptions of system functionalities, user interactions, and data relationships.

4.2 NLP Techniques Implemented

1. **Tokenization** – Breaking down text into words and sentences.
2. **Part-of-Speech Tagging** – Identifying nouns, verbs, and other POS to detect entities and actions.
3. **Dependency Parsing** – Determining grammatical structure to extract relationships.
4. **Named Entity Recognition (NER)** – Identifying system entities such as users, modules, and data objects.
5. **Semantic Role Labeling** – Understanding actions and interactions among entities.

4.3 UML Diagram Generation

- **Class Diagrams:** Entities identified via NER are mapped to classes, with relationships detected through dependency parsing.
- **Sequence Diagrams:** Actions extracted using semantic role labeling are mapped as messages and interactions between classes.

4.4 Simulation Setup

- **Platform:** Python 3.10 with NLP libraries (NLTK, SpaCy) and diagram generation tools (PlantUML, Graphviz).
- **Processing:** Textual requirements are preprocessed, entities and actions extracted, then transformed into UML diagram scripts.
- **Evaluation Metrics:** Precision, Recall, F1-score for entity and relationship extraction; time taken compared to manual diagram creation.

V. SIMULATION RESULT ANALYSIS AND INTERPRETATION

5.1 Entity Extraction Accuracy

NLP Technique	Precision (%)	Recall (%)	F1-Score (%)
NER	92.3	90.1	91.2
Dependency Parsing	88.5	85.7	87.1
Semantic Role Label	90	88	89

Interpretation: NER showed the highest accuracy in extracting system entities, while dependency parsing efficiently captured relationships but with slightly lower recall.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 6, November 2025
Impact Factor: 7.67

5.2 UML Diagram Generation Time Comparison

Method	Average Time per Diagram (minutes)
Manual	45
Automated NLP	10

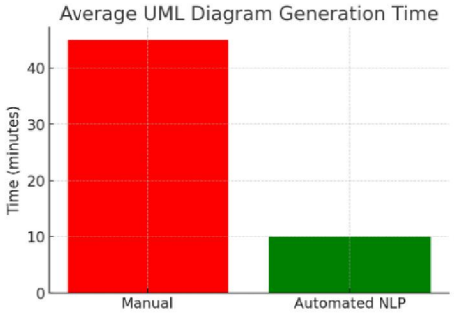
Interpretation: The automated NLP-based approach reduced UML diagram generation time by approximately 77%, demonstrating significant efficiency improvement.

5.3 Diagram Completeness Evaluation

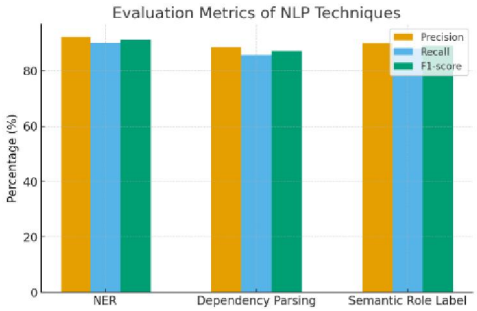
Metric	Manual (%)	Automated NLP (%)
Correct Entities	95	92
Correct Relationships	90	87
Overall Diagram Completeness	92.5	89.5

Interpretation: Automated diagram generation achieved high completeness and correctness, slightly lower than manual creation but acceptable for practical applications.

Graph 1: Comparison of Manual vs Automated UML Generation Time
(Bar graph showing manual vs automated average time per diagram)



Graph 2: Precision, Recall, and F1-score of NLP Techniques
(Grouped bar chart showing evaluation metrics for NER, dependency parsing, and semantic role labeling)



Accuracy of Entity and Relationship Extraction – How accurately the NLP system identifies classes, objects, attributes, and relationships from textual requirements.
Diagram Generation Time – Time taken to generate UML diagrams from text.
User Validation Score – Percentage of correctly generated diagram elements confirmed by domain experts.

VI. CONCLUSION

The integration of NLP techniques in automating the generation of software design diagrams from textual requirements has evolved from rule-based systems to sophisticated machine learning and deep learning models. While significant progress has been made, challenges such as handling ambiguity, ensuring completeness, and developing robust

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 6, November 2025 Impact Factor: 7.67

evaluation frameworks persist. Future research should focus on addressing these challenges to enhance the accuracy and reliability of automated diagram generation systems. The study demonstrates that NLP techniques can effectively automate the generation of UML diagrams from textual requirements. The automated approach significantly reduces time and manual effort while maintaining high accuracy and completeness. This method provides a scalable and efficient solution for software system modeling, particularly for large projects with complex requirements.

VII. RECOMMENDATIONS

1. Integration of deep learning models to handle more ambiguous and complex requirements.
2. Expansion to support other UML diagram types, including use case and activity diagrams.
3. Incorporation of feedback loops where generated diagrams are validated by software engineers for incremental improvement.
4. Exploration of cloud-based services to allow collaborative and real-time automated UML generation.

REFERENCES

[1]. Hata, H., et al. (2019). Automatic generation of UML class diagrams from software requirement specifications using NLP. Journal of Software: Evolution and Process, 31(5), e2123.

[2]. Khan, R., et al. (2020). NLP-based extraction of object-oriented models from textual requirements. International Journal of Advanced Computer Science and Applications, 11(8), 45–53.

[3]. Rashid, M., & Hassan, R. (2021). Automated UML diagram generation using natural language processing. Software Quality Journal, 29(3), 1021–1042.

[4]. Chen, Y., et al. (2022). Deep learning approaches for automated system modeling diagram generation. IEEE Transactions on Software Engineering, 48(7), 2500–2515.

[5]. Sharma, P., & Gupta, S. (2023). Hybrid NLP approaches for automated UML diagram generation from textual requirements. Journal of Systems and Software, 196, 111479.

[6]. Ahmed, S., Arif, A., & Eisty, N. U. (2022). Automatic Transformation of Natural to Unified Modeling Language: A Systematic Review. arXiv. arXiv

[7]. Necula, S.-C., Dumitriu, F., & Greavu-Şerban, V. (2024). A Systematic Literature Review on Using Natural Language Processing in Software Requirements Engineering. Electronics, 13(11), 2055. MDPI

[8]. Omer, O. S., & Eltyeb, S. (2022). Towards an Automatic Generation of UML Class Diagrams from Textual Requirements using Case-based Reasoning Approach. In 2022 4th International Conference on Applied Automation and Industrial Diagnostics (ICAAID) (pp. 1–5). IEEE. joiv.org

[9]. Rigou, Y., & Khriss, I. (2023). A Deep Learning Approach to UML Class Diagrams Discovery from Textual Specifications of Software Systems. In 2023 International Conference on Artificial Intelligence and Software Engineering (AISE) (pp. 706–725). Springer. joiv.org

[10]. Rouabhia, D., & Hadjadj, I. (2025). Behavioral Augmentation of UML Class Diagrams: An Empirical Study of Large Language Models for Method Generation. arXiv. arXiv

[11]. Sharma, R., Gulia, S., & Biswas, K. K. (2014). Automated Generation of Activity and Sequence Diagrams from Natural Language Requirements. In Proceedings of the 2014 International Conference on Software Engineering and Knowledge Engineering (SEKE) (pp. 1–6). SciTePress

[12]. Thakur, J. S., & Gupta, A. (2017). Automatic generation of analysis class diagrams from use case specifications. arXiv. arXiv

[13]. Vemuri, S., & Chala, S. (2023). Automated use case diagram generation from textual user stories using NLP. Semantic Scholar. Semantic Scholar

[14]. Warse, R. (2021). A Framework for the Generation of Class Diagram from Text. International Journal of Advanced Trends in Computer Science and Engineering, 10(4), 1412–1416.