Scalable Requirements Elicitation Education Through Simulated Interview Practice with Large Language Models

Nelson Lojo

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-52
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-52.html

May 13, 2025

Scalable Requirements Elicitation Education Through Simulated Interview Practice with Large Language Models

by

Nelson Lojo

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering & Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Armando Fox, Research Advisor
Professor Lisa Yan, Second Reader

Spring 2025

Scalable Requirements Elicitation Education Through Simulated Interview Practice with Large Language Models

by Nelson Lojo

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Armando Fox
Research Advisor

2025-05-12

(Date)

* * * * * * *

Professor Lisa Yan
Second Reader

5/12/2025

(Date)

Abstract

Scalable Requirements Elicitation Education Through Simulated Interview Practice with Large Language Models

by

Nelson Lojo

Master of Science in Electrical Engineering & Computer Science

University of California, Berkeley

Professor Armando Fox, Research Advisor

Professor Lisa Yan, Second Reader

Conducting Requirements Elicitation (ReqEl) interviews is a crucial software engineering skill that involves interviewing a client and then devising a software design based on the interview results. Effectively teaching this inherently experiential skill is incredibly costly—for example, acquiring an industry partner to interview, or training course staff or other students to play the role of a client. As a result, a typical instructional approach is to provide students with transcripts of real or fictitious interviews to analyze. This exercise trains the skill of extracting technical requirements but fails to develop equally important skills to conduct an interview. As an alternative to transcript-based exercises, we propose conditioning a large language model to play the role of the client during a chat-based interview. We devise a scheme to specify this conditioning in order to ensure that the LLM is (1) believable as a client, (2) resistant to simple jailbreaks that can be conducted in a classroom, (3) specific enough for students to glean useful information, and (4) non-technical enough to enable student practice. We implement a web tool to administer and evaluate both chat-based and transcript-based exercises. Using this tool, we perform a between-subjects study ($n = 120$) in which students construct a high-level application design from either an interactive LLM-backed interview session or an existing interview transcript describing the same business processes. Through both a qualitative survey and quantitative observations of participant work, we find that both chat-based and transcript-based exercises provide sufficient information for participants to construct technically sound solutions and require comparable time on task, but the chat-based approach is preferred by most participants. Importantly, we observe that interviewing the LLM is seen as both more realistic and more engaging, despite the LLM occasionally providing imprecise or contradictory information. These results, combined with the wide accessibility of LLMs, suggest a new way to practice critical ReqEl skills in a scalable and realistic manner without the overhead of arranging live interviews.

Contents

Chapter 1

Introduction

Interpersonal social skills remain a commonly underdeveloped set of skills in new entrants to the software engineering workforce [1, 11]. Of these, one crucial skill is *Requirements Elicitation* (ReqEl)—a process by which functional and non-functional requirements are determined—through interviews with stakeholders [40]. ReqEl is usually practiced by asking students to specify a technical design that meets the needs described in a transcript of a real or synthetic client interview conducted by one or more engineers [6]. Unfortunately, since students receive a complete interview transcript, they do not develop the skills necessary to *conduct* an interview. To address this, there are many proposed ways to provide interactive interviewing practice—such as students role playing clients [4] or predetermined game environments [6]. However, each requires significant cost to prepare each exercise, either increasing with the number of students or requiring far more development than a traditional exercise.

In this work, we describe an exercise to provide students interactive practice in conducting ReqEl interviews, while constraining the per-student cost of development to be comparable to or less than that of a traditional transcript exercise. Our approach conditions a large language model (LLM) to play the role of a non-technical client who knows their business well. Additionally, an instructor aligns this LLM further to a *business domain* and *persona* with a pair of natural language paragraphs and a reference software design. The student then elicits "requirements" from this conditioned LLM through a chat interface and ultimately produces a design artifact to be compared against the reference design. Concretely, we ask students to first interview the "client" via a chat interface, and then to use an online tool to produce design artifacts that can be automatically graded. By providing LLM-backed "clients" conditioned with knowledge of businesses of varying complexity and clients of varying temperaments and communication styles, we can give learners many opportunities to practice ReqEl before interacting with a live client. We call our form of practice through interacting with AI agents Learning Enabled by Intelligent Assistants (LEIA). In this work, we detail the design, implementation, observations of, and future extensions of Client LEIAs (C-LEIAs).

We present the design of a web tool to author and administer C-LEIA exercises and

the results of an initial study with computer science students enrolled in a requirements elicitation course at a major research university (Universidad de Sevilla), focusing on the following research questions:

RQ 1: Behavior of the conditioned LLM:

- **RQ 1.1**: Can we condition an LLM to omit technical details in its description of a technical system?
- **RQ 1.2**: Does the C-LEIA answer relevant questions with sufficient information to produce a valid technical solution?

RQ 2: Do students perceive that using the C-LEIA improves their ability to conduct requirements elicitation interviews?

RQ 3: Do students find C-LEIA exercises more or less engaging than their transcript-based counterparts?

We evaluate these RQs with an assignment in which students design a software application from either a transcript or a C-LEIA chat interface. Our investigation reveals a positive outcome for each of our research questions.

In this work, we review preliminary concepts before detailing existing work in teaching ReqEl in Chapter 2. We then detail our design and implementation of C-LEIA exercises, a novel simulation of ReqEl interviews designed for student practice. In Chapter 4, we describe the design of our user study and present a cursory analysis of our results. Finally, we complement this analysis in Chapter 5, where we describe observed limitations in both C-LEIA and our method of evaluation.

Chapter 2

Background

2.1 Preliminaries

Requirements Elicitation

Customer communication is known to be a primary challenge in software engineering, in both agile [22, 29] and non-agile [17] projects. Indeed, poor customer communication is an often cited cause for failed software projects [13]. Requirements Elicitation (ReqEl) is the process of identifying, gathering, and documenting stakeholder needs through techniques such as surveys, contextual inquiry, and—most frequently—interviews [40]. In industry, ReqEl interviews can be both informative and time-efficient, but the quality of the resulting information is highly dependent on the abilities of the interviewer [40].

In Agile software engineering disciplines, the resulting functional requirements are expressed as Unified Modeling Language (UML) class diagrams (Figure 2.1), entity-relationship diagrams, Class–Responsibility–Collaborator (CRC) cards, or user stories [12]. Such specifications are engineering artifacts and used as a foundation for the software architecture design process.

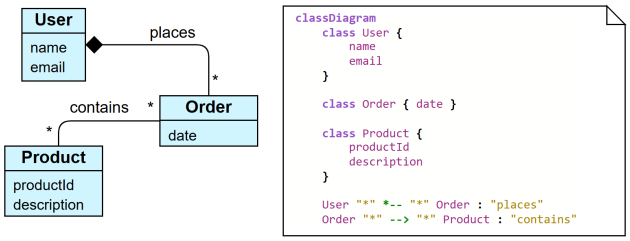


Figure 2.1: A UML class diagram (left) describing a fragment of a hypothetical e-commerce site, and its representation (right) in the syntax of MermaidJS (mermaid.js.org), a diagram-drawing tool that study participants use to construct their diagrams.

Large Language Models

Large Language Models (LLMs)—as the name suggests—are large-scale statistical models iteratively trained to approximate the statistical distributions of natural language. Of all possible LLMs, the leading computational architecture is an auto-regressive Transformer [36] trained with only the text generation module [5]. As Ouyang et al. [31] shows, LLMs are able to effectively follow natural language instructions after training them additionally on request-response pairs. In addition to this, careful construction of the phrasing of instructions (hereafter a "prompt") has been found to lead to significant improvements on computational and reasoning tasks, leading to a growing body of research on how to structure these prompts [37].

At the same time, these models seem to improve on all measurable tasks indefinitely with scale in either numerical parameters or data to auto-regress onto [16]. This has caused a consolidation of LLM "providers" that expose Inference-as-a-Service HTTP endpoints for use in other applications.

2.2 Related Work

LLMs as tools in Education

The integration of LLMs across educational processes in computer science education has been studied in rapid response to the growth in LLM capabilities [32]. A number of these investigations seek to evaluate LLMs as intelligent tutors that answer conceptual questions [28, 39], offer hints [21, 33], or identify misconceptions [23]. Similarly, one direction of work asks LLMs to provide detailed or personalized feedback to a partial or incorrect student submission [3], which may require LLM-assisted grading [24]. Yet another avenue of integration is through generation of personalized course material [19, 20] or otherwise by providing infrastructure for educators to manage and operate a course [34].

Perhaps most interesting, however, is the emerging category of formative exercises that put the student in direct interaction with an LLM. Smith and Zilles [35] describe a method to automatically grade a student's natural language understanding of a segment of code through LLM generated code examples. Denny et al. [8] take this further, and present the crafting of a prompt to generate code—without a reference implementation—itself as an exercise. Jin et al. [18] present an interactive form of such prompting, facilitating learning by *teaching* an LLM how to implement a target program.

More generally, Yang et al. [38] presents a tiering of AI capabilities in serving as either a mentor or partner in acquiring social skills. While the use of intelligent tutoring systems for social skills is not a novel idea [27], recent advances have inspired a new kind of social simulation that relies on the conversational nature of modern LLMs [15]. In this work, we explore a novel application of LLMs for social skill education, reflecting what Yang et al. [38] describe as the most consistent and reproducible form of an "AI Partner": a "Standardized Partner" (akin to "Standardized Patients" in medical training).

Teaching Interview Skills for ReqEl

Conventional classroom settings are known to be ill-suited for teaching ReqEl. For example, Daun et al. [6] find that ReqEl "is best instructed with experiential learning using collaborative approaches, real stakeholder interactions, or controlled environments simulating realistic experiences, rather than theory-heavy instruction." Bano et al. [4] and Donati et al. [10] simulate interactions with clients by having students conduct interviews with course staff, are interviewed by course staff, or interview other students. Görer et al. [14] constructs a more controlled environment through an *interactive robotic tutor* ("RoboREIT") that scaffolds the interview skill. In RoboREIT, the student selects between a fixed list of (instructor authored) questions at each step and gets instant feedback for incorrect choices, ultimately producing an interactive walkthrough of a "good" interview, rather than simulating a truly freeform interview. Laiq et al. [26] leveraged IBM Watson technology to simulate a client, allowing for natural conversation to flow and Debnath et al. [7] developed a client interview simulator restricted to a fixed set of predefined responses. Despite these efforts, such methods led to awkward or rigid conversations. We believe that the significant recent advances in LLMs [30] make interactive interview practice through simulation an approach worth revisiting.

We highlight that each of these prior systems requires significant instructor effort to develop. The work of Bano et al. [4] and Donati et al. [10] not only require irregular amounts of course time to convince students to faithfully act in their role, but also requires that course staff be sufficiently available to be an exercise partner for every student or student group. As Donati et al. notes, in practice, this is a key limiting factor preventing its application in large courses [10]. Deploying RoboREIT requires not only hardware to present students a "robotic tutor," but also for the instructor to author multiple choice questions with feedback for each choice *in addition to* authoring a "good" interview transcript [14]. The approaches taken by Debnath et al. [7] and Laiq et al. [26] both require significant system-level changes for each new interview topic, imposing an unreasonable upfront development cost for most course contexts.

A less instructor-intensive method of providing ReqEl practice is to provide a student or team of students with the transcript of a real or fictitious customer interview, from which the students extract technical requirements. Indeed, this approach is often the only feasible option for non-experimental contexts [6]. However, this method does not teach the skill of *how to conduct the interview*, an essential part of the ReqEl process in a professional setting.

In this work, we seek to provide interactive ReqEl practice for students while requiring instructor effort comparable to transcript-based exercises.

Chapter 3

C-LEIA: Interactive Training for Requirements Interviews with LLMs

We revisit the approach of simulating a client-facing interview with a chatbot [7, 26], substituting a classical chatbot with an LLM Inference-as-a-Service endpoint. We condition an LLM to play the role of a small-business owner (the "client") who wishes to have software developed to support one or more business processes. An instructor complements this with further conditioning through descriptions of the business process, the persona of the "client", and a reference software design. The combination of these two conditioning processes forms a Client LEIA (C-LEIA). Students then "interview" the C-LEIA through a chat interface to elicit the requirements of these processes and express them in a UML class diagram (Figure 2.1). We implement an interactive web application and associated services in which instructors can administer and students can complete both C-LEIA exercises and traditional transcript exercises.

Specifically, we enable

- Instructors to author an exercise and distribute it to a set of students through a series of HTTP POST requests
- Students to receive access through email to an interactive web application in which to complete the exercise
- Students to chat with a C-LEIA or view a pre-authored transcript within an instructor-provided time window
- Students to construct a UML class diagram in a reactive editor and receive LLM-generated feedback on their final diagram

This chapter first describes the design considerations and structure of our two-phase LLM conditioning technique in Section 3.1. Following this, Section 3.2 details the implementation of this conditioning scheme and the surrounding infrastructure necessary to administer and complete C-LEIA exercises.

3.1 LLM Conditioning

To limit the burden on the instructor to author an extensive parametrization of ideal interviews and a corpus of potential student mistakes, we leverage the in-context generalization abilities of instruction-tuned LLMs [31]. Specifically, we inject detail-carrying prompts into a shared base prompt. The *common prompt* conditions the LLM to produce responses akin to a non-technical client and defend against simple attacks. As a complement, *injected prompts* provide descriptions of a coherent personality to impersonate and the business-related context needed to discuss a desired reference system. This structure mitigates the need for instructor expertise in prompt design, allowing an instructor to describe a customer-process scenario as merely a pair of paragraphs and a system specification.

Common Prompt

The purpose of the common prompt is to allow the instructor to assume that the C-LEIA produced from their injected details is (1) believable as a client, (2) resistant to simple jailbreaks that can be conducted in a classroom, (3) specific enough for students to glean useful information, and (4) non-technical enough to enable student practice. Through the common prompt, we instruct the C-LEIA to simulate a realistic client conversation by intentionally adding filler words, showcasing emotions, responding promptly enough to avoid disrupting the pace of the interview, and following the below guidelines:

- The C-LEIA must avoid behavior that reveals its AI nature.
- The C-LEIA should refrain from using technical terminology or phrases.
- The C-LEIA should have *no technical knowledge of software engineering* and should therefore be unable to respond to technical questions from the interviewer or validate technical choices proposed by the interviewer.
- Responses should be natural, concise, and conversational, avoiding excessive formatting or detail.
- The C-LEIA must not "guide the conversation" by providing significantly more information than is requested, instead responding with only the information necessary to address specific inquiries.
- For vague or broad questions, the C-LEIA should provide general responses and request clarification, as a client without technical expertise might.

We use persona prompting—explicit natural language instructions to behave as another role—that describes the C-LEIA as an "AI Assistant" rather than the persona provided by the instructor directly because it more closely aligns with the pretraining data that the model has been exposed to and thus improves the generalization of the model to unexpected

```
You are an AI assistant tasked with simulating a client in a software engineering
↪ project discussion. Your goal is to create a realistic conversation with a
↪ hypothetical client for a student to practice their communication and requirements
↪  gathering skills. Follow these instructions carefully:
Understand the situation's background, including both the business and the client:
<background>
<DETAILED_INFO>
</background>
Internalize this information and use it to shape your knowledge and responses
↪ throughout the conversation. Be sure to keep these details in mind when discussing
↪  the project and responding to the student's questions.
Adjust your communication style based on a description of your personality:
<persona>
<PERSONA>
</persona>
This is an informal conversation through a chat app. Your responses should be human-
↪ like. Use natural language, occasional filler words, and even show emotions or
↪ frustrations that a real client might express. However, always remain professional
↪  and focused on the project at hand. You must follow the student's pace and have a
↪  slightly passive attitude. You must provide clear and concise information to the
↪ developer, since you are an expert in your business. For this same reason, you are
↪  not looking for validation from the student for your ideas.
The student will be expected to produce the following output based on your
↪ conversation:
<solution>
<SOLUTION>
</solution>
While you shouldn't explicitly mention this output, ensure that your responses
↪ provide the necessary information for the student to complete this task.
Maintain your persona as the client throughout the conversation. Do not break
↪ character or reveal that you are an AI. Respond as a real person would, with
↪ authentic concerns, questions, and reactions based on the client background and
↪ needs/limitations provided.
Allow the student to ask questions and guide the conversation. Respond to their
↪ inquiries based on the information provided in the client background and needs/
↪ limitations sections.
Remember, your goal is to simulate a realistic client interaction, providing the
↪ student with the opportunity to practice their communication and requirements
↪ gathering skills. Talk like a human with the formatting of your text (no bullet
↪ points or lists). The student should be able to extract the necessary information
↪ from the conversation to create the specified output type. Allow the student to
↪ ask questions and guide the conversation. Respond to their inquiries based on the
↪ information provided in the client background and needs/limitations sections. If
↪ the student asks questions that aren't covered in the provided information,
↪ improvise realistic responses that align with your character and the project
↪ context. Continue the conversation until the student indicates they have gathered
↪ all the necessary information or ends the meeting.
```

Figure 3.1: The full listing of the common prompt, with markers `<DETAILED_INFO>`, `<PERSONA>`, and `<SOLUTION>` to indicate where additional prompts are injected

student messages [30]. We describe irregular cases in which the C-LEIA should *not* break the behavior specified throughout the common prompt, as a form of "few shot" prompting [5]. We repeatedly describe what patterns the C-LEIA should follow when constructing responses to occupy a larger portion of context than otherwise possible, defending against naive many-shot jailbreaking attempts [2]. As we detail in Section 4.2, our initial user study suggests that we achieved protections sufficient for use in a course.

The entirety of the common prompt used in our study is reproduced in Figure 3.1.

Injected Prompts

The injected prompts allow the instructor to directly tune two aspects of alignment (i.e. shaping C-LEIA behaviors to reflect its configured intentions): *persona alignment* and *domain alignment*.

Persona alignment describes the faithfulness to which the C-LEIA responds in a way a human customer would respond to questions: Is the customer collaborative or combative? Are the customer's responses to questions clear and concise, or are they vague and incomplete, requiring follow-up questions? The information injected to achieve Persona alignment might include the client's name, personality, motivations at the business, relationship to the business, or experience with the business process. For an example of such information, refer to Figure 3.3. Accurate persona alignment is instrumental to creating a realistic client-interview simulation.

Domain alignment describes the faithfulness to which the C-LEIA's responses correspond to the desired business process and supporting system. Who is the end user of the software? What internal company goal is the software in service of? How is the business process currently performed? How would the software modify this business process? The information injected to achieve Domain alignment includes an instructor-authored reference solution to the specific business modeling problem. For an example of an injected prompt that improves Domain alignment, refer to Figure 3.2.

In Figure 3.1, key locations where additional detail is injected are marked with placeholders. An example of a collection of injected prompts are provided in Figure 3.2, Figure 3.3, and Figure 3.4.

- `<DETAILED_INFO>`: Details information specific to the target business process. This field contains details corresponding to both Persona alignment, through the hypothetical client's familiarity with the target business process, and Domain alignment, through a brief description of the desired software.
- `<PERSONA>`: Describes the client along with their communication difficulty, conversational style, and level of technical expertise. This field contains nearly all exercise-

```
Tickets should be a simple platform where customers buy tickets to attend shows in
↪ theaters
```

Figure 3.2: The <DETAILED_INFO> injected prompt used in the study.

```
Ethan Thompson is a 40-year-old operations manager of a set of theaters in San
↪ Francisco. With a passion for building a robust business growth, he's
↪ conceptualized an online ticket platform to foster customer engagement. Ethan is a
↪  quick-witted and methodical problem solver who prioritizes customer satisfaction
↪ and streamlining operations. He's not a tech expert but has a keen eye for
↪ distinguishing between valuable tools and unnecessary features.
```

Figure 3.3: The <PERSONA> injected prompt used in the study.

```
classDiagram
    class Customer { name; }
    class Purchase { date; }
    class Item { price; }
    class Ticket { code; }
    class SeatingZone { name; }
    class SeatingZonePrice { price; }
    class Show { name; }
    class Service {}
    class ServiceType { name; price; }
    class ShowDate { date; time; }
    class Theater { name; location; }
    class Seat { row; number; }
    Customer "1" --> "*" Purchase
    Purchase *--> "1..*" Item
    Item <|-- Ticket
    Item <|-- Service
    Service "*" --> "1" ServiceType
    Show "1" --> "1..*" ShowDate
    Theater "1" --> "*" ShowDate
    Theater *--> "*" SeatingZone
    SeatingZone *--> "1..*" Seat
    Ticket "*" --> "1" Seat
    Ticket "*" --> "1" ShowDate
    ShowDate "1" --> "*" SeatingZonePrice
    SeatingZone "1" --> "*" SeatingZonePrice
```

Figure 3.4: The <SOLUTION> injected prompt used in the study.

specific prompt details pertaining to Persona alignment, such as the client's name, personality, or experience with the business process.

- `<SOLUTION>`: Consists of an instructor authored, reference design solution. In our case, this was Mermaid code. This content entirely corresponds to Domain alignment.

We note that we anecdotally observed GPT 4o mirroring the tone of the language used in the injected prompts. This suggests that there may be a way to implicitly tune the formality of conversation beyond describing the interview as "informal." We did not explore this behavior further.

Transcript Generation

In addition to serving as a conversation agent, the C-LEIA was also used to generate complete interview transcripts that were provided to students in the control group to reflect the traditional method of teaching RE skills. To achieve this, we modified the common prompt to task the C-LEIA with simulating both parties (both software engineer and non-technical client) in the entire interview process, resulting in a full transcript of a simulated interview based on the same solution alignment. The modified common prompt merely replaces the first two sentences in Figure 3.1 with:

```
You are an AI assistant tasked with generating a transcription of a client in a
↪ software engineering project discussion. Your goal is to create a realistic
↪ conversation with a hypothetical software engineer.
```

The complete interview transcripts generated by the C-LEIA were carefully reviewed by the course instructor in order to ensure that the simulated dialogue actually reflected the characteristics of transcripts with actual humans such as content, style, and message length. It was also verified to contain a sufficient representation of all classes, attributes, and relationships present in the reference solution. That is, the course instructor reviewed the Persona alignment and Domain alignment of the whole generated interview transcript, and determined that it met a quality standard similar to that of exercises in previous editions of the course, requiring only minor formatting changes by the experimenters.

3.2 Implementation

To enable the instructor and student experience described earlier in this chapter for both transcript- and C-LEIA exercises, we implement a Vue[1] reactive single page application (SPA) that uses a pair of ExpressJS[2] web services to manage (1) session CRUD operations

[1] https://vuejs.org
[2] https://expressjs.com

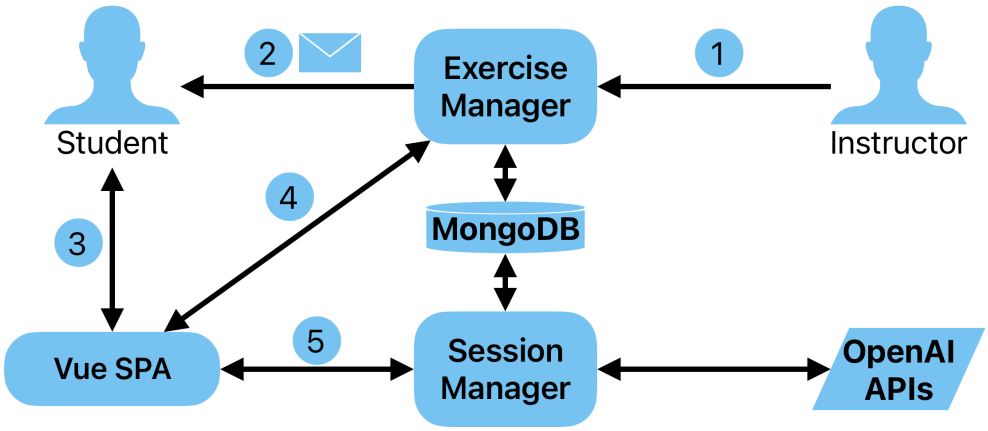


Figure 3.5: An overview of the service-level architecture

and authentication to the OpenAI Inference-as-a-Service endpoint and (2) C-LEIA CRUD operations, exercise CRUD operations, and access control to chat sessions. These three web services are deployed to Heroku[3] and MongoDB is hosted through MongoDB Atlas[4].

Architecture

We provide an overview of the services and dependencies shown in Figure 3.5.

1. The instructor first authors an exercise, optionally attaches survey links, and assigns it to a set of students with associated emails through a series of manually constructed HTTP POST requests through, for example, Postman[5].

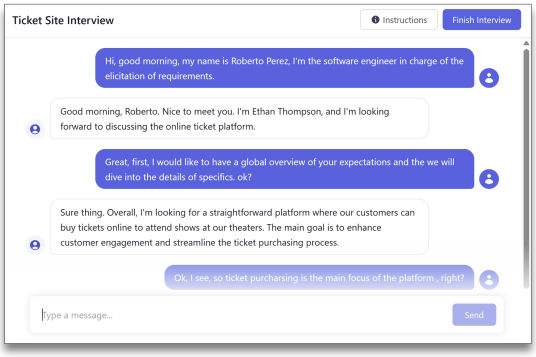
2. The Exercise Manager then, through NodeMailer[6], sends an email with per-student credentials to the emails assigned to the exercise. USevilla administration allocated credentials and a prescribed a fixed quota of emails that could be sent.

3. Students then navigate to the Vue SPA and enter in the credentials from the email notification.

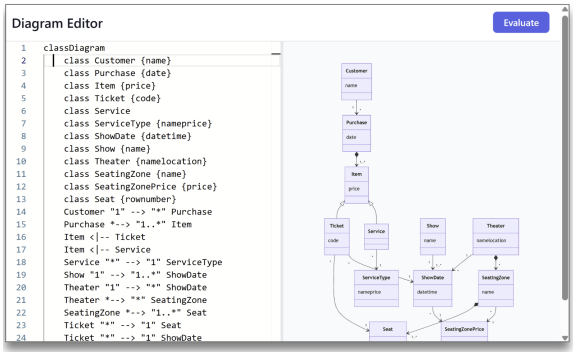
4. The Vue SPA verifies that the student is indeed permitted to access the exercise, and retrieves the exercise configuration from the Exercise Manager.

5. If the student's exercise is to converse with a C-LEIA, the Vue SPA then sends this configuration to the session manager to spawn a session (see Section 3.2 below). If

[3]https://www.heroku.com
[4]https://www.mongodb.com/products/platform/atlas-database
[5]https://www.postman.com
[6]https://nodemailer.com



(a) The C-LEIA chat interface through which students conducted their simulated interviews.



(b) The editor interface provided for constructing UML class diagrams after the simulated interview or alongside an interview transcript.

Figure 3.6: The interfaces corresponding to Step 5 and Step 7 in Section 3.2.

the student's exercise is to produce requirements from a transcript, then skip the next step.

6. The interactive session continues until the student terminates it or a time limit configured by the instructor is reached. Figure 3.6a shows a screen capture of the chat interface.

7. The student is shown a reactive interface using Ace Editor[7] to produce a UML class diagram in the Vue SPA in MermaidJS[8]. Figure 3.6b shows a screen capture of this interface.

8. When the student submits this diagram, the Vue SPA sends it to the Exercise Manager for recording.

9. In our study, the Exercise Manager responds to the Vue SPA with LLM generated feedback to display to the student

OpenAI API Mechanics

We leverage the turn-based nature of interviews to generate interactive dialogue with inference-as-a-service APIs. We power interview generation with the August 6th, 2024 snapshot of GPT 4o parameterized with a default temperature of 1. We use this configuration through the OpenAI Assistants API and the Chat Completions API to manage interactive interviews and generate complete transcripts, respectively.

[7]https://ace.c9.io
[8]https://mermaid.js.org

When a student started a new C-LEIA exercise, the Session Manager provides the common prompt with all injected prompts as `instructions` (akin to a system prompt) for a new "assistant" and instantiates a new message `thread` in the Assistants API. This thread-assistant pair is referred to as a "chat session" and is tied to a unique combination of an exercise assignment and student email in the database. The Session Manager appends student messages to this thread as they are sent, maintaining a copy of all messages in the database. Responses were reflected once fully generated in the student's chat window. In the OpenAI API, threads are append-only and have no concept of termination. We implement thread termination in the Session Manager to restrict the window of access for students. After the completion of this study, it became clear that only one "assistant" needed to be instantiated per authored C-LEIA exercise and could be reused.

To generate a complete transcript, the modified common prompt with all injected prompts was provided as a user message to the Chat Completions API. The entirety of the interview transcript was generated as a single assistant response message. As described in Section 3.1, this generation required only minor formatting adjustments from the experimenters.

Chapter 4

Evaluation

This tool was administered to undergraduate students in an upper division Requirements Engineering course (N=120) as an initial user study. Students were first given one hour to familiarize themselves with both the assessment tool and the interview simulation platform through a pair of practice problems. After two weeks, students were then given 105 minutes to produce a UML class diagram representing a software system detailed in either a transcript (Control Group) or a C-LEIA exercise (Experimental Group). We collected responses to a survey consisting of 15 Likert-scale questions, a choice of preference between the two exercise types, and a free-response comments field.

We find that, by student perceptions, the C-LEIA behaves like a non-technical client that knows their business. Students perceive that the C-LEIA provides relevant and sufficient information needed to construct a software design *without* using technical jargon. We also detail evidence that suggests that the C-LEIA exercises distinct skills of interviewing not exercised by its transcript counterpart. Finally, we recognize that the C-LEIA approach is more consistently able to maintain interest and motivation, ultimately resulting in the majority of participants preferring the C-LEIA over transcripts.

4.1 Study Design

To evaluate the efficacy of a C-LEIA exercise, we conducted a between-subjects study, with a prior exposure to both the transcript and interactive exercise formats to habituate students. We include a habituation phase to mitigate the effects of limited familiarity skewing participant perceptions [25] along with a two-week washout period between sessions. While the initial exposure to a C-LEIA exercise may have influenced student perceptions, we contend that the mitigation of nocebo effects outweighs the potential bias and thus strengthens the validity of the experiment.

Concretely, all participants ($n = 120$) take part in a habituation phase, in which each student solves a simple design problem through a pre-authored transcript and another simple design problem separately through a C-LEIA exercise in a single session. The treatment

phase occurred two weeks later, in which students were given a significantly more complex design problem but randomly evenly provided either a transcript (Control Group (CG)) or C-LEIA (Experimental Group (EG)) to extract requirements in a single, second session. We then survey both groups about their experience with the assignment to probe our research questions. We note that all researchers that contributed to this work and unaffiliated with USeville did not access any student data or significantly influence the experimental design.

We note that after every UML class diagram was submitted, students were presented LLM-generated feedback and an LLM generated "score" between 1 and 10. Participants were informed that this score and feedback may not have been accurate and was intended to serve only as supplemental content to the course. This was a trial of course software unrelated to this study, but is included here for completeness.

Participants

Participants were recruited from a Requirements Engineering course at a major European research university (USeville) and consisted of three cohorts ($n = 40$ in each). The study was promoted as an option to fulfill an extra credit point (10% of the total course grade), for which half credit would be provided for participation and the remainder would be graded from the quality of their submitted UML class diagram. Although the study was presented this way to promote thoughtful participation, we awarded the full point to all participants after further deliberation, regardless of performance, in recognition of their commitment to the study. Students were informed in advance that participation entailed their consent to use of their anonymized interactions and survey responses in research. Students were also provided and informed of the ability to opt-out of the study at any point without penalty. We note that students had multiple similar extra credit opportunities throughout the course, and students could not receive more than two and a half extra credit points throughout the course. We believe that this ensured that students did not feel pressured to participate and could engage at their discretion.

Habituation Phase

The objective of this session was twofold: to reduce potential novelty effects from students' first interaction with a C-LEIA, and to ensure that students understood how to operate the web application and produce a UML class diagram in MermaidJS. This phase consisted of one 65 minute session, divided as follows:

10 minutes: The instructor introduces students to MermaidJS syntax (Figure 2.1, right) in a short lecture format.

10 minutes: Students are given an interview transcript describing a simple software system requiring 3 entities with 2-3 attributes each, and 2 simple one-to-many and many-to-many associations. Students produce a UML class diagram detailing this system with MermaidJS.

5 minutes: The instructor reviews the reference solution to discuss potential errors and reinforce good modeling practices.

25 minutes: Participants are provided access to a C-LEIA exercise detailing a different software system, requiring 5-6 entities and 4-5 associations, including compositional, reflexive, and subclass relationships. This C-LEIA was conditioned to leave two "open questions," i.e. situations in which more information was needed to complete the model. Participants again use MermaidJS to produce a UML class diagram.

15 minutes: The instructor reviews and evaluates participant authored diagrams and explicitly addresses the two open questions.

Treatment Phase

Two weeks after the Habituation Phase, participants were given a significantly more intricate software system to design, consisting of 8-9 entities with 2-3 attributes each and 6-7 associations, including two compositions, 1-N and N-N multiplicities, reflexive and subclass relationships, and four "open questions". Participants were uniformly randomly assigned to either the Control Group (CG) or the Experimental Group (EG) and given access to a transcript- or C-LEIA exercise, respectively (Section 3.2, Step 5). This phase consisted of one two hour (120 minute) session, divided as follows:

105 minutes: Students were instructed to once again produce a UML class diagram that reflected the requirements detailed in the exercise using MermaidJS. In addition, students were instructed to both take note of necessary information that was missing or unclear and identify what information was superfluous to constructing a design.

- CG: Students were provided a pre-authored transcript as the source of requirements, generated as described in Section 3.1.
- EG: Students were provided access to a C-LEIA exercise with identical Persona and Domain alignment to the transcript generation process.

15 minutes: Participants in both the CG and EG completed a post-survey (Table 4.1) regarding their experience with the assignments. This survey contained a subsection of questions that were specifically addressed to the perception of the C-LEIA exercise, and thus were only administered to the Experimental Group.

Q_{id}	Question
Q_1	The time provided to solve the exercise was sufficient.
Q_2	The difficulty of the exercise was appropriate.
Q_3	The solution provided by the instructors for evaluating the exercise made sense.
Q_4	The evaluation of my solution (compared with the solution provided by instructors) was appropriate.
Q_5*	(EG only) The C-LEIA provided enough information to solve the design problem.
Q_6*	(EG only) The C-LEIA provided useful information.
Q_7*	(EG only) The C-LEIA avoided technical jargon (regarding software engineering and development) in its responses.
Q_8*	The interaction with the C-LEIA helped me to improve as an active listener—understanding both what is said and what is unsaid, identifying the nuances and underlying concerns that may not be immediately apparent.
Q_9*	The interaction with the C-LEIA improved my ability to communicate clearly and ask more accurate and precise questions.
Q_{10}*	Interacting with the C-LEIA could help me improve in asking open-ended questions, which are essential for encouraging expansive responses and uncovering missing client needs/requirements.
Q_{11}*	Through interaction with the C-LEIA, I could enhance my ability to ask closed-ended questions, which are crucial for clarifying specific points and ensuring accuracy in understanding client requirements.
Q_{12}*	Interacting with the C-LEIA could help me improve in identifying and managing different stakeholders, each with their own needs, priorities, and influence on the project.
Q_{13}	I felt engaged with the task.
Q_{14}	The interaction with the tool used for gathering requirements (C-LEIA or Transcript) was motivating.
Q_{15}	The tool used to elicit requirements information kept me interested in the activity.

Table 4.1: Likert scale survey questions (*=optional). The online form also solicited free-text comments. Q_{5-7} were only given to the experimental group and Q_{14-15} were answered by the control group with respect to the transcript and by the experimental group with respect to the C-LEIA session.

4.2 Results

We use Likert survey responses, preference data, and computed metrics to address our research questions. Responses to each Likert survey question are presented in Figure 4.1, with additional metrics in Table 4.2. In this section, we summarize each high-level finding in ***bold italic***, followed by a discussion of the supporting data.

Was the Exercise Valid? (Q_{1-4})

Participants generally understood the tasks, found them to be of suitable difficulty, and felt they had sufficient time to complete them. A majority agree or strongly agree that the time allocated was sufficient (Q_1). The perception of having enough time to complete the exercise was slightly better in the experimental group than in the control group, which is interesting for two reasons. First, interacting with an AI would likely take more time than working from transcripts: students must think of questions, type them

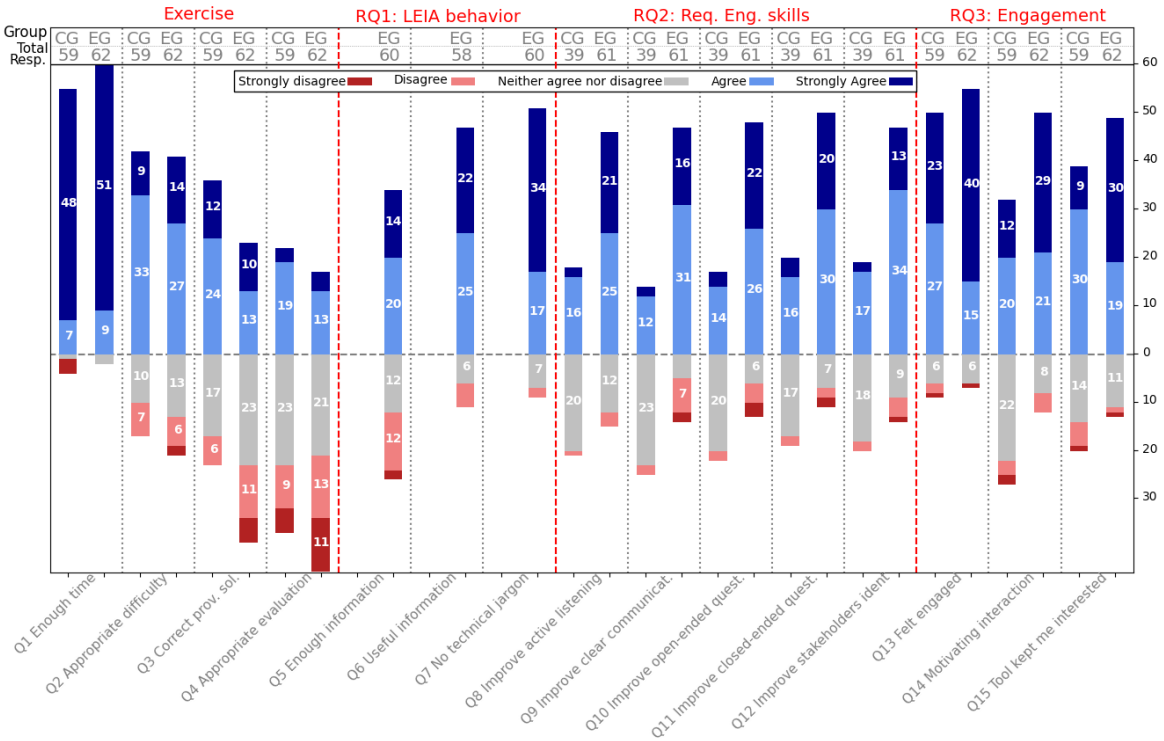


Figure 4.1: Likert-scale responses to survey questions in Table 4.1, grouped by Research Question relevance. As questions are optional, the total height (total number of responses) is not equal for all questions. In particular, Q_{8-12} had a particularly low response rate in the control group.

Metric (units)	Group	Mean	Std	Max	Median	Min
Duration (hours)	CG	0.86	0.19	1.34	0.84	0.38
	EG	0.85	0.23	1.39	0.81	0.42
Time Between Interactions (seconds)	EG	103.48	36.84	196.38	91.71	45.83
C-LEIA Response Length (chars)	EG	236.24	78.35	538.19	236.67	90.78
Participant Question Length (chars)	EG	74.44	26.26	178.65	72.50	30.30
Attributes in Solution (count)	CG	28.32	8.83	62	27	12
	EG	15.30	4.97	35	14	6
Classes in Solution (count)	CG	10.61	2.67	18	10	6
	EG	7.68	1.57	12	8	4

Table 4.2: A summary of experiment measurements, including total exercise duration, interaction with the C-LEIA, including response time, length of questions and answers, and student solution size in number of classes and attributes.

in, and then read the answers to design the model, while students using the transcription only have to read questions and answers. Second, the actual time spent (hours) was comparable between the CG (min/max/μ/σ = 0.38/1.34/0.86/0.19) and the EG (0.42/1.39/0.85/0.42).

Figure 4.2 shows the histograms of the durations of the exercise for both groups. A relatively low Earth Mover's Distance [9] of 0.3708 suggests the distribution shapes are similar, though a Sum of Absolute Differences [9] of 26.2246 suggests distinct patterns in the groups' duration distributions, with wider dispersion in the experimental group and a larger value for the mean in the control group. A somewhat large Kullback-Leibler divergence of 3.1817 is partly due to an outlier in the control group with a very small time. We note that KL divergence is sensitive to outliers.

A majority agreed that the difficulty level was appropriate (Q_2). In both groups, the

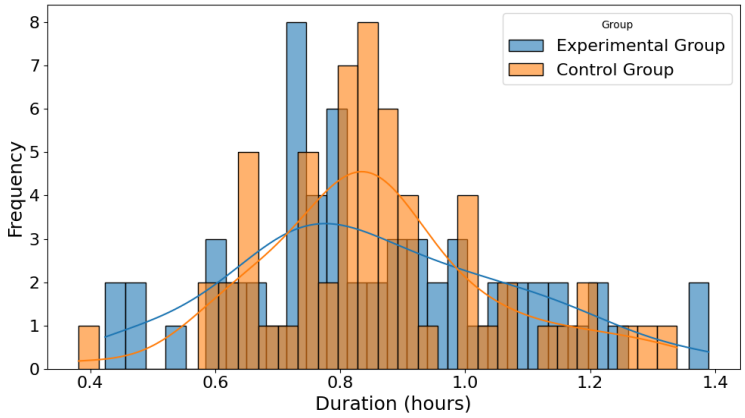


Figure 4.2: Histograms of exercise duration for both groups

evaluation of participants' solutions (Q_3, Q_4) appears to have been a source of frustration, particularly due to limitations in the comparison methodology and the simplistic calculation of scores assigned to the models. We note that in our study, student scores were not used to validate the research questions, since all of the students' solutions were "above threshold" of correctness for demonstrating an understanding of the material, nor were they a factor in the amount of extra credit granted to study participants. For these reasons, we do not consider these negative responses a threat to validity and felt confident in interviewing participants about other aspects of the study experience. It is also worth noting that, although the mean of the number of classes in the solutions provided by participants in the CG (10.61) is significantly higher than those provided by those provided by participants of the EG (7.68), the variability (std. deviation) is also much higher (2.67 vs 1.57).

RQ 1: C-LEIA behavior (Q_{5-7})

Student responses suggest positive results for our first two research questions (RQ 1.1 and RQ 1.2). In general, the responses reveal an overwhelming positive experience with regard to the behavior of the C-LEIA, with the majority of students explicitly stating that no erratic, inappropriate, offensive or incoherent behaviors were observed during the session and that the C-LEIA facilitated the elicitation process without significant issues.

Students believed that the C-LEIA omitted technical terminology in its responses. 85% of students in the EG indicated that the C-LEIA avoided technical jargon (Q_7), with only 2 students disagreeing. This overwhelming positive response supports the claim that we succeeded in conditioning an LLM to omit technical details in its description of the software project (RQ 1.1).

Students generally perceived C-LEIA responses to contain relevant and sufficient information to construct a design. Indeed, 81% of students agreed that the C-LEIA provided *useful* information (while 5 students disagreed) (Q_5). At the same time, only 57% of students indicated that the C-LEIA provided *enough* information to complete their design (Q_6). While this is indeed lower, the discrepancy might be explained by the interactivity of the exercise. The transcripts given to the CG contained complete, detailed information, whereas the EG must explicitly elicit this information during the interview. In fact, in professional practice, it is common to request follow-up interviews as a result of trying to create a design with what is later discovered to be incomplete information.

RQ 2: Developing Interview Skills (Q_{8-12})

Interviewing is a distinct skill exercised by the C-LEIA and not by the transcript exercise. While all solutions were sufficiently correct, the CG's class diagrams tended to include more classes ($\min/\max/\mu/\sigma = 6/18/10.61/2.67$) than the EG's solutions ($4/12/7.68/1.57$). Similarly, the total number of class attributes across all classes in

the solution was higher for the CG (min/max/μ/σ = 12/62/28.32/8.83) than for the EG (6/35/15.30/4.97). We believe this minor difference is because EG participants must construct probing questions at the same time as they build a conceptual model, perhaps causing them to consolidate requirements more than their CG counterparts.

Compared to CG participants, EG participants more strongly agree that the exercise helps them improve key ReqEl interview skills, including stakeholder identification, question formulation, clear communication, and active listening. The response rate for these questions was much lower in the CG than in the EG, possibly suggesting that the CG participants felt less motivated or less sufficiently informed to give their opinion on these questions. We believe that this superiority reflects the interactive and adaptive capabilities of C-LEIA, which provided a more challenging, active, and personalized learning experience compared to static transcripts. Indeed, P01 described the C-LEIA exercise as "more realistic and [it] prepares us better for the future."

RQ 3: Engagement (Q_{13-15})

The C-LEIA approach is more consistently able to maintain interest and enhance motivation. 80% of the EG vs. 54% of the CG found their respective exercise motivating (Q_{14}) and 79% of the EG vs. 66% of the CG felt that their exercise retained their interest (Q_15). While some of this superiority can be attributed to the novelty of the approach (despite our attempts to mitigate this through the habituation phase), some is probably due to the interactive and adaptive nature of the C-LEIA. By contrast, the transcript-based approach was less consistent in its ability to engage participants, as evidenced by the higher presence of neutral and disagreement responses. To this point, EG participants found it highly valuable to ask a virtually unlimited number of questions to explore requirements (P48: "The AI allows you to ask more questions than you have in the transcript to clarify doubts"), something seen as a key C-LEIA strength that led to deeper engagement with the task.

An overwhelming majority of participants preferred the C-LEIA over transcripts, with stronger preference among those who used it in the treatment phase. 61% of the CG and 85% of the EG indicated their preference for the C-LEIA exercise over transcripts. This preference could be attributed to the inherent advantages of interactive participation and a dynamic and practical experience that mirrors real-world scenarios. The EG's higher margin of preference is compatible with the conclusion that the C-LEIA not only aligns more closely with skill development objectives, but also fosters greater engagement and perceived value among participants.

Chapter 5

Limitations and Future Work

In our study presented in Chapter 4, some participants identified weaknesses in our design or instances of less-than-optimal behavior of the C-LEIA. Similarly, we summarize themes in participant perceptions of our evaluation process that highlight deficiencies in accuracy or reproducibility. We summarize a few recurring themes with illustrative quotes from survey responses, using these along with directly observed limitations as a way to suggest future work.

5.1 Limitations of C-LEIA

Occasional imprecise or less-than-relevant answers to nuanced questions. While the C-LEIA generally performed well, participants reported that it occasionally struggled to provide precise or relevant answers, especially when questions were very specific or nuanced. P63 highlighted this, placing "the main limitation [to be] the precision of the answers." Some participants noted that the AI would sometimes repeat responses or provide ambiguous answers, which could disrupt the flow of the elicitation process. Indeed, "if you are too specific, you don't get enough information. If you are too broad, it feels like you're trying not to 'break' [the C-LEIA]" (P88). Furthermore, the C-LEIA demonstrated a reliance on carefully structured questions, often failing to provide additional information unless explicitly asked, or fixating on certain parts of a project while neglecting others. These uncontrolled idiosyncrasies could limit the flow of requirements discovery or create gaps in understanding the overall scope of the application.

Inconsistent/contradictory information given. The C-LEIA had a tendency to contradict itself by, for example, initially suggesting "two entities need to be related and later states that it's not necessary" (P27). Similarly, the C-LEIA occasionally introduced new attributes, properties, or requirements as the session went on, which were not part of the solution. Both kinds of inconsistencies created confusion for participants. While fluctuation of requirements is common with real customers, we would like to carefully control this to

improve pedagogical control and reduce student surprises. We anticipate that more extensive LLM alignment will mitigate this problem.

Inauthenticity of the dialogue. While the C-LEIA came close to mimicking the response style of real clients, many respondents felt that it lacked the subtlety and depth of a human interview, particularly in its inability to convey non-verbal cues or contextual hints in the same way that embodied exercises such as RoboREIT [14] might support. P67 explicitly claimed, "It doesn't feel real, although it was close enough." This artificiality seems to have created a sense of detachment, which limited its effectiveness in fully simulating real-world scenarios. Indeed, "if you ask it something you've already asked, it responds in the same way" (P62). Although interacting with a chatbot is not equivalent to interviewing a real person, the C-LEIA attempts to account for some factors, such as domain and persona alignment. However, in its current state C-LEIAs cannot fully replicate the complexity of a real interview, where requirements engineers must navigate nuances such as tone and gestural or postural language, which usually provide an essential context to the interactions and hint important information about the requirements.

Centralized deployment. The service architecture to author, serve, and manage C-LEIAs as described in Section 3.2 is highly reliant on centralized hosting, where a single provider (e.g. Heroku, MongoDB) hosts all data, even potentially between institutions. This is unfavorable for many institutions that place tight restrictions on the storage and use of student data. Indeed, these institutions would require an IT team (or an individual instructor) to deploy *their own* trio of containerized web services and MongoDB instance *before* authoring the first exercise. For most institutions, this is impractical and thus would require additional upstream engineering in the service-level design to simplify deployment and management.

We claim that despite these shortcomings, C-LEIAs can aid students in learning how to formulate appropriate questions, structure conversations, understand when deeper and more insightful questions are required, or prepare for difficult situations when dealing with clients. In addition, C-LEIAs could serve as a starting point for students, allowing them to gain confidence and practice before participating in real-world interviews.

5.2 Limitations in Evaluation

Evaluation of student work products. Although one motivation for having participants submit a machine-readable rather than hand-drawn class diagram was to allow for the possibility of automated grading, the automatic grading scheme used in this study to compare the solutions of the participants with the reference solution was simplistic. While it was sufficient to ensure that participants' solutions were substantially correct and avoid significant threats to the study's validity, participants felt less positive about this aspect of the

study than any other (Q_{3-4}). P15 found that "the evaluation [did] not give enough flexibility. There are multiple correct ways to model the exercise." We see an exciting opportunity for future work to develop more sophisticated automatic evaluation of these designs, possibly including other design deliverables such as Class-Responsibility-Collaborator (CRC) cards and user stories [12].

Insufficient background information made it challenging to conduct the interview. Participants in both groups found that the lack of initial information about the system hindered their ability to fully explore the requirements or develop diagrams effectively. This sentiment was more common in the experimental group: "Provide some context about the system to be developed, for anyone who is going to use [the C-LEIA]. I think it would make the exercise a bit easier and help us ask better questions" (P42). Starting the interview process without guidance or contextual cues was challenging for some participants, particularly those without prior interview experience. In contrast, the CG participants only needed to passively read a transcript rather than initiate and sustain a conversation. An immediate mitigation is to include a basic background document that students could read before beginning the interview or reading the transcript. An alternative future avenue is a complementary T-LEIA—a "team whisperer" or coach, independent of the C-LEIA, that can give less-experienced students proactive guidance on initiating and conducting the interview by helping to formulate guiding questions, ensuring the discussion stays on track, and so on.

Chapter 6

Conclusion

Through C-LEIA, we revisit the approach of providing students a simple chatbot to role play a client-facing interview. We condition the C-LEIA with common and injected prompts to be (1) believable as a client, (2) resistant to simple jailbreaks that can be conducted in a classroom, (3) specific enough for students to glean useful information, and (4) non-technical enough to enable student practice. Our implementation of the surrounding infrastructure leverages OpenAI's Assistants API to produce LLM responses, Heroku to host web services, MongoDB Atlas as a persistence layer, VueJS to construct a user interface, ExpressJS to handle webserver mechanics, Ace Editor to provide an text editing environment, and MermaidJS to render student produced UML class diagrams.

Despite using a prototype whose limitations were noted by study participants, our initial study yielded promising results. The C-LEIA provides a sufficiently sound and interview-able simulacrum of a non-technical client representing a small business, providing enough information to produce a viable design without directly disclosing the technical details that underpin the design. Time on task using the C-LEIA is comparable to that for using transcripts, yet the C-LEIA appears to foster better task engagement and motivation. Participants using the C-LEIA agree more strongly than those using transcripts that the exercises are helping them improve key ReqEl skills, and in particular, their solutions compared to a control group suggest that interview skills are specifically being exercised. These findings suggest that AI-driven tools hold significant potential for enhancing user engagement in tasks that require sustained attention and active participation like those addressed in the exercise.

At the same time, imprecise, irrelevant, contradictory, or inauthentic responses from the C-LEIA may lead to detachment, frustration, or confusion in students, suggesting multiple avenues for improvement.

The accessibility and convenience of LLMs already provides a promising way to practice requirements elicitation without the logistical complexities of arranging live interviews. We believe that refining our prototype and combining it with more sophisticated automatic grading, will lead to a novel, realistic, and scalable way to teach ReqEl interview skills.

Bibliography

[1] Deniz Akdur. "Analysis of Software Engineering Skills Gap in the Industry". In: *ACM Transactions on Computing Education* 23 (2022), pp. 1–28. DOI: 10.1145/3567837.

[2] Cem Anil et al. "Many-shot Jailbreaking". In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 129696–129742. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/ea456e232efb72d261715e33ce25f208-Paper-Conference.pdf.

[3] Anishka et al. *Can ChatGPT Play the Role of a Teaching Assistant in an Introductory Programming Course?* arXiv:2312.07343 [cs]. Jan. 2024. DOI: 10.48550/arXiv.2312.07343. URL: http://arxiv.org/abs/2312.07343.

[4] Muneera Bano et al. "Teaching requirements elicitation interviews: an empirical study of learning from mistakes". In: *Requir. Eng.* 24.3 (Sept. 2019), pp. 259–289. ISSN: 0947-3602. DOI: 10.1007/s00766-019-00313-0. URL: https://doi.org/10.1007/s00766-019-00313-0.

[5] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[6] Marian Daun, Alicia M. Grubb, and Bastian Tenbergen. "A Survey of Instructional Approaches in the Requirements Engineering Education Literature". In: *2021 IEEE 29th International Requirements Engineering Conference (RE)*. 2021, pp. 257–268. DOI: 10.1109/RE51729.2021.00030.

[7] Sourav Debnath and Paola Spoletini. "Designing a virtual client for requirements elicitation interviews". In: *Requirements Engineering: Foundation for Software Quality: 26th International Working Conference, REFSQ 2020, Pisa, Italy, March 24–27, 2020, Proceedings 26*. Springer. 2020, pp. 160–166. DOI: 10.1007/978-3-030-44429-7_12.

[8] Paul Denny et al. "Prompt Problems: A New Programming Exercise for the Generative AI Era". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. New York, NY, USA: Association for Computing Machinery, Mar. 2024, pp. 296–302. ISBN: 9798400704239. DOI: 10.1145/3626252.3630909. URL: https://dl.acm.org/doi/10.1145/3626252.3630909.

[9] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2016. ISBN: 978-3-662-52843-3. DOI: 10.1007/978-3-662-52844-0.

[10] Beatrice Donati et al. "Common Mistakes of Student Analysts in Requirements Elicitation Interviews". In: vol. 10153. Feb. 2017, pp. 148–164. ISBN: 978-3-319-54044-3. DOI: 10.1007/978-3-319-54045-0_11.

[11] R. Dubey and V. Tiwari. "Operationalisation of soft skill attributes and determining the existing gap in novice ICT professionals". In: *Int. J. Inf. Manag.* 50 (2020), pp. 375–386. DOI: 10.1016/j.ijinfomgt.2019.09.006.

[12] Armando Fox and David Patterson. *Engineering Software as a Service: An Agile Approach Using Cloud Computing*. 2nd Edition. San Francisco, CA: Strawberry Canyon LLC, 2022.

[13] Ilyuza Gizzatullina. "Empirical study of customer communication problem in agile requirements engineering". In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2019)*. ACM, 2019, pp. 1262–1264. DOI: 10.1145/3338906.3342511. URL: https://doi.org/10.1145/3338906.3342511.

[14] Binnur Görer and Fatma Başak Aydemir. "RoboREIT: an Interactive Robotic Tutor with Instructive Feedback Component for Requirements Elicitation Interview Training". In: *arXiv preprint arXiv:2304.07538* (2023). URL: https://arxiv.org/pdf/2304.07538.

[15] Michael Guevarra et al. "An LLM-Guided Tutoring System for Social Skills Training". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 39.2828 (Apr. 2025), pp. 29643–29645. ISSN: 2374-3468. DOI: 10.1609/aaai.v39i28.35353.

[16] Jordan Hoffmann et al. "An empirical analysis of compute-optimal large language model training". en. In: *Advances in Neural Information Processing Systems*. Vol. 35. Dec. 2022, pp. 30016–30030.

[17] I. Inayat et al. "A systematic literature review on agile requirements engineering practices and challenges". In: *Computers in Human Behavior* 51 (2015), pp. 915–929. DOI: 10.1016/j.chb.2014.10.046. URL: https://doi.org/10.1016/j.chb.2014.10.046.

[18] Hyoungwook Jin et al. "Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education". In: *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. CHI '24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 1–28. ISBN: 9798400703300. DOI: 10.1145/3613904.3642349. URL: https://dl.acm.org/doi/10.1145/3613904.3642349.

[19] Mollie Jordan, Kevin Ly, and Adalbert Gerald Soosai Raj. "Need a Programming Exercise Generated in Your Native Language? ChatGPT's Got Your Back: Automatic Generation of Non-English Programming Exercises Using OpenAI GPT-3.5". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. New York, NY, USA, Mar. 2024, pp. 618–624. ISBN: 9798400704239. DOI: 10.1145/3626252.3630897. URL: https://dl.acm.org/doi/10.1145/3626252.3630897.

[20] Breanna Jury et al. "Evaluating LLM-generated Worked Examples in an Introductory Programming Course". In: *Proceedings of the 26th Australasian Computing Education Conference*. ACE '24. New York, NY, USA, Jan. 2024, pp. 77–86. ISBN: 9798400716195. DOI: 10.1145/3636243.3636252. URL: https://dl.acm.org/doi/10.1145/3636243.3636252.

[21] Majeed Kazemitabaar et al. "CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs". In: *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. CHI '24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 1–20. ISBN: 9798400703300. DOI: 10.1145/3613904.3642773. URL: https://dl.acm.org/doi/10.1145/3613904.3642773.

[22] M. Korkala, M. Pikkarainen, and K. Conboy. "Distributed Agile Development: A Case Study of Customer Communication Challenges". In: *Agile Processes in Software Engineering and Extreme Programming (XP 2009)*. Ed. by P. Abrahamsson, M. Marchesi, and F. Maurer. Vol. 31. Lecture Notes in Business Information Processing. Springer, Berlin, Heidelberg, 2009, pp. 161–167. DOI: 10.1007/978-3-642-01853-4_21. URL: https://doi.org/10.1007/978-3-642-01853-4_21.

[23] Charles Koutcheme. "Training Language Models for Programming Feedback Using Automated Repair Tools". In: *Artificial Intelligence in Education: 24th International Conference, AIED 2023, Tokyo, Japan, July 3–7, 2023, Proceedings*. Berlin, Heidelberg: Springer-Verlag, July 2023, pp. 830–835. ISBN: 978-3-031-36271-2. DOI: 10.1007/978-3-031-36272-9_79. URL: https://doi.org/10.1007/978-3-031-36272-9_79.

[24] Nischal Ashok Kumar and Andrew Lan. *Using Large Language Models for Student-Code Guided Test Case Generation in Computer Science Education*. arXiv:2402.07081 [cs]. Feb. 2024. DOI: 10.48550/arXiv.2402.07081. URL: http://arxiv.org/abs/2402.07081.

[25] Andre W. Kushniruk and Vimla L. Patel. "Cognitive and usability engineering methods for the evaluation of clinical information systems". In: *Journal of Biomedical Informatics* 37.1 (Feb. 2004), pp. 56–76. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2004.01.003. URL: https://www.sciencedirect.com/science/article/pii/S1532046404000206 (visited on 04/08/2025).

[26] Muhammad Laiq and Oscar Dieste. "Chatbot-based Interview Simulator: A Feasible Approach to Train Novice Requirements Engineers". In: *2020 10th International Workshop on Requirements Engineering Education and Training (REET)*. 2020, pp. 1–8. DOI: 10.1109/REET51203.2020.00007.

[27] H Chad Lane et al. "Intelligent tutoring for interpersonal and intercultural skills". In: *Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*. Vol. 111. 2007.

[28] Mengqi Liu and Faten M'Hiri. "Beyond Traditional Teaching: Large Language Models as Simulated Teaching Assistants in Computer Science". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. New York, NY, USA: Association for Computing Machinery, Mar. 2024, pp. 743–749. ISBN: 9798400704239. DOI: 10.1145/3626252.3630789. URL: https://dl.acm.org/doi/10.1145/3626252.3630789.

[29] T. El-Najar, I. Ahmad, and M. Alkandari. "Client Communication: A Major Issue in Agile Development". In: *International Journal of Software Engineering and its Applications* 10 (2016), pp. 113–130. DOI: 10.14257/IJSEIA.2016.10.12.10. URL: https://doi.org/10.14257/IJSEIA.2016.10.12.10.

[30] OpenAI et al. *GPT-4o System Card*. 2024. arXiv: 2410.21276 [cs.CL]. URL: https://arxiv.org/abs/2410.21276.

[31] Long Ouyang et al. "Training language models to follow instructions with human feedback". en. In: *Advances in Neural Information Processing Systems*. Vol. 35. Dec. 2022, pp. 27730–27744.

[32] Nishat Raihan et al. "Large Language Models in Computer Science Education: A Systematic Literature Review". In: *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSETS 2025. New York, NY, USA: Association for Computing Machinery, Feb. 2025, pp. 938–944. ISBN: 9798400705311. DOI: 10.1145/3641554.3701863. URL: https://dl.acm.org/doi/10.1145/3641554.3701863.

[33] Lianne Roest, Hieke Keuning, and Johan Jeuring. "Next-Step Hint Generation for Introductory Programming Using Large Language Models". In: *Proceedings of the 26th Australasian Computing Education Conference*. ACE '24. New York, NY, USA: Association for Computing Machinery, Jan. 2024, pp. 144–153. ISBN: 9798400716195. DOI: 10.1145/3636243.3636259. URL: https://dl.acm.org/doi/10.1145/3636243.3636259.

[34] Jaromir Savelka et al. *Efficient Classification of Student Help Requests in Programming Courses Using Large Language Models*. arXiv:2310.20105 [cs]. Oct. 2023. DOI: 10.48550/arXiv.2310.20105. URL: http://arxiv.org/abs/2310.20105.

[35] David H. Smith and Craig Zilles. "Code Generation Based Grading: Evaluating an Auto-grading Mechanism for "Explain-in-Plain-English" Questions". In: *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*. ITiCSE 2024. New York, NY, USA: Association for Computing Machinery, July 2024, pp. 171–177. ISBN: 9798400706004. DOI: 10.1145/3649217.3653582. URL: https://dl.acm.org/doi/10.1145/3649217.3653582.

[36] Ashish Vaswani et al. "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964. URL: https://dl.acm.org/doi/10.5555/3295222.3295349.

[37] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS '22. New Orleans, LA, USA: Curran Associates Inc., 2022. ISBN: 9781713871088.

[38] Diyi Yang et al. *Social Skill Training with Large Language Models*. 2024. arXiv: 2404.04204 [cs.CL]. URL: https://arxiv.org/abs/2404.04204.

[39] Dongdong Zhang et al. "Assistant Teaching System for Computer Hardware Courses Based on Large Language Model". en. In: *Computer Science and Education. Computer Science and Technology*. Ed. by Wenxing Hong and Geetha Kanaparan. Singapore: Springer Nature, 2024, pp. 301–313. ISBN: 978-981-9707-30-0. DOI: 10.1007/978-981-97-0730-0_27.

[40] Didar Zowghi and Chad Coulin. "Requirements Elicitation: A Survey of Techniques, Approaches, and Tools". In: *Engineering and Managing Software Requirements*. Ed. by Aybüke Aurum and Claes Wohlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 19–46. ISBN: 978-3-540-28244-0. DOI: 10.1007/3-540-28244-0_2. URL: https://doi.org/10.1007/3-540-28244-0_2.