Automated Use Case Diagram Generator: Transforming Textual Descriptions into Visual Representations using a Large Language Model

Maxmillan Giyane
Department of Computer Science
Midlands State University
Gweru, Zimbabwe
email: giyanem@staff.msu.ac.zw

Dzinaishe Mpini
Department of Computer Science
Midlands State University
Gweru, Zimbabwe
email: mpinid@staff.msu.ac.zw

Abstract— **Software Architects often use Use Case diagrams, a type of Unified Modelling Language (UML) behavior diagram, to capture user needs and system functionalities. These diagrams aid in project estimation by identifying system requirements and reducing ambiguity. Creating them manually is a time-consuming task prone to errors. This research aims to automate Use Case diagram generation from text using the Generative Pretrained Transformer 3.5 (GPT-3.5) Turbo model. The developed tool uses a Natural Language Processing (NLP) technique to extract actors, use cases, and associations from descriptions, and convert these elements into UML-compliant diagrams. It also includes an interactive interface for Use Case diagram refinement. The system processes user input text to identify relevant elements, visualizes them using jCanvas, and allows real-time user interaction for refinement. Testing showed an 89.33% accuracy in element identification but highlighted areas for improvement like handling edge cases and optimizing performance. This research demonstrates the potential of NLP and visualization tools to improve Use Case diagram generation efficiency and accuracy, with future work focusing on enhancing usability and functionality.**

Keywords- Use Case Diagram; Large Language Model; GPT 3.5 Turbo; Natural Language Processing.

I. INTRODUCTION

The software Architect task relies on different methods to capture user needs. One method widely used is Use Case diagrams. According to Fauzan et al. [1], Use Case diagrams are a specific type of behavior diagram in the Unified Modelling Language (UML) which are primarily meant to help visualize a system's functionalities. UML itself is a prominent notation system commonly employed in software architecture [2]. Use Case diagrams capture system behavior from the user's perspective, detailing interactions, and system boundaries [3]. They essentially define what the software should do [3]. Beyond functionality, Use Case diagrams play a valuable role in project estimation as they highlight system requirements which are in turn used to estimate development effort [4][5][6]. Furthermore, they help reduce ambiguity within requirement specifications [7].

The core elements of a Use Case diagram are actors, use cases, and their associations. Actors are external entities (individuals or groups) that interact with the system. Use cases represent the interactions themselves, specifying how actors achieve goals within the system. These elements are connected by associations, signifying the communication between actors and use cases [8].

Creating well-structured Use Case diagrams requires adherence to specific conventions due to the complexity of placement rules [2]. Unlike typical graph layouts, Use Case diagrams demand specialized methods to ensure clarity, particularly as diagram size increases. The guidelines in Use Cases diagram generation encompass naming conventions for actors, systems, and use cases themselves. They advocate for simplicity and clarity, emphasizing the use of nouns and verbs to clearly define elements within the diagram [9].

These conventions make drawing of a Use Case diagram difficult. Filipova and Nikiforova [2] alludes manual layout of Use Case diagrams is a time-consuming activity; and one can fail to produce effective diagrams. With the way technology is advancing nowadays, researchers were compelled to create more efficient tools for drawing Use Case diagrams that follow the UML notation.

This research is aimed at developing a tool for automated Use Case diagrams generation from text that utilizes Large Language Models (LLMs). The specific objectives of the research are:

1. To develop a Natural Language Processing (NLP) technique which utilizes the GPT 3.5 Turbo LLM to extract relevant information which includes actors, use cases, and associations from textual system descriptions.
2. To develop an engine that can automatically convert the extracted elements (actors, use cases and associations) from the NLP analysis into a coherent and accurate Use Case diagram compliant with UML standards.
3. To design an interface that allows users to interact with the generated Use Case diagram and refine its actors, associations, and use cases manually.

The remainder of this paper is organized as follows: In Section 2, we review related work on automated software documentation and LLM applications. Section 3 details our methodology, including the prompt engineering framework. Section 4 presents a case study validating the approach use cases. Section 5 discusses results, limitations, and comparisons with traditional methods. Finally, Section 6 concludes with future directions, including integration with generative media tools.

II. RELATED WORK

Use case diagrams were first proposed by Ivar Jacobson in 1986 as part of his work on object-oriented software engineering [8]. These diagrams have since become a fundamental tool in software development, aiding in the visualization of system functionality from a user perspective. In drawing tools, one can use the manual approach, electronic drag and drop tools, and automated tools.

This research noted a lack of recent scholarly articles focusing on manual tools for Use Case diagram creation. Electronic drag and drop tools research have also not been clearly documented but there exist several tools for Use Case diagram generation. These include Lucid Chart, Visual Paradigm, Smart Draw, DrawIO, Miro, Microsoft Visio and Wondershare Edraw Max. Table 1 shows the top found tools and the analysis done on them.

In the realm of automating the generation of Use Case diagrams from textual descriptions, several significant studies have been conducted. Elallaoui et al. [7] conducted pioneering research aimed at transforming user stories into UML Use Case diagrams automatically. By leveraging NLP techniques, their approach achieved impressive accuracy scores ranging from 87% to 98%. This evaluation was based on a comparison of the outputs automatically generated by the plugin against manual modelling of each user story. This demonstrated the potential of NLP in interpreting and converting textual requirements into structured diagrams. Similarly, Nasiri et al. [17] presented a comprehensive framework for the automatic generation of various UML diagrams, including class, Use Case, and package diagrams. Their approach involved processing user stories written in natural language (English) using the Stanford Core NLP engine. By incorporating artificial intelligence through Prolog rules and ontology, they enhanced their previous methodologies, resulting in improved outcomes. Despite reporting that the results of the approach have been validated by several case studies, the methodology for assessing the approach was not documented and the metric values of results were not specified. While both studies showcased promising results, they also had limitations. Notably, the carried out researches lack implementation. However, a Google search revealed an implemented automated Use Case diagram generation tool called Diagramming AI [18]. The underlying technology behind its working is not publicly available and at the time of analysis the tool did not adhere to the UML Use Case diagram standard, limiting its utility for standard-compliant projects.

Recent advances in prompt engineering have become pivotal for optimizing LLM outputs in software engineering tasks. Sahoo et al. [19] conducted a systematic survey of prompt engineering methods for LLMs, categorizing techniques, such as zero-shot, few-shot, and role-based prompting. Their work highlights how tailored prompts improve accuracy in structured outputs, a finding directly relevant to our Use Case extraction process.

TABLE I. USE CASE DIAGRAMS GENERATION TOOLS

Source	Tool	Access Channel	Cost
[10]	Lucid Chart	Web based application, and embedded in Google platforms	USD7.95-9.95/month. Free trial available
[11]	Visual Paradigm	Web based application and desktop applications	USD4.00-15.00/month. Free trial available
[12]	Smart Draw	Web based application	USD9.95
[13]	DrawIO	Web based application, desktop application, and embedded in Google platforms	USD34.00/20 users. Free trial available
[14]	Miro	Web based application	USD8.00-16.00/month. Free trial available
[15]	Microsoft Visio	Desktop application	USD44.15 for the software license
[16]	Wonder Share Edraw Max	Web based application	USD5.99-79.99/month

Wang et al. [18] explored the application of LLMs in generating UML diagrams. Use Case diagrams were part of the UML diagrams under study. 45 undergraduate students explored the platform. The research demonstrated 100% correctness of LLMs in identifying users, relationships and functional requirements from a given scenario. However, the research only encompassed identifying users, relationships and functional requirements and did not create the Use Case diagram from this information.

Additionally, Carrazan [20] provides critical insights into LLM applications for automating software requirements, particularly Use Case diagrams and narratives. The study demonstrates that when guided by carefully engineered prompts, Chat Generative Pre-trained Transformer (ChatGPT) can effectively generate accurate requirements documentation while significantly reducing development time. Carrazan's methodology [20] emphasizes a structured input-process-output framework where tailored prompts serve as inputs to produce validated UML artifacts - an approach that directly informs our work's prompt design strategy (Section III.B). Notably, the dissertation [20] confirms that LLM-generated requirements can achieve sufficient quality for stakeholder communication and effort estimation, though it cautions that human validation remains essential. These findings complement existing literature [7][19] while providing empirical evidence of LLMs' potential to streamline early-phase software documentation.

III. METHODOLOGY

A. System Architecture

The proposed tool went through a streamlined process designed to facilitate the creation and refinement of Use Case diagrams from textual descriptions. Initially, users provide a textual description of their desired system via a

web page interface. This input is then sent to the backend, where the GPT-3.5 Turbo model processes the text to identify and extract relevant actors, use cases, and associations. The extracted information is subsequently transferred to the frontend, where the jCanvas engine generates an initial Use Case diagram based on the provided data. Users can interact with the diagram through a user-friendly interface, allowing them to modify actors, system names, use cases, and associations. These modifications are reflected in real time on the Use Case diagram, thanks to the dynamic capabilities of the jCanvas engine. Once users are satisfied with the refined diagram, they have the option to save or export the final version for their documentation or further use. The architecture of this tool ensures a seamless and interactive experience from the initial text input to the final output, enabling users to efficiently create and refine Use Case diagrams. Figure 1 shows the architectural diagram of the proposed system.

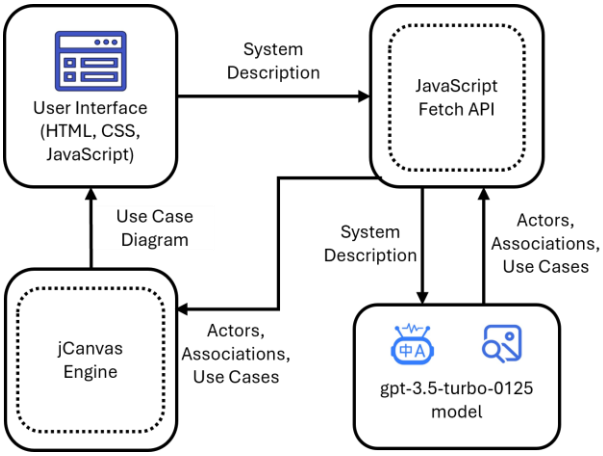

Figure 1. Architecture diagram of the proposed system.

B. GPT3.5 Turbo Model Working Mechanism

GPT-3.5 Turbo was chosen because of its Instruct Architecture. According to Yeow et al. [19] this architecture comprises multiple layers, each containing a self-attention mechanism and a feed-forward neural network. The self-attention mechanism enables the model to weigh the importance of different parts of the input when making predictions, enhancing its contextual understanding. The feed-forward neural network then makes the final predictions, allowing GPT-3.5 Turbo to generate coherent and contextually relevant text across various applications.

Bandara et al. [20] outlines GPT-3.5, released by OpenAI in 2020, is the foundational language model for the original ChatGPT and represents significant advancements in NLP and generation. With 175 billion parameters, it is one of the largest language models, demonstrating improved language understanding, enhanced text generation, and the ability to produce human-like text across various domains. GPT-3.5's architecture allows ChatGPT to engage in natural, context-aware dialogues, leveraging its extensive pre-training to draw on a vast knowledgebase. However, GPT-3.5 has limitations, such as struggles with logical reasoning, potential biases from its training data, and a restricted context window of 2,048 tokens. Understanding these strengths and limitations is essential for setting realistic expectations when using ChatGPT and similar Artificial Intelligence (AI) applications built on GPT-3.5. In generating the Use Case diagrams, jCanvas was used. jCanvas is a jQuery plugin that makes it easy to work with the Hypertext Markup Language 5 (HTML5) canvas element [21]. It provides a convenient Application Programming Interface (API) for drawing shapes, text, and images, as well as handling animations and user interactions. The plugin integrates seamlessly with jQuery, enabling efficient manipulation of canvas elements and real-time updates, making it an excellent choice for creating and modifying Use Case diagrams [22].

C. Interface Design

The interface design for the web page should be a one-page, user-friendly and dynamic visualization of system descriptions through Use Case diagrams. Upon loading, users encounter a central text input box where they can enter detailed descriptions of the system they intend to diagram. Adjacent to this input area is a "Generate Diagram" button, signaling the action to transform the entered text into a visual representation. Once activated, the system processes the input using GPT-3.5 Turbo via the OpenAI Application Programming Interface (API) and displays the resulting Use Case diagram on a canvas. This canvas initially presents elements, such as system boundaries, actors, use cases, and their associations based on the processed text.

Each element within the diagram becomes interactive and editable directly on the canvas, enabling users to click, drag, and modify elements effortlessly. This interactive capability extends to renaming actors or use cases, adjusting connections, and repositioning elements to suit specific requirements. Real-time updates ensure that any changes made by the user are immediately reflected in the displayed diagram, maintaining continuity and allowing for iterative refinement. Options for saving or exporting the finalized diagram, typically in formats like PNG or PDF, provide users with the means to preserve their work or share it as needed. The interface design emphasizes clarity, intuitive usability, and responsiveness across different devices, aiming to facilitate seamless interaction and effective visualization of system structures from textual descriptions.

IV. Tool Description

The tool was designed as a comprehensive web application that utilized HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and the Bootstrap framework to create an intuitive user interface. jQuery was employed to enhance user interaction, ensuring smooth and responsive handling of dynamic elements within the interface. A Representational State Transfer Application Programming Interface (RESTful API) endpoint was developed in PHP: Hypertext Preprocessor (PHP) to facilitate seamless communication with the 'gpt-3.5-turbo-0125' model from OpenAI.

The core functionality of the tool was driven by jCanvas, a powerful jQuery plugin that enabled the creation and modification of Use Case diagrams directly within the web page. Users interacted with a straightforward interface where they input system descriptions and, upon triggering the "Generate Diagram" function, observed a visual representation of their system structure. This design emphasized usability and real-time responsiveness, allowing users to refine and customize their diagrams effortlessly. The tool's integration of modern web technologies ensured an efficient and engaging experience for users who sought to visually conceptualize system architectures from textual descriptions.

V. Results and Discussion

In testing the developed system, a comprehensive suite of tests was conducted to ensure functionality, accuracy, performance, usability, and integration across its core objectives. A group of 3 Computer Science students and 2 Computer Science lecturers who were not involved in the development of the system conducted the tests. Each user created their own user story and evaluated the performance of the system with those user stories.

A. Large Language Model Performance in Extraction of Actors, Use Cases and Associations

Initially, the accuracy of the NLP technique was rigorously evaluated through test cases that assessed the extraction of actors, use cases, and associations from diverse textual use case descriptions. This included edge case scenarios to gauge robustness. Performance testing focused on measuring processing speeds and scalability under varying system descriptions. In the test, reviewers analyzed the output actors, Use Cases and associations to gauge the accuracy of the tool. Additionally, the time taken to produce an output was recorded.

The testers' scores revealed a promising average accuracy of 89.33%, indicating the tool successfully identifies elements from the descriptions. However, there were some missed elements, like deposit/withdrawal use cases in one instance and the bank teller actor itself for the first test case. These highlight areas for improvement, particularly in handling operations which have not been mentioned. The loading time for testers ranged from 5 to 7 seconds, averaging at 6.2 seconds. While acceptable, further optimization can enhance user experience. Additionally, testers 3 and 5 noted overly long system names generated by the tool. This suggests the system might be assigning generic, lengthy descriptions. Implementing logic to generate concise and descriptive names would be beneficial. The NLP technique shows promise with its accuracy. However, improvements are needed to handle edge cases, optimize loading times, and generate better quality names for actors and use cases. This will further enhance the tool's effectiveness and user experience.

B. Use Case Diagram Generation

The engine's capability to convert extracted elements into UML-compliant Use Case diagrams was verified through validation tests against UML standards and guidelines, ensuring diagrams met syntax and semantic requirements. Customization features were tested to validate user-defined preferences and styles, ensuring flexibility in diagram presentation. Integration tests ensured seamless interoperability with external systems, assessing data consistency and compatibility.

Largely, the test results show that the system has the capability to convert extracted elements into UML-compliant Use Case diagrams, but there are some areas for improvement, such as refining extracted elements to avoid cluttered diagrams and ensuring that generated names fit within the designated space. Only Tester 2 found that the tool generated a poor diagram due to the identification of too many use cases. This suggests that the system might need improvement in refining the extracted elements to ensure a clear and concise Use Case diagram. Tester 3 identified an issue where the system name spanned outside the boundary of the diagram. This indicates that the name generation process might need to consider the available space within the diagram to ensure all elements are well presented.

C. Interactive User Interface for Refining Generated Diagrams

User Interface (UI) testing involved usability assessments with potential end-users to gauge ease of use and navigation. Feedback mechanisms were tested to capture user inputs on diagram quality and interface improvements. Compatibility tests were conducted across different devices to ensure consistent performance and responsiveness. Error handling was scrutinized through various error scenarios to assess how the system managed and communicated errors effectively to users.

While testers commended the tool's ease of use and functionality for adding, deleting, or modifying elements, they highlighted the need for improved visual clarity. This suggests that while the core functionalities are present, the user interface might benefit from enhancements that ensure a clearer visual representation of the Use Case diagram during the editing process.

VI. Conclusion and Future Work

This research focused on the development of a tool capable of extracting a Use Case diagram elements from a given textual system description using a large language model. The tool should further draw the Use Case diagram using jCanvas and allow a user to manually refine the generated Use Case diagram. The testing approach utilized validated each aspect of the tool objectives. Notably, there was no comparable researches available for direct comparison, as existing literature either lacked publicly available implementations which follow the UML Use Case diagrams standard or did not employ an automated diagram generation approach like the one proposed in this research.

The system is poised for deployment in real-world environments where efficient Use Case diagram generation is paramount. The research recommends scalability to verify the system's capability to manage increasing volumes of use case descriptions without performance degradation, ensuring

robustness under varying workloads. Additionally, it is recommended to provide comprehensive user training and support material to facilitate smooth and effective utilization of the system.

While this study demonstrates the efficacy of GPT-3.5 Turbo in automating Use Case diagram generation, the reliance on a single LLM poses a limitation. Recent advancements in code-specific LLMs, such as Codex, StarCoder or fine-tuned variants, such as Llama-3 with UML datasets may yield higher accuracy in extracting structural UML elements. Future work should also include a comparative analysis of multiple LLMs, evaluating their performance in parsing textual descriptions and adhering to UML standards. This expansion will help identify optimal models for specific tasks, such as handling “include” and “extend” relationships or complex system boundaries, further improving the tool’s robustness. Beyond testing other LLMs, future research could explore AI-generative media tools, such as Stable Diffusion or DALL-E to automatically enhance diagram aesthetics and layout, enabling direct conversion of textual descriptions into polished UML figures while maintaining compliance with standards through hybrid human-AI validation frameworks. The tool was also tested by only 5 individuals from the same department at a university. This presents potential bias on the effectiveness of the tool and future work must be evaluated by many participants from varying backgrounds.

Additionally, the developed tool outputs an image file of the generated Use Case diagram without the XML code for that can be used in other diagramming tools. Future work could focus on producing both the Use Case diagram image as well as standard XML code for a Use Code which can be integrated in other languages.

Furthermore, there is an opportunity for comparative studies with other GPT variants or large language models to identify and integrate the most efficient model for improving system performance and accuracy. These enhancements and comparisons will contribute to advancing the capabilities and effectiveness of the system in generating and manipulating Use Case diagrams.

REFERENCES

[1] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "A Different Approach on Automated Use Case Diagram Semantic Assessment," International Journal of Intelligent Engineering and Systems, vol. 14, no. 1, 2021, pp. 496-505, doi - 10.22266/ijies2021.0228.46.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, pp. 68–73, 1892,

[2] O. Filipova and O. Nikiforova, "Definition of the Criteria for Layout of the UML Use Case Diagrams," Applied Computer Systems, vol. 24, no. 1, 2019, pp. 75-81, doi - 10.2478/acss-2019-0010.

[3] F. Mokhati and M. Badri, "Generating Maude Specifications From UML Use Case Diagrams," Journal of Object Technology, vol. 8, no. 2, 2009, pp. 119-136.

[4] P. Jayadi, R. S. Dewi, and K. Sussolaikah, "Activity-based function point complexity of Use Case diagrams for software effort estimation," Journal of Soft Computing Exploration, vol. 5, no. 1, 2024, pp. 1-8, doi - 10.52465/joscex.v5i1.252.

[5] A. B. Nassif, L. F. Capretz, and H. Danny, "A Regression Model with Mamdani Fuzzy Inference System for Early Software Effort Estimation Based on Use Case Diagrams", PhD dissertation, Graduate Program in Electrical and Computer Engineering, The University of Western Ontario, 2012.

[6] P. Sahoo and J. R. Mohanty, "Early Test Effort Prediction using UML Diagrams," Indonesian Journal of Electrical Engineering and Computer Science , vol. 5, no. 1, 2017, pp. 220-228.

[7] M. Elallaoui, K. Nafil, and R. Touahni, "Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques," in The 8th International Conference on Ambient Systems, Networks and Technlogies (ANT 2018), 2018, pp. 42-49.

[8] A. Y. Aleryani, "Comparative Study between Data Flow Diagram and Use Case Diagram," International Journal of Scientific and Research Publications, vol. 6, no. 3, 2016.

[9] P. Danenas, T. Skersys, and R. Butleris, "Natural language processing enhanced extraction of SBVR business vocabularies and business rules from UML Use Case diagrams," Data and Knowledge Engineering, 2020.

[10] Lucid Chart, "Draw Chart," 2024. [Online]. Available: https://lucid.app/lucidchart/. [Accessed June 2025].

[11] Visual Paradigm, "Visual Paradigm," 2024. [Online]. Available: https://online.visual-paradigm.com. [Accessed June 2025].

[12] Smart Draw, "Use Case Diagram," 2024. [Online]. Available: https://www.smartdraw.com/use-case-diagram/. [Accessed June 2025].

[13] Drawio, "Draw Use Case Diagram," 2024. [Online]. Available: https://drawio-app.com/ . [Accessed June 2025].

[14] Miro, "Use Case Diagram," Miro, 2024. [Online]. Available: https://miro.com/templates/use-case-diagram/. [Accessed June 2025].

[15] Microsoft , "Create a UML Use Case Diagram," Microsoft, 2024. [Online]. Available: https://support.microsoft.com/en-us/office/create-a-uml-use-case-diagram-92cc948d-fc74-466c-9457-e82d62ee1298 . [Accessed June 2025].

[16] EdrawMax, "Use Case Diagram," Edraw, 2024. [Online]. Available: https://www.edrawmax.com/online/en/ . [Accessed June 2025].

[17] S. Nasiri, Y. Rhazali, M. Lahmer and A. Adadi, "From User Stories to UML Diagrams Driven by Ontological and Production Model," International Journal of Advanced Computer Science and Applications, vol. 12, no. 6, 2021, doi - 10.14569/IJACSA.2021.0120637.

[18] B. Wang, C. Wang, P. Liang, B. Li, and C. Zeng, "How LLMs Aid in UML Modeling: An Exploratory Study with Novice Analysts," 2024 IEEE International Conference on Software Services Engineering (SSE), Shenzhen, China, 2024, pp. 249-257, doi: 10.1109/SSE62657.2024.00046.

[19] L. Naimi, E. Bouziane, A. Jakimi, R. Saadane, and A Chehri, "Automating Software Documentation: Employing LLMs for Precise Use Case Description", Procedia Computer Science, vol. 246, no 1, 2024, pp. 1346-1354, doi 10.1016/j.procs.2024.09.568.

[20] P. F. V. Carrazan, Large Language Models Capabilities for Software Requirements Automation, Ph.D. dissertation, Dept. Comput. Eng., Politecnico di Torino, Torino, Italy, 2023.