

# ECHO: An Approach to Enhance Use Case Quality Exploiting Large Language Models

Gabriele De Vito,<sup>1</sup> Fabio Palomba,<sup>1</sup> Carmine Gravino,<sup>1</sup> Sergio Di Martino,<sup>2</sup> Filomena Ferrucci<sup>1</sup>

<sup>1</sup>Software Engineering (SeSa) Lab—Department of Computer Science, University of Salerno (Italy)

Department of Electrical, Engineering and Information Technologies—University of Naples "Federico II" (Italy)  
gabriele.devito@gmail.com, fpalomba@unisa.it, gravino@unisa.it, sergio.dimartino@unina.it, fferrucci@unisa.it

**Abstract**—UML use cases are commonly used in software engineering to specify the functional requirements of a system since they are an effective tool for interacting with stakeholders thanks to the use of natural languages. However, producing high-quality use cases can be challenging due to the lack of precise guidelines and suitable tools. This can lead to problems, e.g. inaccuracy and incompleteness, in the derived software artifacts and the final product. Recent advancements in Natural Language Processing and Large Language Models (LLMs) can provide the premises for developing tools supporting activities based on natural languages. In this paper, we propose ECHO, a novel approach for supporting software engineers in enhancing the quality of UML use cases using LLMs. Our approach consists of a co-prompt engineering approach and an iterative and interactive process with the LLM to improve the quality of use cases, based on practitioners' feedback. To prove the feasibility of the proposal, we instantiated the approach using ChatGPT and performed a controlled experiment to assess its effectiveness by involving seven software engineering professionals. Three were part of the experimental group and used ECHO to improve the quality of the use cases. Three others were the control group and enhanced the quality of use cases manually. Finally, the last participant acted as an oracle, blind w.r.t. the groups, and evaluated the quality of the enhanced use cases, both qualitatively by means of a questionnaire, and quantitatively, by means of the Use Case Points metric. Results show that ECHO can effectively support software engineers to improve use cases' quality thanks to the prompts suitably designed to interact with ChatGPT.

**Index Terms**—UML Use Cases; Large Language Models; Prompt Engineering; Size/Effort estimation.

## I. INTRODUCTION

UML use cases are widely used in software engineering to express the functional requirements of a system [15]. The main advantage of use cases is that they are written in natural language, and thus can be easily understood also by stakeholders, to verify if the described functional requirements correspond to their needs [14]. Nevertheless, this key advantage can also result in their main drawback, as the absence of precise binding rules can lead to poorly-written use cases. Let us observe that the quality of use cases is critical for the quality of all the other software artifacts deriving from them, such as more detailed UML models, including sequence diagrams and analysis object models [6]–[12], and then of the final product. Accurate use cases are also critical for deriving better software functional size measures, such as Use Case Points (UCP) [19] employed for effort/cost estimation, benchmarking, productivity assessment, etc. [1]. Producing high-quality use cases

is time-consuming due also to the absence of adequate tool support.

Given the importance of the quality of use cases, many attempts have been made to apply Natural Language Processing (NLP) techniques to reduce the risks of ambiguity, incorrectness, and incompleteness [2]–[5]. However, as highlighted by Zhao et al. [16], these approaches have found limited adoptions by practitioners, mostly due to technological limitations of past NLP approaches. On the other hand, in the last few years, the introduction of deep neural networks to NLP has significantly changed the scenario, giving rise to Large Language Models (LLMs). In particular, LLMs like GPT-3 [23], PaLM [24], Megatron-Turing NLG [25], ChatGPT [13] have billions of parameters and have been trained on enormous amounts of text data, demonstrating to be effective in various applications, including text summarization, machine translation, and question-answering systems.

To the best of our knowledge, LLMs have not yet been applied for supporting requirements engineers in tasks involving natural language and in particular for improving the quality of use cases. To verify whether LLMs can be effective also in this domain, we propose ECHO (Effective Communication with llms to enHance the quality Of use cases), a novel approach based on LLMs for supporting software engineers in enhancing the quality of UML use cases. ECHO consists of two main elements: (1) a novel co-prompt engineering approach, which lets the LLM learn how to react to users' inputs and produce the expected output. This technique allows us to design a suitable prompt for the LLM to obtain the qualitative analysis of use cases under our identified criteria (prompt P1 in the sequel); and (2) an iterative process to enhance the quality of use cases. This process allows software engineers to interact with the LLM which, starting from prompt P1, can give accurate suggestions according to the expected quality criteria, consider the practitioners' feedback on the suggested improvements (prompt P2), and formulate the enhanced version of the use cases based on the agreed suggestions (prompt P3).

To validate our approach we have performed a study aiming to address the following research question:

**RQ:** *Can the ECHO approach support software engineers to improve the quality of UML use cases?*

We instantiate ECHO using ChatGPT Web interface. The study involved seven software engineering professionals. One

of them acted as the oracle, and the other six were randomly assigned to the experimental group (i.e., they used ECHO to improve use cases) or to the control group (i.e., they manually improved use cases). Both groups were given the same use cases and asked to complete questionnaires. The oracle assessed the quality of the enhanced use cases at the final stage, unaware of the group that improved them.

The outcome of this research can contribute to designing effective prompts to interact with language models and to develop approaches for enhancing the quality of software requirements and estimating software development efforts, leading to improved software development project outcomes.

**Structure of the paper.** Section II discusses the background and the motivations leading to our approach. In Section III we report on the design of our approach, while Section IV overviews the research method employed to assess it. Section V discusses the results achieved when measuring the performance of our approach, with Section VI that presents the potential limitations of our work and how we addressed them. Finally, Section VII concludes the paper and outlines our future research agenda.

## II. BACKGROUND AND MOTIVATION

UML use cases are a software engineering technique that specifies system requirements in natural language. However, creating high-quality use cases is challenging also since no precise binding rules are available. To be considered high-quality, use cases must meet various criteria, including correctness, completeness, and clarity. Correctness requires the use case to describe the system’s behavior accurately (i.e., the steps must be in the correct order, and should be atomic). Completeness demands that all possible scenarios, including alternative and error scenarios, be covered. Clarity mandates that all stakeholders can easily understand the use case without ambiguity. In addition, the use case must follow a logical sequence of events, provide the same level of detail for each step, and use consistent terminology. Many attempts have been undertaken to examine requirements specifications using NLP to reduce the risk of quality issues before employing them in subsequent stages of the software development lifecycle [2]–[5]. Nonetheless, the approaches employed consider traditional analysis techniques, general-purpose tools, and generic language lexicons, which limit their adoption [16].

Over the last years, there have been significant advances in the NLP domain thanks to the introduction of deep neural networks and LLMs. LLMs exhibit learning and generalization abilities with only a few examples (few-shot learning), where a prompt can guide LLMs to generate the correct answers for desired tasks [21], [22]. Several studies have shown the effectiveness of LLMs in supporting software development tasks [17], [18] (i.e., code generation, bug fixing, and architectural analysis). Therefore, it is reasonable to think that LLMs can help in other software engineering tasks, such as supporting practitioners in enhancing the quality of UML use cases. However, the effectiveness of the results produced by LLM is closely linked to the quality of the prompts the user

gives. Indeed, although various versions of the same sentence can have the same meaning for humans, the same is not true for LLMs, which can then provide different answers to essentially identical questions formulated in a slightly different way. This behavior might happen because LLMs have learned that various sentences are employed in different situations, contexts, and goals.

This means that prompts can heavily impact the subsequent interactions and output generated by an LLM through specific instructions for the conversation based on a predetermined set of rules because “prompting is programming, representing the query language for large language models” [20].

To better understand the importance of prompt engineering, see the example in Fig. 1, where we asked ChatGPT to produce a specific response to improve use case quality. Without providing ChatGPT with the right prompt, the tool generates an incomplete and undetailed response, which does not consider preconditions and does not comprise details for all the criteria and aspects we need.

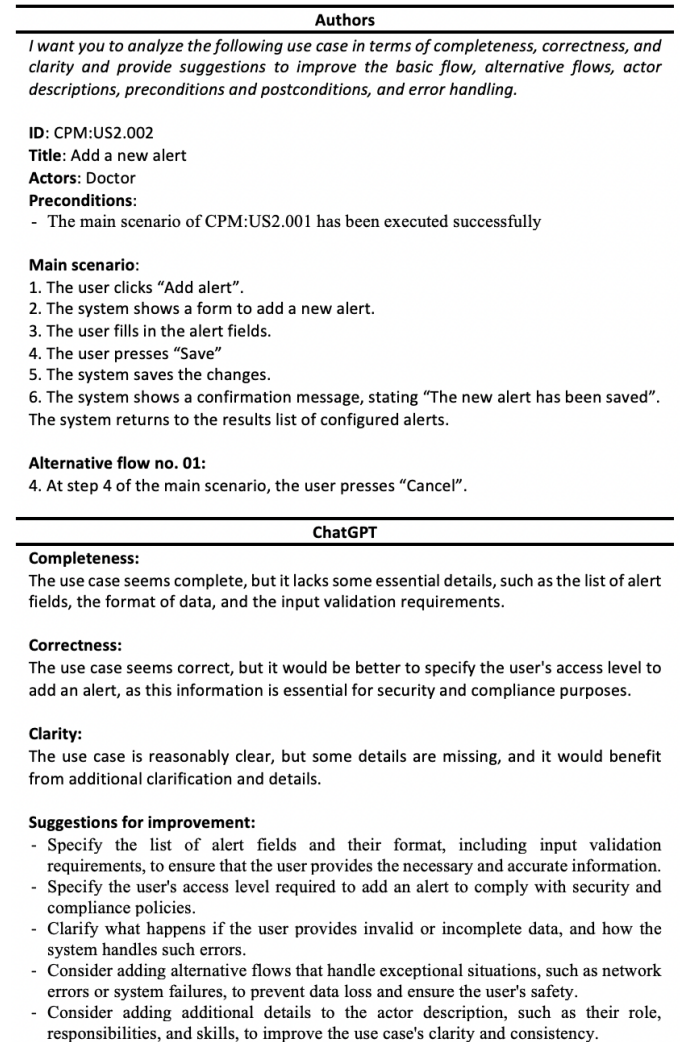


Fig. 1. Unoptimized instructions provided to ChatGPT.

As we can note, the default "behavior" of ChatGPT is inadequate to achieve the desired result. Therefore, we need to find the correct prompts and rules to instruct an LLM for improving use cases quality, which motivated our approach, as discussed in detail in the next section.

### III. THE ECHO APPROACH

This section illustrates ECHO which comprises two primary phases: 1) Co-Prompting Method Engineering, 2) Feedback-Based process to enhance the quality of use cases.

#### A. Co-Prompting Method Engineering

Prompt engineering is a creative fine-tuning approach with multiple promptings until we get the desired responses from the model. Some prompt engineering patterns exist. Among them, the "Persona pattern," gives the LLM a persona or role to play when generating output [26]. This pattern intends to give the LLM a "role" that helps it to select what types of output to generate and what details to focus on. Based on this pattern, we set a role for the LLM, i.e., starting our prompt with the words "Act as a ..." (e.g., "Act as a software engineer...") and then add as many details as possible. The answer will be more pertinent to the role indicated.

However, this approach still makes a detailed analysis of the quality of use cases hard to obtain. This issue concerns the high number of quality criteria and aspects the LMM must consider simultaneously, acting in that particular role. Therefore, it is necessary to proceed with subsequent feedback to allow the LLM to formulate a more precise analysis, such as: "You did not consider the preconditions. Moreover, I need details for each criterion regarding the following aspects: basic flow, alternative flows, actor descriptions, preconditions, postconditions, and error handling."

This process can continue with constant feedback and tuning until we obtain the desired result.

Taking into account the above considerations, the question is: having the desired analysis, how can we design the right prompt to instruct the LLM to produce exactly that output? Aiming to find an answer, we decided to use ChatGPT as a prompt engineering companion, asking it how to provide input to enable it to generate the desired results, as follows: "Act as a prompt engineer. Write the prompt I should have written so that you understood how to provide an analysis similar to the last one." This led ChatGPT to suggest the correct instructions to get that particular response. The final version of the prompt (P1) to send to ChatGPT in order to provide a preliminary use case analysis is shown in Fig. 2, while the ChatGPT's response to P1 is shown in Fig. 3.

As we can see, the response produced by the tool is now complete and detailed because it comprises all the criteria and, for each criterion, the aspects we requested to improve the use case quality.

#### B. Iterative process to enhance the quality of Use Cases

The second stage of our approach is related to the design of an effective process that, starting with the initial qualitative

**G** Act as a software engineer.  
I need you to analyze the following use case in terms of completeness, correctness, consistency, and clarity. For each criterion, I need you to provide suggestions for improvement on the following aspects: actors, preconditions, postconditions, basic flow, alternative flows, and error handling. Please make sure to provide specific details for each aspect and criterion, as well as any suggested changes to the use case to improve atomicity.

**Completeness:**

- Actors: please evaluate whether the actors are reported.
- Preconditions: please evaluate whether the preconditions are reported.
- Postconditions: please evaluate whether the postconditions are reported.
- Basic Flow: please evaluate whether the basic flow is complete and suggest any improvements that could be made. Evaluate if input data validation is reported in the basic flow.
- Alternative Flows: please evaluate whether the alternative flows are not missing, and suggest any improvements that could be made.
- Error Handling: please evaluate whether the error handling is reported and complete, and suggest any improvements that could be made.

**Correctness:**

- Actors: please evaluate whether the actor is correctly identified.
- Preconditions: please evaluate whether the preconditions are accurate and correctly identify the success criteria for the main scenario of the use case.
- Postconditions: please evaluate whether the postconditions accurately describe the state of the system after the use case has been executed.
- Basic Flow: please evaluate whether the basic flow is correctly ordered and atomic and suggest any improvements that could be made to improve atomicity. Precisely suggest how to split sentences so that the steps are atomic. The user and the system must execute only one action in each step.
- Alternative Flows: please evaluate whether the alternative flows are correctly ordered and atomic.
- Error Handling: please evaluate whether the error handling is accurate, atomic, and covers all potential errors that may occur.

**Clarity:**

- Please evaluate the use case based on whether it is easily understandable for all stakeholders, including developers and non-developers. Provide comments on whether the use case includes extraneous information or jargon that could impede understanding. In addition, evaluate whether the use case is based on logical sequence and terminology consistency. For example, the use case should not use "user" or "actor" interchangeably in different steps, and the inconsistency should be resolved in favor of one term.

Fig. 2. Prompt P1: Starting Use Case Analysis.

analysis of a use case using the prompt P1, (i) allows us to tune the analysis, considering the practitioners' feedback iteratively, and (ii) produces the enhanced version of the use case.

Feedback is critical because we want ChatGPT able to remove suggestions from the qualitative analysis or simply need more details on a specific criterion. For example, let us suppose that ChatGPT provides the following suggestion to improve the use case:

*"The use of examples or screenshots could also enhance clarity."*

While the suggestion is generally valid, adding examples of values for input (or output) fields does not necessarily make the use case clearer. In this case, we can ask ChatGPT to ignore the suggestion in its analysis using the prompt (P2) shown in Fig. 4, where the "SUGGESTIONS" placeholder represents the list of analysis suggestions we want the tool to ignore.

On the other hand, to ask ChatGPT to provide the improved version of a use case, the prompt (P3) to employ is shown in Fig. 5. Indeed, we can ask ChatGPT to reanalyze the improved version of the analyzed use case, and the process can continue until we are satisfied with the quality of the use case version provided by the tool.

With these building blocks to instruct the LLM, we can design the overall iterative ECHO process shown in Fig. 6.

Considering the limitations of the ChatGPT conversational context (maximum number of tokens 4096, including input

#### Completeness:

##### Actors:

The use case reports only one actor, the Doctor. It is not clear whether other actors, such as a Nurse or an Administrator, may have a role in this scenario. Therefore, it is recommended to add a list of actors who may interact with the system during this use case.

##### Preconditions:

The use case mentions only one precondition, that the main scenario of CPM:US2.001 has been executed successfully. It is not clear what CPM:US2.001 is, and there is no information on how to execute it successfully. It is recommended to add a brief description of CPM:US2.001 and how to execute it successfully.

##### Postconditions:

The use case mentions only one postcondition, that the system returns to the results list of configured alerts. It is not clear whether any other changes occur in the system state after the use case is executed. It is recommended to add a list of all changes that occur in the system state after the use case is executed.

##### Basic Flow:

The basic flow seems to be complete. However, there is no mention of input data validation. It is recommended to add a step to validate input data before saving it in the system.

##### Alternative Flows:

There is only one alternative flow mentioned in the use case, which is when the user presses "Cancel" at step 4. There may be other alternative flows that are not reported. It is recommended to review the use case and identify all possible alternative flows.

##### Error Handling:

The use case does not mention any error handling. It is recommended to add a section to the use case that identifies potential errors and how to handle them.

##### Correctness:

##### Actors:

Fig. 3. ChatGPT analysis of a use case in response to P1

**G** Rewrite your analysis, ignoring the following suggestions. <<SUGGESTIONS>>

Fig. 4. Prompt P2: Feedback to ChatGPT

and output tokens), i.e., the content of the dialogue between the user and the chatbot, it is necessary to repeat prompt P1 (and following P2 and P3) for each new use case to analyze.

The transcripts of the conversations with ChatGPT for our prompt engineering approach are available as supplemental material on GitHub at <https://github.com/gadevito/FGG>.

## IV. EXPERIMENTAL METHOD

This section describes the design of our experimentation while the results are presented in the next section.

### A. Experiment definition and context

To answer our research question and assess the proposed approach, we conducted an experiment involving 6 professionals from an Italian IT company and a consultant. The company was founded in 2002 and has over 50 employees. It has adopted the Agile development process, using UML regularly. The involved professionals had the same experience level. All participants have worked with UML use cases for 5-7 years. They had the same distribution of roles, with all participants being business analysts or product owners. Most of them

**G** Apply your suggestions and provide the improved version of the use case.

Fig. 5. Prompt P3: Asking for the Enhanced Use Case version

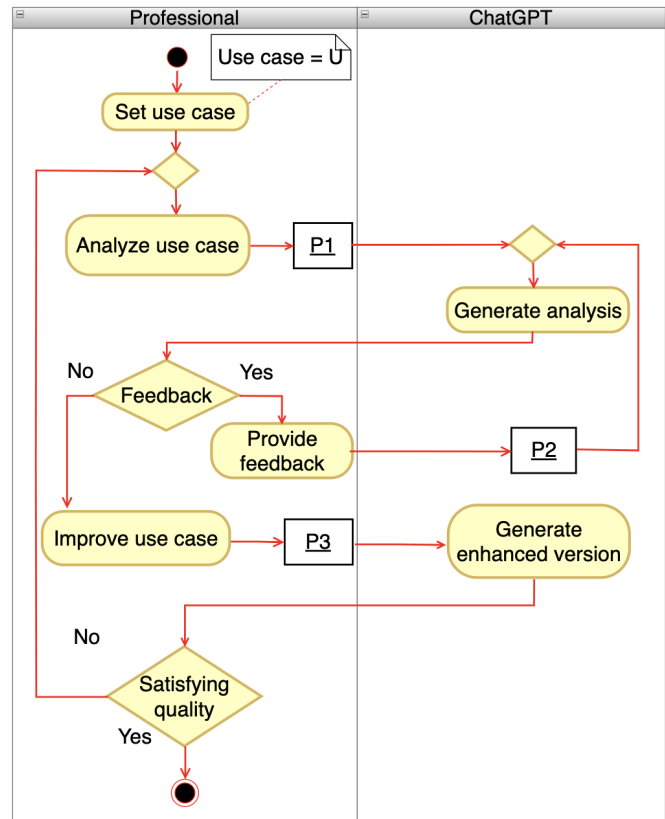


Fig. 6. Enhancing use case quality activity diagram.

had never used LLMs. The consultant was an experienced professional with over 25 years of experience in software engineering and more than 15 years of experience in software effort/size estimation.

For our experiment, we used 5 use cases, written in English, from a use case model provided by an Italian IT Company that works in the Telemedicine field. The use case model contained the requirement specifications of a microservice application based on the IoT paradigm. The use cases had different complexity, quality, and linguistic styles.

### B. Variable selection

We identified the following variables for our experiment:

- **Quality:** high-quality use cases must include elements, such as the basic flow, alternative flows, actor descriptions, preconditions, post condition, and error handling, and must satisfy the following criteria:
  - *Correctness:* use cases must accurately describe the system's behavior regarding the steps in all flows, actors, and pre and post-conditions.
  - *Completeness:* the steps in all flows must be present and in the correct order, all the actors must be

TABLE I  
CRITERIA AND ASPECTS FOR EVALUATING THE QUALITY OF USE CASES

	Basic Flow	Alternative flows	Actor descriptions	Preconditions and postconditions	Error handling
<b>Completeness</b>	<i>No. of missing steps</i>	<i>No. of missing or incomplete flows</i>	<i>No. of missing or inaccurately described actors</i>	<i>No. of missing or incomplete conditions</i>	<i>No. of missing or incomplete scenarios</i>
1. Poor	> 3	> 3	> 3	> 3	> 3
2. Fair	2-3	2-3	2-3	2-3	2-3
3. Sufficient	1	1	1	1	1
4. Good	All essential steps are present	All essential flows are present and complete	All actors are present	All essential conditions are present	All essential scenarios are present
5. Excellent	All steps are present	All alternative flows are present and complete	All actors are present and accurately described	All preconditions and postconditions are present and complete	All error scenarios are present and ensure the system's reliability
<b>Correctness</b>	<i>% of incorrect steps</i>	<i>% of incorrect flows</i>	<i>% of incorrect descriptions</i>	<i>% of incorrect conditions</i>	<i>% of incorrect scenarios</i>
1. Poor	> 50%	> 50%	> 50%	> 50%	> 50%
2. Fair	30 – 50%	30 – 50%	30 – 50%	30 – 50%	30 – 50%
3. Sufficient	10 – 30%	10 – 30%	10 – 30%	10 – 30%	10 – 30%
4. Good	< 10%	< 10%	< 10%	< 10%	< 10%
5. Excellent	All the steps are correct	All the alternative flows are correct	All the actor descriptions are correct	All the conditions are correct	All the error scenarios are correct
<b>Clarity</b>	<i>No. of unclear or ambiguous steps</i>	<i>No. of unclear or ambiguous flows</i>	<i>No. of unclear or ambiguous descriptions</i>	<i>No. of unclear or ambiguous conditions</i>	<i>No. of unclear or ambiguous scenarios</i>
1. Poor	> 3	> 3	> 3	> 3	> 3
2. Fair	2-3	2-3	2-3	2-3	2-3
3. Sufficient	1	1	1	1	1
4. Good	All essential steps are clear and unambiguous	All flows are clear and unambiguous	All actors are clear and unambiguous	All conditions are clear and unambiguous	All scenarios are clear and unambiguous
5. Excellent	All steps are unambiguous and easy to follow	All flows are unambiguous, and add value to the use case	All actors are clearly, and accurately described, and their interactions are clear	All conditions are clear, unambiguous, and accurately reflect the system state	All scenarios are clear and unambiguous, and help ensure system's reliability

accurately described, and pre and post-conditions must be present and complete.

- *Clarity*: use cases must be clear, without ambiguities, and easily understandable by stakeholders. They must not have discrepancies, follow a logical sequence of events, provide the same level of detail for each step, and use consistent terminology.

The above quality criteria are evaluated according to Table I, where the total score (sum of all the scores of the aspects analyzed) for each criterion is calculated as follows: Poor (5-10), Fair (11-15), Sufficient (16-20), Good (21-24), Excellent (25-25).

- **Unadjusted UCP**: as a sort of sanity check, we used the number of Unadjusted Use Case Points (UUCP) of the analyzed use cases to quantitatively assess the impact of the improvement. In particular, we selected UUCP method because it is less subjective and less dependent on weights [19].

### C. Experiment setting

We chose the consultant as the oracle and randomly assigned the other software engineers to the experimental and control groups. The members of the two groups did not know the group they belonged to. We asked both groups to improve the use cases provided. In particular, the experimental group was asked to use ECHO, while the control group was asked to proceed manually.

As a preliminary step, we trained the experimental group's members in an hour-long session on using ECHO. In particular, we taught them how to use the prompts we designed (described in Section IV) to ask ChatGPT to analyze the quality of use cases and get suggestions (**P1**), ask ChatGPT to consider their feedback (**P2**) and provide the new, improved version of the use case (**P3**). We also asked the experimental group to execute at least two iterations of the ECHO process to improve the use cases. We asked each group to keep track of the changes made to the initial use cases regarding the criteria and aspects listed in Table I and to fill in the questionnaire n. 1 (see Tables II and III), reporting the scores for each criterion, and the time taken to improve each use case. Afterward, each group worked independently while members of each group performed the required tasks together.

### D. Data analysis

The use cases improved by the two groups were subjected anonymously to the evaluation of the oracle, based on the criteria indicated in Table I, who proceeded to fill in questionnaire no. 2 (see Table V), reporting the scores of the various criteria and calculating the number of UUCPs of the improved use cases. At the end of the experiment, we asked the experimental group to fill in questionnaire no. 3 (see Table IV), which included questions to rate the level of satisfaction with using ECHO.

TABLE II  
QUESTIONNAIRE NO. 1 – QUALITY ASSESSMENT OF USE CASES EXECUTED BY THE CONTROL GROUP

Use Case	Completeness (Score: 1-5)	Correctness (Score: 1-5)	Clarity (Score: 1-5)	Number of changes	Effort (Minutes)
1	3	3	2	13	13
2	2	3	2	17	16
3	2	3	3	8	9
4	2	3	2	13	17
5	2	3	2	10	12

TABLE III  
QUESTIONNAIRE NO. 1 – QUALITY ASSESSMENT OF USE CASES EXECUTED BY THE EXPERIMENTAL GROUP

Use Case	Completeness (Score: 1-5)	Correctness (Score: 1-5)	Clarity (Score: 1-5)	Number of changes	Number of discarded suggestions	Effort (Minutes)
1	2	3	2	16	0	5
2	2	3	2	14	0	6
3	2	3	2	14	0	3
4	2	3	2	14	0	6
5	2	2	2	20	1	6

TABLE IV  
QUESTIONNAIRE NO. 3 – EXPERIMENTAL GROUP’S EXPERIENCE WITH USING ECHO

No.	Questions	Answers
1	How easy was it to use ECHO to enhance the quality of the use cases? (1 Very difficult – 5 Very easy)	- 4.3 (Mean)
2	How useful do you think ECHO was in enhancing the quality of the use case scenarios? (1 Not useful at all - 5 Extremely useful)	- 4.7 (Mean)
3	How satisfied are you with ECHO in improving use cases? (1 Not satisfied at all - 5 Extremely satisfied)	- 4.7 (Mean)
4	Would you use ECHO to improve the quality of use cases? (1 Not probable at all - 5 Very probable)	- 4.3 (Mean)
5	Do you plan to experiment with ChatGPT for other software engineering tasks? (1 Not at all - 5 Sure)	- 4.3 (Mean)
6	If yes, which? (Requirements Engineering - System Design - Software Development - Testing - Other)	- Requirements Engineering: 100% - Testing: 66%

The initial use cases, the use cases improved by the two groups, the logs of the changes made, and the ChatGPT responses in the different phases of the experiment are available as supplemental material on GitHub at: <https://github.com/gadevito/FGG>.

## V. RESULTS AND DISCUSSION

Tables II and III present the outcomes of questionnaire no. 1, obtained after the two groups improved the provided use cases in the initial phase of our experiment. Fig. 7 illustrates an instance of a use case enhanced by the two groups.

The control group spent from 9 to 17 minutes making 12.2 changes on average per use case, while the experimental group spent from 3 to 6 minutes making 15.6 changes on average and discarding only one suggestion of those provided by ChatGPT, indicating that the use of ECHO can reduce the effort required to improve use cases.

Then, as planned, the oracle assessed the use cases improved by the two groups, calculated their UUCP values, and answered questionnaire no. 2 (to specify the quality of the enhanced use cases) whose results are shown in Table V.

We can observe that the oracle’s assessment of the use cases improved by the control and experimental groups was similar in clarity. However, the experimental group’s improved versions were slightly better in correctness and completeness, indicating that ECHO effectively supports professionals in enhancing the quality of UML use cases. As for the use case size measurement, the original versions were characterized by 101 UUCPs in total, as measured by the oracle. Differently, 126 UUCPs were obtained for the use cases enhanced by the control group and 131 UCCPs for the use case versions enhanced by the experimental group. Analyzing the number of UUCPs obtained and use cases, the Oracle found that the control group merged some system actions, leading to an error in determining the total number of transactions. This incorrectness impacted the obtained UUPCs value, which relies on an accurate transaction count.

Finally, we assessed the level of satisfaction with using ECHO for software engineering tasks from the answers to the questionnaire no. 3 provided by the experimental group. The results are shown in Table IV and indicate that users found ECHO to be easy to use, helpful, and satisfying in enhancing

TABLE V  
QUESTIONNAIRE NO. 2 – QUALITY ASSESSMENT OF THE ENHANCED USE CASES EXECUTED BY THE ORACLE

Use Case	Completeness (Score: 1-5)		Correctness (Score: 1-5)		Clarity (Score: 1-5)		UUCP		
	Control Group	ECHO Group	Control Group	ECHO Group	Control Group	ECHO Group	Initial	Control Group	ECHO Group
1	4	5	4	5	5	4	23	28	33
2	4	4	5	5	5	4	30	30	30
3	4	5	5	5	4	5	20	30	30
4	4	4	4	5	4	4	15	20	20
5	4	5	4	5	4	5	13	18	18

ID: CPM:US1.001

Title: View vitals measurements

Actors: Actors: Doctor, nurse, social and health workers (only for home records), Caregivers (only for home records), Patient (only for home records)

Main scenario:

1. The user selects the 'View vitals measurements' tab for the given Patient.
2. The system searches for the measurements and displays the result list.
3. The actor browses the results.

Alternative flow no. 01:

1. If at step 3 of the main scenario, the actor selects the sub-option 'Add a measurement,' include the use case CPM:US1.002– Add a measurement.

Alternative flow no. 02:

1. If at step 3 of the main scenario, the actor selects the 'display trend,' include the use case CPM:US1.003– Display the graph of the measurements.

ID: CPM:US1.001

Title: View Vitals Measurements

Actors: Doctor can view the patient's recorded vitals measurements to monitor or change the treatment. Caregivers, nurse, and social-health workers can view the patient's recorded vitals measurements to monitor the Patient's health conditions and inform or alert the Patient's Doctor. Patient can view his (or her) vitals measurement to verify his (or her) health conditions.

Preconditions: At least one set of vitals measurements is stored for the given patient. Vitals can be measured more than once per day.

Postconditions:

- The user has successfully viewed the patient's vitals measurements.

Main scenario:

1. The user selects the "View Vitals Measurements" tab for the given patient.
2. The system searches for the measurements and displays them in a table format, including the date and time of each measurement and the corresponding vital sign values.
3. The user views the results.

Alternative flow no. 1:

1. If at step 3 of the main scenario, the actor selects the sub-option 'Add a measurement,' include the use case CPM:US1.002– Add a measurement.

Alternative Flow no. 2:

1. If at step 3 of the main scenario, the actor selects the 'display trend,' include the use case CPM:US1.003– Display the graph of the measurements.

Exceptions:

- If the system cannot find any measurements for the given patient, an error message is displayed to the user.
- If there is an error in displaying the measurements, an error message is displayed to the user.

ID: CPM:US1.001

Title: View vitals measurements

Actors: Doctor, nurse, healthcare professionals (only for home records), caregivers (only for home records), patient (only for home records)

Preconditions:

- The patient must be registered in the system
- There must be existing measurements for the patient.

Postconditions: The system displays the measurements to the actor, and the actor can view or manipulate the data as needed. step 5 of the main scenario.

Main scenario:

1. The actor selects the 'View vitals measurements' tab for the given patient.
2. The system prompts the actor to select a date range or measurement type.
3. The actor selects a valid date range or measurement type.
4. The system retrieves the measurements from the database.
5. The system displays the measurements to the actor.
6. The actor navigates the results.

Alternative flow 01:

1. At step 5 of the main scenario, the actor selects the sub-option 'Add a measurement.'

2. The system transitions to the use case CPM:US1.002– Add a measurement.

Alternative flow 02:

1. At step 5 of the main scenario, the actor selects the 'display trend.'
2. The system transitions to the use case CPM:US1.003 – Display the graph of the measurements.

Error handling:

- If the patient is not registered in the system, the system displays an error message and prompts the actor to register the patient.
- If there are no measurements found for the patient, the system displays a message indicating that there are no measurements available for the selected date range or measurement type.
- If the input date range or measurement type is invalid, the system displays an error message and prompts the actor to enter a valid date range or select a valid measurement type.

Original Use Case

Control Group's improved version

ECHO improved version

Fig. 7. Example of a use case improved by the two groups.

the quality of use cases. In particular, the average score for how easy it was to use ECHO was 4.3 out of 5, suggesting that most users found it straightforward to work with the tool. The average score for how valuable ECHO was in enhancing the quality of use case scenarios was 4.7 out of 5, indicating that users found the approach very effective in improving use cases. Additionally, the average score for satisfaction with ECHO in improving use cases was 4.7 out of 5, which implies that users were generally pleased with the results they obtained from using the tool. Regarding whether users would use ECHO to improve the quality of use cases, the average score was also 4.3 out of 5, indicating that the majority of users were likely to continue using our approach. Furthermore, the results show that participants plan to experiment with ChatGPT for other software engineering tasks, with a special interest for Requirements Engineering and Testing tasks.

**Summary for RQ:** *ECHO can effectively support software engineers in improving the quality of UML use cases in terms of completeness, correctness, and clarity. Moreover, it can help to identify missing details or unclear requirements, obtain*

*more complete transactions in the use case scenarios and, therefore, improve the accuracy of their size estimation in terms of UCPs.*

## VI. THREATS TO VALIDITY

In the following we outline potential issues that may have affected our findings and how we addressed them.

One potential threat to internal validity is the possibility of participant bias. Participants may have answered the questionnaires in a way that they believed would please the researchers or may have been influenced by the knowledge that they were part of an experiment. To mitigate this threat, the researchers ensured that the participants were unaware of the hypothesis being tested and provided anonymity to ensure honest responses. Moreover, the Hawthorne effect is possible, where participants may change their behavior because they know they are being observed. To mitigate this threat, we divided the participants into two groups, reduced the study duration, and allowed the groups to operate without being physically observed.

One potential threat to external validity is the generalizability of the results to the entire population. The study was conducted with a small sample of all business analysts or product owners with experience in UML use cases. Therefore, the results may not generalize to other groups, such as software developers or designers. Moreover, the number and the domain of use cases used for the experiment were limited. Future studies could include a more diverse sample of participants and a higher number of use cases related to different domains to ensure generalizability. Another potential threat is the potential for a novelty effect. Most of the experimental group's participants were using LLMs for the first time, which may have influenced their results. Future studies could include a more extended period for participants to become familiar with LLMs before experimenting.

## VII. CONCLUSIONS AND FUTURE WORK

This study provides initial evidence that ECHO can be effective in enhancing the quality of UML use cases, also reducing the effort with respect to a manual verification. Our study also found that starting from the use cases improved by the experimental group a more accurate size estimation in terms of UUCPs can be derived w.r.t. the ones based on the use cases enhanced by the control group, indicating that ECHO can be effective in improving the accuracy of UCP estimation, given the better quality of the use cases.

Moreover, the study revealed that our approach could help identify missing or unclear requirements, improving overall system quality. The satisfaction survey results show that the users found ECHO easy to use, helpful, and satisfying in enhancing the quality of use cases. Additionally, most users reported continuing to use ECHO to improve the quality of UML use cases. Many expressed interest in using ChatGPT for other software engineering tasks.

Future research could investigate the effectiveness of LLMs in enhancing the quality of other types of software artifacts, such as requirements documents, design specifications, and test cases. In addition, studies could explore the impact of varying the different LLM model hyperparameters.

## ACKNOWLEDGMENT

We thank Kiranet s.r.l. for providing the use case model for the study and thank the professionals for their commitment.

## REFERENCES

- [1] Azzeh, M., Nassif, A. B., Attili, I. B. (2021). "Predicting software effort from use case points: A systematic review." *Science of Computer Programming*, 204, 102596.
- [2] Antonio Bucchiarone, Stefania Gnesi, Giuseppe Lami, Gianluca Trentanni, and Alessandro Fantechi. 2008. QuARS Express - A Tool Demonstration. "In Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE '08). IEEE Computer Society," Washington, DC, USA, 473–474.
- [3] Leah Goldin and Daniel M. Berry. 1997. AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation. "Automated Software Engg." 4, 4 (Oct. 1997), 375–412.
- [4] Miles Osborne and CK MacNish. 1996. Processing natural language software requirement specifications. "In Proceedings of the Second International Conference on Requirements Engineering." 229–236.
- [5] Daniel Popescu, Spencer Rugaber, Nenad Medvidovic, and Daniel M. Berry. 2008. Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs. Springer-Verlag, "Chapter Reducing Ambiguities in Requirements Specifications Via Automatically Created Object-Oriented Models," 103–124.
- [6] Alejandro Rago, Claudia Marcos, and J. Andrés Diaz-Pace. 2013. Uncovering quality-attribute concerns in use case specifications via early aspect mining. "Requirements Engineering" 18, 1 (01 Mar 2013), 67–84
- [7] Alessandro Fantechi, Stefania Gnesi, Giuseppe Lami, and Alessandro Maccari. 2003. Applications of linguistic techniques for use case analysis. "Requirements Engineering" 8, 3 (01 Aug 2003), 161–170
- [8] Avik Sinha, Stanley M Sutton Jr, and Amit Paradkar. 2010. Text2Test: Automated Inspection of Natural Language Use Cases. "In 2010 Third International Conference on Software Testing, Verification and Validation." 155–164.
- [9] Bente Anda and Dag I. K. Sjøberg. 2002. Towards an Inspection Technique for Use Case Models. "In Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE '02). ACM," New York, NY, USA, 127–134.
- [10] William Brace and Kalevi Ekman. 2014. CORAMOD: a checklist-oriented modelbased requirements analysis approach. "Requirements Engineering" 19, 1 (2014), 1–26.
- [11] Karl Cox, Aybuke Aurum, and Ross Jeffery. 2004. An Experiment in Inspecting the Quality of Use Case Descriptions. "Journal of Research and Practice in Information Technology" 36, 4 (2004), 211–229.
- [12] KeithThomas Phalp, Jonathan Vincent, and Karl Cox. 2007. Improving the quality of use case descriptions: empirical assessment of writing guidelines. "Software Quality Journal 15," 4 (2007), 383–399.
- [13] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. "arXiv".
- [14] Alistair Cockburn. 2001. Writing effective use cases. Vol. 1. "Addison-Wesley Boston."
- [15] Ivar Jacobson, Magnus Christerson, Patrik Jonsson and Gunnar Overgaard. 1992. Object-Oriented Software Engineering: A Use-Case Driven Approach ("AddisonWesley," Reading, MA, 1992 ed.).
- [16] Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E. V., and Batista-Navarro, R. T. (2021). Natural language processing for requirements engineering: A systematic mapping study. "ACM Computing Surveys (CSUR)," 54(3), 1-41.
- [17] Sobania, D., Briesch, M., Hanna, C., and Petke, J. (2023). An analysis of the automatic bug fixing performance of chatgpt. "arXiv preprint arXiv:2301.08653."
- [18] Ahmad, A., Waseem, M., Liang, P., Fehmideh, M., Aktar, M. S., and Mikkonen, T. (2023). Towards Human-Bot Collaborative Software Architecting with ChatGPT. "arXiv preprint arXiv:2302.14600."
- [19] G. Karner, Metrics for objectory, Master's thesis, University of Linköping, Sweden, 1993.
- [20] Beurer-Kellner, L., Fischer, M., and Vechev, M. (2022). Prompting Is Programming: A Query Language For Large Language Models. arXiv preprint arXiv:2212.06094.
- [21] Yogatama, D., d'Áutume, C. D. M., Connor, J., Kocisky, T., Chrzanowski, M., Kong, L., ... & Blunsom, P. (2019). Learning and evaluating general linguistic intelligence. arXiv preprint arXiv:1901.11373.
- [22] Gao, T., Fisch, A., & Chen, D. (2020). Making pre-trained language models better few-shot learners. arXiv preprint arXiv:2012.15723.
- [23] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877-1901.
- [24] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A. ... & Fiedel, N. (2022). Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311.
- [25] Kharya, P., & Alvi, A. (2021). Using DeepSpeed and megatron to train megatron-turing NLG 530B, the world's largest and most powerful generative language model. NVIDIA Developer Blog.
- [26] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., ... & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:2302.11382.