AI-Assisted System Design: Improving Sequence Diagrams Through Use Case Scenarios

Harsh Sarvaiya
hsarvaiya.work@gmail.com
Department of Engineering
Thompson Rivers University
Kamloops, BC, Canada

Dr. Munima Jahan
munimajahan@yahoo.com
Department of Engineering
Thompson Rivers University
Kamloops, BC, Canada

***Abstract*—This paper explores two AI-driven approaches for transforming user stories (USs) into sequence diagrams (SDs). Pipeline A directly converts USs into SDs, while Pipeline B introduces an intermediate step of generating use case scenarios. Both pipelines leverage OpenAI's GPT-4 model to automate these tasks. Ten user stories were processed through both pipelines and evaluated for accuracy, clarity, efficiency, and scalability. Pipeline B's use of intermediate use case scenarios significantly improved the quality and detail of SDs while maintaining logical consistency and reducing ambiguity. These findings demonstrate the potential of AI-assisted tools to streamline software development, enhance collaboration, and make technical workflows accessible to non-technical stakeholders. While promising, challenges such as computational overhead and reliance on language models require further research. This study lays the groundwork for integrating AI into model-driven engineering, advancing efficiency and collaboration in software design.**

***Index Terms*—Large Language Models, Sequence Diagram, User Story, Use Case Scenario, ChatGPT**

I. Introduction

Agile software development often uses user stories to convey initial requirements effectively [5] [6]. These stories, written in natural language from the end user's perspective, describe software features informally to communicate their value to customers. However, poorly defined user stories can create challenges, particularly when translating them into technical artifacts [9].

User stories, typically formatted as "As a ¡actor¿, I want ¡goal¿ so that ¡reason¿," foster clarity but often lack the precision needed for technical implementation [5]. To address this, visual models like sequence diagrams (SDs) are commonly used to illustrate system interactions and identify potential design flaws early [8]. However, generating accurate SDs from user stories is complex, and poorly constructed diagrams can lead to miscommunication, design flaws, and inefficiencies [1] [9]. This issue is particularly critical for complex systems, where precise interaction modeling is essential [10].

Current methods for generating SDs from user stories are either manual or automated [5] [6]. Manual approaches are time-consuming and inconsistent, while automated methods, including NLP and AI techniques like ChatGPT, improve efficiency but often miss crucial details [1] [7]. Despite these advancements, the potential of using an intermediate layer, such as use case scenarios, to enhance diagram quality remains underexplored in existing literature.

This paper addresses the gap in how user stories are translated into sequence diagrams by introducing an intermediate step that involves creating use case scenarios [5] [6]. This structured approach aims to improve the clarity, detail, and consistency of the resulting diagrams compared to directly generating them from user stories. The hypothesis is that this intermediate step helps preserve the essential details and context of user stories, ensuring that the diagrams are both more comprehensive and easier to interpret [9]. In addition, the paper explores how the integration of AI tools, such as ChatGPT, can further enhance the generation process by combining the strengths of manual insights and automated scalability [1] [7]. By addressing these gaps, the proposed solution offers a more effective approach to sequence diagram generation, bridging the gap between informal user stories and formal design artifacts.

Key contributions of this paper include:

A comparative analysis of two workflows:

- Generation SD from user stories with the help of ChatGPT
- Generating UCS from the US and then creating SD from the UCS with the help of ChatGPT

Empirical evaluation of results, highlighting the strengths and weaknesses of each method.

The remainder of this paper is organized as follows. Section II reviews related work, focusing on existing methodologies and their limitations. Section III outlines the methodology of this study. Section IV presents the results and analysis. The Section V discusses the evaluation process and the outcome. Finally, Section VI discusses the findings and suggests future directions for research in this area.

II. Literature Review

Automating the generation of sequence diagrams (SDs) from textual requirements has gained significant attention in software engineering, with approaches ranging from rule-based transformations to the use of large language models (LLMs) [1] [5] [6]. Early methods, such as those by Elallaoui et al. [5], employed rule-based approaches to detect actors, messages, and objects in user stories, converting them into XMI files for UML sequence diagram generation. While

effective for structured inputs, these methods struggle with ambiguous or non-standard user stories, requiring manual refinement. Nasiri et al. [8] extended rule-based approaches with ontologies and Prolog rules to enhance the accuracy of UML diagram generation, showing how semantic enrichment can improve results.

More recent studies have explored the potential of LLMs like ChatGPT to directly generate SDs from text. Jahan et al. [6] compared LLM-driven methods to traditional rule-based approaches, finding that while LLMs capture more context, they may introduce irrelevant details that require expert oversight. Abualhaija et al. [1] evaluated ChatGPT's performance with real-world requirements, noting that while the generated diagrams generally adhered to UML syntax, they were often incomplete or inaccurate, highlighting the need for iterative refinement and human-in-the-loop strategies. Similarly, Wang et al. [10] found that LLMs helped identify basic UML elements but struggled with complex relationships, emphasizing the benefits of a hybrid approach with human validation.

Despite these advances, most research focuses on directly generating SDs from user stories without considering intermediate steps like use case scenario modeling. While rule-based and LLM-based methods show promise, they often lack a structured approach to evaluate the clarity and consistency of intermediate artifacts. Incorporating use case scenarios before generating SDs could improve diagram accuracy and completeness, offering a more reliable two-step process. This gap in the literature suggests that combining LLMs with intermediate modeling could enhance diagram quality, ensuring both contextual richness and structural accuracy [9].

III. Methodology

This study proposes two distinct approaches for transforming user stories (USs) into sequence diagrams (SDs). The first approach, referred to as Pipeline A, directly converts USs into SDs. The second approach, Pipeline B, introduces an intermediate step of generating use case scenarios before creating the SDs. Both approaches were designed and implemented using the same foundational codebase, with the only difference being the inclusion of the use case scenario generation step in Pipeline B.

The input requirements for this study were textual USs that adhered to a standard format: "As a [Actor], I want to [Action], so that [Benefit]." This format ensured that each story described a single action or a compound sentence specifying multiple tasks using the conjunction 'and.'. 50 USs meeting these criteria were selected from a well-recognized dataset [9], to ensure a diverse and representative dataset.

Figure 1 illustrates the end-to-end workflow for the two pipelines described above. After loading the user stories (USs) from a CSV file, the process branches into two paths. In Pipeline A, each US is passed directly to the GPT-4 model, producing a PlantUML (PUML) script that is then rendered as a sequence diagram. In Pipeline B, by contrast, the USs are first converted into detailed use case scenarios; these scenarios are then transformed into PUML sequence diagrams.

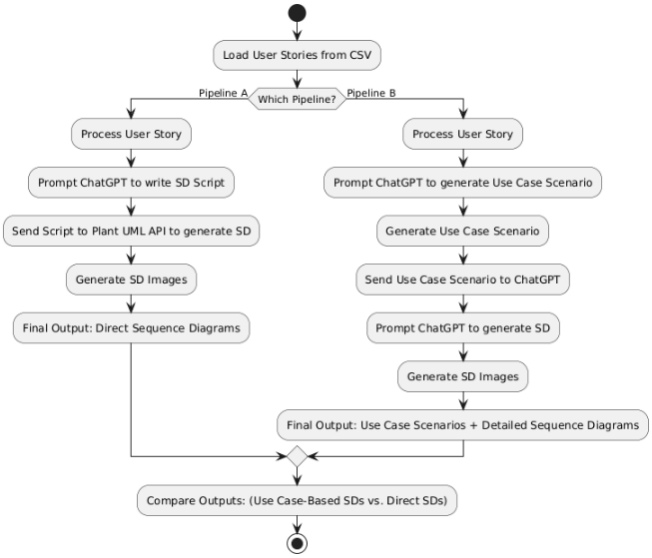


Fig. 1: Workflow Diagram of Methodology

Finally, the resulting diagrams—either direct (Pipeline A) or use-case-based (Pipeline B)—are compared to assess diagram accuracy, completeness, and clarity. This diagrammatic overview highlights the primary difference between the two approaches: Pipeline B includes an additional layer of detail via use case scenarios, whereas Pipeline A provides a more direct route to sequence diagram generation.

Pipeline A processes USs in a straightforward manner. A Python-based program interacts with OpenAI's GPT-4 model to convert the USs directly into PUML sequence diagram scripts. Each US is input as a prompt to the GPT-4 model, requesting a sequence diagram that includes all necessary actors (e.g., "Library Member," "Library System") and visualizes key interactions, such as searching for a book or viewing details. The program ensures that the generated PUML scripts are syntactically correct, starting with "@startuml" and ending with "@enduml." The scripts are then saved and optionally included in a Word document for further analysis. Pipeline A provides a direct and efficient method for generating SDs, minimizing preprocessing and intermediate steps.

Pipeline B, however, follows a direct process where the USs are passed to OpenAI's GPT-4 model to generate detailed use case scenarios. The predefined prompt in Pipeline B differs from Pipeline A by explicitly requesting a breakdown of primary actors, actions, and system responses, aiming to capture a more structured intermediate representation before the generation of the sequence diagram scripts. The program uses a predefined prompt to instruct GPT-4 to create scenarios that outline the primary actors, actions, and system responses. The generated use case scenarios are then converted into PUML sequence diagram scripts, which visualize the main flow of events.

The program generates detailed use case scenarios for each US by interacting with OpenAI's GPT-4 model. These scenarios outline the primary actors, actions, and system

responses. The generated use case scenarios are then converted into PUML sequence diagram scripts, using predefined prompts to instruct GPT-4 to output syntactically correct diagrams that visualize the main flow of events. The scripts are saved and optionally processed using the PUML API to generate visual SDs.

The results of the two pipelines were validated through manual inspection, they were reviewed to ensure they accurately represented the intended processes. This validation confirmed the value of incorporating the use case scenario generation step in Pipeline B, as it led to more detailed and accurate SDs while maintaining clarity.

IV. Experimental Results

A. Experimental Setup

The experimental framework was developed in Python 3 and executed on Google Colab, leveraging its cloud-based runtime environment for rapid prototyping and reproducibility [1] [6]. OpenAI is used to interact with OpenAI's API for generating use case scenarios and UML sequence diagrams. We also integrated PUML for generating UML diagrams via a PUML server.

These specific versions were chosen to ensure compatibility with the codebase and the features provided by each library. For instance, `openai==0.28` guarantees access to the latest functionalities of the GPT-4 model, which is critical for producing detailed technical outputs [6].

B. Dataset Selection and Rationale

To validate the approach, two publicly available benchmark datasets from Jahan [7] were used. These datasets [2], [9] have been widely utilized in previous studies [3], [4] and provide diverse user stories for empirical analysis. These datasets were chosen for their comprehensiveness and domain-specific focus, as it contains user stories derived from real-world scenarios in library and restaurant management systems. These contexts offer a balanced mix of structured requirements and varied complexity, making the dataset an ideal candidate for assessing the performance of our sequence diagram generation pipelines [6] [8].

To ensure high-quality inputs, an initial filtering process was applied. Specifically, ChatGPT was employed to review the dataset and select the 50 most representative user stories based on criteria such as clarity, completeness, and relevance to system design [1] [9]. The resulting subset was then compiled into a CSV file, which served as the input for both Pipeline A and Pipeline B in our experimental setup. This targeted selection not only streamlined the evaluation process but also ensured that the stories used were of sufficient quality to generate meaningful UML artifacts.

C. Prompt Configuration and Model Usage

The pipelines leverage OpenAI's GPT-4 model to transform textual user stories into technical artifacts [6]. Two distinct prompts were used in the process:

1) Use Case Scenario Generation (Pipeline B): In Pipeline B, the intermediate step involves generating a detailed use case scenario. The exact prompt used is as follows:

"You are an expert in UML use case design. For the following user story, generate a complete UML use case scenario with detailed sections: - Use Case Name: A short descriptive name for the use case. - Actors: List all actors involved. - Preconditions: Any conditions that must be met before the use case starts. - Triggers: The events or conditions that start the use case. - Main Flow of Events: A step-by-step description of the primary flow. - Alternate Flow of Events: Any alternate or exception flows. - Postconditions: The final state after the use case completes.

User Story: "{user_story}"

Ensure the output is complete and does not contain placeholders such as "### Use Case Name"."

This prompt instructs GPT-4 to not only identify key actors and actions but also to provide a structured breakdown that can later be transformed into a sequence diagram.

2) Sequence Diagram Generation: Subsequently, a second prompt is used to convert either the original user story (in Pipeline A) or the generated use case scenario (in Pipeline B) into a PUML script for a sequence diagram:

"You are an expert in software engineering and UML diagramming. Based on the following UML user story, generate a PUML script for a sequence diagram that visualizes the main flow of events. Only answer with the script.

Use Case Scenario: "{user_story}"

The PUML script should include all relevant actors and system components involved in the main flow. Ensure the script is syntactically correct and can be rendered without errors. "

Both prompts have been crafted with technical precision to ensure that the GPT-4 model generates outputs that are both syntactically valid and rich in domain-specific details.

D. Results

The sample sequence diagrams generated from two different approaches are shown in the Figure 2.

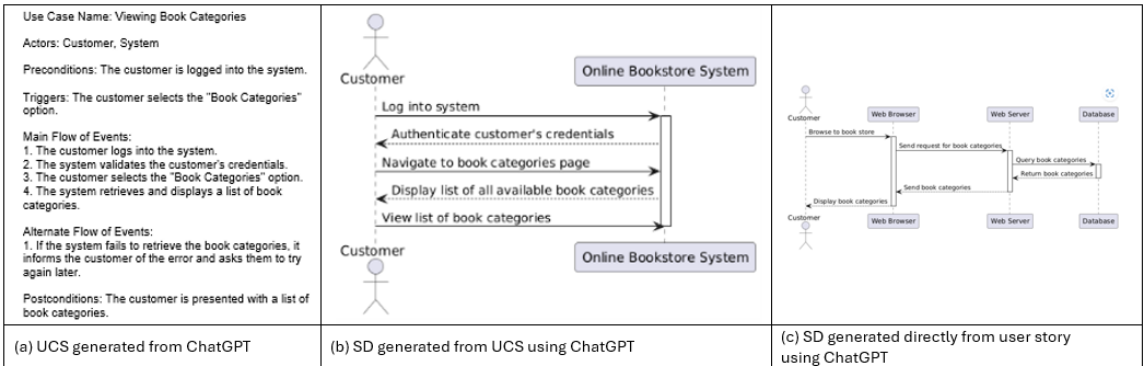
V. Preliminary Evaluation

1) Evaluator Setup and Scoring Methodology: The evaluators assessed the generated SDs based on three key criteria: accuracy, relevance, and unnecessary complexity. Each SD was rated on a 1 to 10 Likert scale, where 10 represented the highest score. To ensure consistency in evaluation, a pre-assessment meeting was held to standardize the scoring methodology among the three evaluators, including one of the authors of this study.

The evaluation criteria were defined as follows:

Accuracy: Assessed whether the SD correctly represented essential system elements, including processes, data stores, external entities, and message transfers.

Evaluated the correctness of objects, interactions, and message order in the diagram. Rated on a scale of 1 to 10. Relevance: Measured how well the SD aligned with the system's intended functionality and accurately represented the semantic content of the user story.



(a) UCS generated from ChatGPT | (b) SD generated from UCS using ChatGPT | (c) SD generated directly from user story using ChatGPT

Fig. 2: SD generated from different approaches

Assessed whether the sequence diagram faithfully depicted interactions between components. Rated on a scale of 1 to 10. Unnecessary Complexity (Irrelevant Information): Evaluated the extent of extraneous or irrelevant details that increased diagram complexity.

Assessed whether additional information hindered clarity or complicated the SD. Rated on a scale of 1 to 10.

2) Evaluation Process: Each evaluator received 20 SDs generated from 10 user stories using two different pipelines. To analyze inter-rater reliability, five common user stories were included in each evaluator's set, allowing for a correlation analysis between evaluators' ratings.

To generate SDs, user stories were automatically fed to ChatGPT using Python Code, following a structured prompt as mentioned in the previous section. The responses, formatted in PUML, were then processed using a PUML web server to create the final diagrams.

All script files, SD images, and evaluation materials were documented and made publicly available in a dedicated repository (https://github.com/Harsh-Sarvaiya/AI-Assisted-System-Design-Paper) for transparency and reproducibility.

A. Performance Results

Figure 2 presents two sample Sequence Diagrams (SDs) generated from a single User Story (US) using different pipelines. Table 1 summarizes the median accuracy, relevance, and complexity scores assigned by each evaluator for ten SDs produced by each pipeline. The results indicate that both pipelines achieve high scores in generating SDs from user stories, highlighting GPT-4's potential in interpreting and transforming textual requirements into structured diagrams.

One notable observation is that the prompt does not require explicit syntax rules, demonstrating the LLM's internal understanding of standard SD conventions. While the generated SDs may not achieve perfect completeness and correctness, integrating human oversight and refinement can enhance their applicability.

Furthermore, the interactive chat-based nature of LLMs presents opportunities for iterative improvement in SD generation. Future research could explore how iterative prompting enables LLMs to adapt and refine SDs more effectively. Additionally, understanding the threshold of human acceptance—the point at which users are willing to refine AI-generated diagrams within editable diagramming tools—is an intriguing direction for further empirical investigation.

Criteria	US to SD (Pipeline A)	US to UCS to SD (Pipeline B)
Accuracy	9.0	9.4
Relevance	9.0	9.6
Complexity	0.46	1

TABLE I: The Median Scores of different criterion for two different pipeline

Evaluator Pair	Pearson	Spearman	Kendall
1 and 2	0.99	0.57	0.35
1 and 3	0.99	0.72	0.55
2 and 3	0.99	0.92	0.82
Average	0.99	0.73	0.57

TABLE II: Correlation Coefficients between Evaluator

B. Evaluator Agreement and Correlation Analysis

Table 2 presents the correlation coefficients among evaluator pairs, measuring the level of agreement in their assessments to ensure reliability. To standardize the evaluation, scores were converted into rankings for SDs generated using two different approaches, even when identical scores were assigned to both.

The results indicate statistically significant positive correlations across all evaluator pairs. Spearman's correlation coefficients range from moderate to strong (average: 0.73), while Kendall's tau values, though slightly lower, also demonstrate significant positive relationships (average: 0.57). The Pearson's correlation coefficient is 0.99 for each pair of evaluators.

These findings suggest consistent agreement among evaluators, where higher ratings by one evaluator correspond to higher ratings by others. Additionally, the statistical significance ($p < 0.001$) reinforces the reliability of these correlations, confirming that the observed concordance is not due to random chance but reflects a genuine alignment in evaluator scoring.

C. Discussion

The findings from this study demonstrate the potential impact of incorporating an intermediate step of use case scenario generation into the process of transforming user stories (USs) into sequence diagrams (SDs). The results underscore the importance of structured intermediate representations, as evidenced by the improved clarity and accuracy of the SDs generated by Pipeline B. These enhancements have significant implications for both academic research and industry practices.

One of the key contributions of this work is the demonstration of how automated pipelines, particularly those leveraging advanced AI models like OpenAI's GPT-4, can streamline traditionally manual and time-intensive tasks in software engineering. Pipeline B's ability to generate use case scenarios as an intermediate step not only improves the quality of SDs but also provides additional insights into the requirements elicitation process. By offering a more structured approach, this methodology could reduce misinterpretation of user requirements, a common pain point in software development projects. From Table I, we observe that Pipeline B achieves a higher score in terms of accuracy and relevance. However, it has a lower complexity associated with the generated sequence diagrams (SDs). User stories represent requirements from the user's perspective but do not provide details on how to complete a function or the specific actions involved in a task. In contrast, generating use case scenarios (UCS) from user stories offers a more detailed sequence of activities needed to achieve the user's goal, thereby adding more context to the SDs.

D. Threat to the Validity

This study faces two main threats to validity: the subjective evaluation of Sequence Diagrams (SDs) generated by ChatGPT and the selection of user stories. Subjective evaluations may not fully capture constructs like accuracy and relevance due to evaluator variability, oversight, and biases, undermining construct validity. Additionally, the selection of a limited set of user stories may introduce bias, potentially skewing results toward more favorable outcomes and threatening external validity.

To address the first threat, the author held a meeting with other two evaluators to set clear expectations for the Likert scale. For the second threat, a set of predefined criteria for user story selection was established. However, there is still room to improve the validity of future research.

VI. Conclusion and Future Work

This paper presents two approaches that leverage the power of large language models (LLMs), specifically ChatGPT, to generate sequence diagrams (SDs) from textual requirements presented in the form of user stories. The results demonstrate that LLMs have significant potential in automating the process of model generation from textual requirements. In particular, incorporating a middleware that converts intermediary information, such as use case scenarios, can further enhance the quality of the generated diagrams. These findings are particularly relevant for organizations working on large-scale and complex systems where accuracy in requirements analysis is critical. Moreover, the integration of AI-driven tools into software engineering workflows could democratize access to technical diagramming for non-technical stakeholders.

Looking ahead, there is significant potential to integrate these pipelines into broader model-driven engineering workflows. Combining these approaches with automated testing, continuous integration, and deployment pipelines could create highly efficient and intelligent software development environments. Future work could also explore leveraging other large language models (LLMs) beyond ChatGPT to enhance the accuracy and diversity of generated models. Additionally, extending AI-driven automation to generate not only sequence diagrams (SDs) but also other UML artifacts, such as class diagrams and activity diagrams, would further expand the applicability of these methodologies.

Acknowledgment

This research is partially supported by the Internal Research Fund of Thompson Rivers University. The authors are also thankful to the reviewers, Dr. Zeinab Teimoori and Dr. Shaista Jabeen, for their contributions in reviewing the results and providing valuable feedback.

References

[1] Saja Abualhaija, Chirag Arora, and Alessio Ferrari. Model generation with llms: from requirements to uml sequence diagrams. In *Proceedings of the IEEE*, 2024.

[2] Fabiano Dalpiaz. Requirements data sets (user stories). *Mendeley Data, v1*, 2018.

[3] Fabiano Dalpiaz, Patrizia Gieske, and Arnon Sturm. On deriving conceptual models from user requirements: An empirical study. *Information and Software Technology*, 131:106484, 03 2021.

[4] Fabiano Dalpiaz, Ivor Schalk, Sjaak Brinkkemper, Fatma Aydemir, and Garm Lucassen. Detecting terminological ambiguity in user stories: Tool and experimentation. *Information and Software Technology*, 110, 12 2018.

[5] Meryem Elallaoui, Khalid Nafil, and Raja Touahni. Automatic generation of uml sequence diagrams from user stories in scrum process. In *2015 10th international conference on intelligent systems: theories and applications (SITA)*, pages 1–6. IEEE, 2015.

[6] Munima Jahan, Zahra Shakeri Hossein Abad, and Behrouz Far. Generating sequence diagram from natural language requirements. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 39–48. IEEE, 2021.

[7] Munima Jahan, Mohammad Mahdi Hassan, Reza Golpayegani, Golshid Ranjbaran, Chanchal Roy, Banani Roy, and Kevin Schneider. Automated derivation of uml sequence diagrams from user stories: Unleashing the power of generative ai vs. a rule-based approach. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, pages 138–148, 2024.

[8] 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 (IJACSA)*, 12(6), 2021.

[9] Fatma Başak Aydemir Salih Göktuğ Köse. A user story dataset for library and restaurant management systems. *Zenodo*, 2023.

[10] Bo Wang, Bin Li, Chen Wang, Peng Liang, and Chang Zeng. How llms aid in uml modeling: An exploratory study with novice analysts. In *Proceedings of [Conference Name, if available]*, 2024.