Exploring the Potential of Large Language Models in Supporting Domain Model Derivation from Requirements Elicitation Conversations

Sander van Nifterik

A master thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
with the
Master Business Informatics
of
Utrecht University.

Department of Information and Computing Sciences

July 18, 2024

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

ChatGPT will only be used clarify concepts unknown to the author, or to help rewrite an original sentence. It’s output will in not be used verbatim.

Signed:

Date: July 18, 2024

Abstract

In the last few years, digital meetings to elicit requirements for a system-to-be have become commonplace. This allows for collecting data about the system-to-be; sometimes in the form of conversation transcripts. After such conversation have taken place, domain models are created in order to represent the domain in which the system-to-be will operate. This thesis aims to semi-automate this process by exploring the potential of Large Language Models (LLMs) to support the domain model derivation task from requirements elicitation conversations.

To evaluate effectiveness, we devised metrics for comparing domain models created by a human with domain models created by an LLM. The results indicate that LLMs can aid in the creation of domain models, showing high levels of agreement with human-generated models regarding entities, attributes, and relationships. However, LLMs also produced several unusable elements, highlighting the necessity of human oversight.

The study concludes that LLMs offer substantial promise in enhancing domain modeling efficiency, but require careful integration to mitigate errors. Key limitations include the rapid evolution of LLM technology and the specific contexts of the datasets used. Future work should focus on including more than one human modeler, or including people from the industry. Recommendations include developing best practices for LLM use in the field of requirements engineering, and exploring the balance between automated assistance and human expertise.

Acknowledgements

I would like to thank my first supervisor Fabiano Dalpiaz for all the support throughout this project. I really appreciated the in which he supervised the project. This has really helped me a lot.

Furthermore, I would like to thank Tjerk Spijkman. Although not an official supervisor, his input has been greatly appreciated.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The process of designing software systems typically starts with having conversations to elicit requirements. Requirements elicitation is concerned with the activities of seeking, uncovering, acquiring, and elaborating requirements [1]. Domain modelling is another initial activity that takes place early on in this process. It can be done without input requirements, for example as a synthesis of a requirements elicitation conversation, based on a initial set of requirements. Domain modelling helps getting to a more concrete overview of the domain/environment in which the system will operate [2].

The extraction of useful information from requirements elicitation conversations is a time consuming task. This task relies almost completely on the notes and memory of the analyst(s) present at these conversations. This research aims to identify if such Large Language Models (LLMs) form an opportunity to support the derivation of domain models from requirement elicitation conversations transcripts.

This chapter will introduce the research topic by discussing first the background and context, followed by the research problem, aims, and objectives. Thereafter, the significance and limitations will be discussed.

1.1 Background

1.1.1 Conversational requirements engineering

The research that is conducted in this thesis is closely related to a larger research effort: the PhD research that is being conducted by Tjerk Spijkman. Relevant to this research is the focus on conversational RE. The aim of conversational RE part of his research is to try and build an understanding of the processes that are conducted by humans, in order to guide possible automation and tool supported designs. Examples of such processes are domain modelling, process modelling, or conducting elicitation interviews. The first paper on this topic provided an empirical understanding of the fit-gap analysis elicitation technique [3]. Through the analysis of transcripts of requirements elicitation conversations, they defined requirements categories, each with their own keywords and phrases. This set of keywords and phrases is offered as a potential starting for the possible automation of identifying such categories. The second paper focused on the development of Trace2Conv, a (prototype) tool that used Natural Language Processing (NLP) to identify concepts in transcripts [4]. The third paper focuses on pre-requirements specification traceability. It introduced a NLP tool that allows for tracing certain user story requirements back to the relevant speaker turns in the transcript [5]. Lastly, the fourth paper focuses on the summarizing requirement elicitation conversations [6]. They have created a prototype which summarizes transcribed requirements elicitation conversations, using NLP. An overview of the research on conversational RE, and how this research relates to the thesis, can be found in Figure 1.1. In the visualization, it is illustrated how all the research starts with requirement elicitation conversations. Next, the general direction of the paper is shown, followed by the output of paper. Lastly, the general outcome of the paper is mentioned.

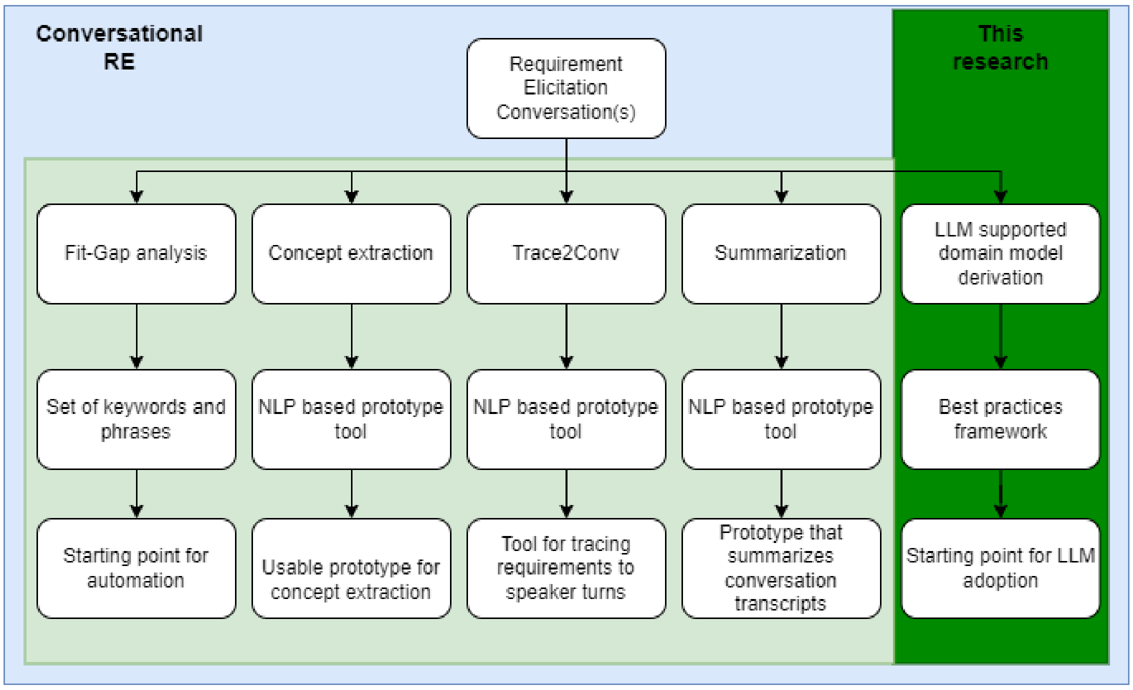

Conversational RE
This research
Requirement Elicitation Conversation(s)
Fit-Gap analysis
Concept extraction
Trace2Conv
Summarization
LLM supported domain model derivation
Set of keywords and phrases
NLP based prototype tool
NLP based prototype tool
NLP based prototype tool
Best practices framework
Starting point for automation
Usable prototype for concept extraction
Tool for tracing requirements to speaker turns
Prototype that summarizes conversation transcripts
Starting point for LLM adoption

Figure 1.1: An overview of conversational RE research.

1.1.2 Domain modelling in RE: An Overview

To understand the relevance of this research, it is important to highlight the relevance of domain modelling in the RE process. In earlier research, several studies focused on the relationship between requirements engineering and conceptual modelling [7]–[10]. An example is the use of a requirements engineering based conceptual modelling approach to improve the software production process [7]. Another example looks into extending standard conceptual model elements, such as data, process, event, with extra elements [8]. These extensions can be used to create enriched conceptual models.

Later, research efforts were focused on the relationship between requirements engineering and the agile way of working [11]–[14]. An example of this is research that aimed to enhance the body of knowledge in software development by evaluating the engagement of stakeholders and users in Agile Requirements Engineering (RE) [11]. The study offered methods to apply agile software development with a human centered focus. Furthermore, it presented an overview on how to do requirements management within agile software development. De Lucia et al. examined the challenges that can be encountered in agile requirements engineering [13]. It is a systematic literature review that looked into the requirements engineering practices in agile software development, and the challenges that agile team members face when working on requirements engineering activities.

More recently, research efforts are more focused on automation, traceability, and NLP [15]–[17]. For example, using a LLM to check for requirement completeness [17] or using AI in order to trace back domain modelling decisions to problem descriptions, or vice versa [16].

Overall, it becomes clear that the focus of current research efforts into domain modelling for RE, are now looking into AI supported automation.

1.1.3 This research

As mentioned before, extracting important information from requirements elicitation conversations is a time-consuming activity. Imagine having to work through all the physical contents produced in a meeting, for example handwritten notes, flip-over papers, or lo-fi prototypes. More recently, however, virtual meetings have become more and more commonplace [18]. During the recent COVID-19 pandemic, virtual meetings have even become mainstream [19]. These digital meetings, for example via Microsoft Teams, can be recorded and transcribed automatically, and with good accuracy [20], [21]. This has made it easier to keep data on these conversations without effort. With the emergence of LLMs, processing of text, or leveraging a model to generate text, has become a bigger research interest. Researchers from all kinds of domains, such as education, health care, real-estate, and software production, are studying the potential applications and the effects of LLMs in their field [22]–[25]. The same goes for the field of RE [17], [26], [27].

1.2 Research gap

As established in the background, research into LLMs is very active at the moment. However, since research into LLMs is relatively new, there is still a lot left to explore. This is also true for LLMs in conjunction with requirements engineering, as will become clear in this section. Examples of current research, and their shortcomings, can be found taking different kinds of perspectives.

An example of this is letting a LLM deal with incompleteness of requirements [26]. In this study, a LLM is being trained to recognize incompleteness in requirements by means of a masked-word prediction task. The incompleteness of a requirement was simulated by leaving out words, instead of analyzing human made mistakes. According to the researchers, this limited their ability to generalize the findings to a more practical application. Furthermore, it was stated that more research should be done in order to test their approach. This example clearly shows that there is still a need for validation of LLM approaches in a real world setting with human participants, and that the practical implications of LLM in RE is not established on a large scale.

Another perspective that is currently looked into is prompt patterns. A study was done to try and identify prompt patterns for software engineering (SE) tasks, such as: improving code quality, refactoring, requirements elicitation, and software design [27]. The authors identified various prompt patterns and the format in which these were presented. One of the two primary conclusions of the study was the big potential LLMs have in the SE life-cycle. According to the authors, the potential of LLMs is not widely understood/appreciated. According to them, LLMs hold an immense potential for automating common tasks in the SE life-cycle. Furthermore, they state that, although such patterns are helpful, strong human involvement is still required in order to leverage LLMs effectively. This example validates the relevancy and potential that LLMs have, in the field of requirements engineering. Furthermore, it emphasizes the point that the full potential of LLMs in RE/SE, is yet to be uncovered.

Research akin to the topic of this thesis has explored the extraction of domain models from textual requirements [17]. In this work, Arulmohan *et al.* compared three approaches: Conditional Random Field (CRF), GPT 3.5, and Visual Narrator, on how well they were able to extract domain concepts from agile product backlogs. Next to their more technical findings on the performance of each of the approaches, the relevance of further research became clear from their concluding statements. In their future directions, they suggest to look into which type of solution to choose, e.g. a cheaper and easier to adopt of-the-shelf solution, or a more expensive tailor made solution that takes a longer time to build. Even though this example is closely related to the topic of this thesis, it gets a clear message across: this research direction is still open ended, and there is still a lot left to discover. Once again, it demonstrates a need for researching the practical implications of using LLMs in RE. Specifically, identifying important considerations in the decision making process, of deciding if and how to implement a LLM.

To summarize, there is a clear research gap in the current literature, that justifies further research on the topic. The high-level issues that are prevalent in current research, revolve around two things: the rapid developments in LLM techniques, and the difficulties that arise when trying to integrate research into practice. Because of the potential that LLMs hold in RE/SE, it is important to get a better understanding of the possibilities for implementation into practice.

1.3 Aims and objectives

Given the rapid development of LLM techniques, and the potential they seem to hold for the field of RE, this thesis aims to explore the potentials of LLMs when extracting domain models from requirements elicitation conversations.

1.3.1 Research questions

In order to try and contribute to closing the gap in the current research, the following main research question has been established.

MRQ: "What is the potential of Large Language Models in supporting the derivation of domain models from requirements elicitation conversations, and what factors influence their effectiveness in this context?"

To help answer the main research question, the following sub-research questions have been established.

- **SQ1:** What are the best practices for designing an LLM-based approach that supports domain model derivation from requirement elicitation conversation transcripts?

- **SQ2:** What is the effectiveness of the LLM compared to the human, when deriving domain models from requirement elicitation conversation transcripts?

- **SQ3:** What is the similarity between the LLM and the human, when deriving domain models from requirement elicitation conversation transcripts?

In the research questions above, some terminology is used that needs some further elaboration. In SQ2 effectiveness is mentioned. Effectiveness means the production of modelling elements. Both the human and the LLM will create modelling elements that can either be usable or not usable. The creation of those elements is what is captured in the term effectiveness. In SQ3 similarity is mentioned. Similarity means the extent to which the human and the LLM agree on modelling decisions. By comparing the effectiveness and the similarity, between the human and the LLM, it will become clear what the potential of the LLM is; whether or not adopting a LLM supported approach makes sense or not.

By identifying best practices for designing a LLM-based approach to support the domain model derivation task, it might become clear what the factors are that may influence the performance of the LLM. This could influence the extent to which the full potential of the LLM can be unlocked.

Furthermore, having a good perspective of the effectiveness of the LLM, helps understand the capabilities of the LLM. Understanding this will help get a better idea of the potential of the LLM.

Similarly, comparing the similarity between the human and the LLM, when it comes to performing the domain model derivation task, will help get a better understanding of the potential of the LLM in supporting the domain model derivation task. It will give a good idea of whether or not the LLM is capable to create acceptable domain models, or whether the human needs to intervene more often.

Altogether these sub-questions help answer the main research question.

1.4 Significance

The fast development of LLM techniques and their potential applications in RE/SE is in itself already a reason to pursue this research direction. As highlighted in the research gap section, existing studies on LLMs in RE show that there is still room for further refinement. The significance of this study lies in addressing these gaps, and contributing to the ongoing effort of research in the field. Several key aspects emphasize the importance of this research.

Identifying the gap between research and practice The identified research gap sheds light on the challenge of integrating LLM research findings into practical applications effectively. Despite the promises and potential benefits of LLMs in RE/SE, there remains a need to understand how to translate theoretical developments into real-world solutions. This study aims to identify this gap by exploring the potential that LLMs might have in deriving domain models from requirement elicitation conversation transcripts.

Validating LLM approaches In the research gap section, multiple studies are discussed that are all struggling with identifying the practical implications of their findings. By trying to identify prompt patterns for using LLMs for domain model extraction, the gap between theory and practice could become smaller. If such prompt patterns can be established and tested, it might be possible to generalize those findings to other SE domains.

Decision-making considerations in LLM implementation The complexity of deciding whether and how to implement LLMs in RE is another point of interest this research could help improve. As identified in the research gap, current research is already proposing new research to focus on decision making. This includes identifying factors influencing the choice of LLM solutions, offering insights that can guide organizations in making informed decisions aligned with their specific needs and goals. Being able to contribute to answering this question, by for example coming up with best practices, the decision making process will become easier.

In summary, this research aims to significantly contribute to the ongoing research on LLMs in RE by providing practical insights, validating approaches, and uncovering the full potential of LLMs. By addressing these aspects, the study aims to make way for a more seamless integration of LLM research into practical applications, fostering advancements in the field, and contributing to the broader landscape of requirements engineering.

1.5 Limitations

While this research aims to make meaningful contributions to the understanding and application of LLMs in RE, it is essential to acknowledge certain limitations that may impact the scope and generalizability of the findings.

Rapidly developing nature of LLMs The field of LLMs is characterized by rapid developments, with new techniques and models emerging frequently. The ongoing research efforts, make it so that it could be a challenge to come up with a definitive deliverable. Consequently, the thesis may not be able capture the most recent innovations, potentially limiting the completeness of the findings in regards to the ever-evolving landscape of LLM technologies.

Generalizability The empirical validation of LLM approaches in real-world settings, while great to strive for, may cause challenges in achieving broad generalizability. Factors such as varying datasets, evolving LLM architectures, but primarily diverse application contexts, may impact the extent to which the findings can be applied to different scenarios. It is important to acknowledge the potential limitations in terms of generalizability of specific outcomes. The study aims to explore the practical implications of LLMs in the context of RE. However, the diversity of applications of LLMs within RE could be very broad. They could also be limited on account of organizational practices, or certain confidentiality agreements. Therefore, it could be the case that the findings are context dependent, and that it ends up being uncertain whether or not the findings can be used in other contexts.

Ethical considerations LLMs The ethical implications that come with the use of LLMs, could significantly limit the usefulness of the findings. Despite efforts to address ethical concerns, the study may not be able to take all ethical considerations associated with LLMs into account. Limitations in understanding and mitigating ethical issues may impact the practical recommendations derived from the study. Ethical issues in this case, would be about privacy concerns and dealing with sensitive data.

Resource constraints The scope of this research is bounded by resource constraints, including time and access to specific datasets. While efforts have been made to provide a thorough examination of LLMs in RE, these constraints may limit the depth of analysis in certain aspects. Consequently, the study may not fully explore all potential avenues or delve into exhaustive details in specific areas of interest, due to time constraints.

In conclusion, while this research endeavors to contribute significantly to the discourse on LLMs in RE, it is necessary to recognize and acknowledge possible limitations. By acknowledging these constraints, the study aims to provide a realistic framework for interpreting its findings, encouraging future research to build upon, and address these limitations.

1.6 Outline

In chapter two, a literature study will follow, that functions as the scientific foundation of this research. It will focus on elicitation interviews, large language models, and domain modelling. In chapter three, the research method will be discussed and elaborated upon. It will also go into detail on the design of the research.

In chapter four, the results of the research will be shown. Furthermore, it will give an explanation of the results.

In chapter five, the results are discussed. It will elaborated on the results and the factors that might have affected them.

In chapter six the conclusion follows. It will look back at the research and discuss what can be learned from it.

Chapter 2

Literature review

2.1 Introduction

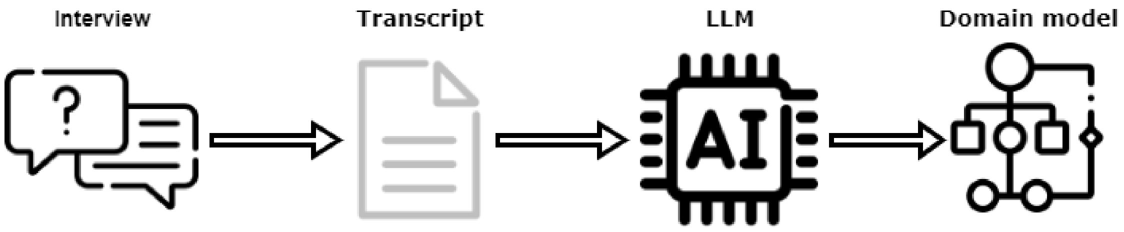


Figure 2.1: An overview of the scope of the thesis, with the three essential parts of the literature review highlighted.

In Figure 2.1, an overview can be found of the primary concepts that are relevant to the research. First, *Interviews* takes place in which a business analyst gathers information about the client company. The function of an interview may be different from interview to interview. The domain model, and its function, could therefore also change depending on the goal of the interview(s). It is therefore important to establish a proper understanding of interviews within RE.

Next, a *Transcript* is created. This transcript captures the information from the interview meeting. This is either done manually, or automated if the interview is recorded. In this research, the assumption is made that such techniques exist. The literature review will therefore not go into detail on how recorded requirements elicitation conversations are transcribed.

The third concept is the LLM. The LLM takes the transcript and uses it to generate the domain model. The literature review will delve into this topic further to get to know the state of art of LLMs in RE better. Furthermore, it will consider important related concepts such as, prompt engineering, types of LLMs, and protocols for using LLMs. Lastly, the literature review will focus on the different domain modelling languages within RE to get an understanding of the differences between them.

To summarize, from the four primary concepts relevant to this thesis, three of them will be elaborated upon in a more detailed manner. This will serve as a common understanding of the concepts throughout the rest of this thesis.

2.2 Procedure

For each of the three primary concepts of this research, as defined in Figure 2.1, literature was gathered. By querying Google Scholar with terms directly relating to the concept, e.g. "requirements elicitation interviews", literature was found. Next, it's relevance was taken into consideration. A paper needed to have a professional appearance, and needed to be cited in other research. Furthermore, the theme of the paper needed to be relevant to the theme of the section it was going to be in. Only papers originally written in English were considered. Furthermore, since requirements elicitation through interviews for example, is not necessarily a new topic, there was not real constraint in place limiting the age of a paper. However, for concept such as LLMs or GPT models, a time constraint was in place. Papers about these topics needed to be from 2018 or later. This is due to the highly volatile nature of such concepts.

For each of the gathered papers, relevant meta-data, such as year, authors, title, context, or quotes, was entered into an excel file. Altogether, this formed a literature catalog that could be called upon when writing the literature review. Furthermore, it helped identify recurring themes, or recurring authors, both of which could be used as a lead to related papers. Lastly, it not only allowed for gathering information quickly, it also helped find other relevant papers through snowballing. All of the reference data for each paper was stored immediately into Overleaf.

2.3 Requirement elicitation interviews

When it comes to software engineering, requirements elicitation is one of the most important activities in the life-cycle [28]. It is also one of the first major activities that takes place when producing software [28]. When eliciting requirements, the focus is on identifying the needs of the stakeholder(s) for the software to be built [1]. This is still a difficult task, since it deals with complex factors such as: the limited memory of the business analyst, the limited capability of the analyst to take notes, or the limited capability of the client to sufficiently express his/her needs [29]. First, the boundaries of what is known, are explored. Second, an analyst needs to deal with people who posses the knowledge. Lastly, the gathered knowledge are turned into requirements, meaning that they need to be organized and managed in an appropriate way.

One of the most common and effective ways to elicit requirements is through interviews [30]. Essentially, there are three primary types of interviews, structured interviews, semi-structured interviews, and unstructured interviews [1].

Structured interviews feature a set of predetermined questions in order to gather specific information. Whether or not such interviews are successful depends knowing the right questions to ask, when you should ask those questions, and who to ask the questions. In general, structured interviews seem to be rigorous and effective. A possible downside is that they could possibly limit the exploration of new concepts.

Unstructured interviews are more akin to a conversation, where the power to direct the interview is a lot lower than in a structured interview. While the flow of such an interview can feel a lot more natural, there are downsides to unstructured interviews. The emphasis put on certain topics can cause problems. Putting too much emphasis on a topic could make it so that there is too much focus being put on the topic. On the other hand this can other topics to be neglected, or even entirely ignored [31]. In general unstructured interviews are best used to explore a domain, or as a starting point before conducting more structured interviews.

Semi-structured interviews fit in between structured and unstructured interviews, borrowing elements from both other types of interviews.

Interview prompts play an important role in requirements elicitation interviews. The use of information-specific prompts is a recommended practice in requirement elicitation interviews [32]. Furthermore, the different types and categories of prompts have been captured in previous research [33], [34]. A technique for prompting that could be used, is questions using interrogatories such as: "Why", "How","What" ,"Who", "When", and "Where" [33]. Such interrogatories can be used to make questions such as "What should the user see when X", or "Who should be able to access the database". However, follow-up research on this, by Pitts and Browne [34], has proposed procedural prompts. These are prompts that should help overcome the cognitive limitations that analysts face. An overview of the types of procedural prompts, and examples of such prompts, can be found in Table 2.1.

Table 2.1: Procedural prompt for requirement elicitation interviews [34].

Strategy	**Sample prompts**
Summarization and Feedback	Can you summarize the functions and capabilities of the system?
	Can you summarize the features necessary for a successful system?
	When using the system, what type of feedback or information would you expect to get?
Repetition and rephrasing	Tell me again, what are the important features of the system?
	Can you restate, in detail, the steps for using the system?
	What decisions will you make while using the system?
Scenario building/Elaboration	What can you do now, but will not be able to do with the new system?
	Under what circumstances would (a specific function/capability) be necessary?
	Imagine that it is 6 months from now and you are evaluating the success of the system. What measures would you use to make that assessment?
Counterargument	Can you think of any reasons for not using the system?
	Why would you not want to use the system?
	Can you think of any reason that the system would fail or malfunction?

Furthermore, the procedural prompts as mentioned in Table 2.1, have been evaluated in the work of Pitts and Brown [34]. **Summarization and Feedback** helps elicit more complete information. It also lead to the discovery of unique alternatives. **Repetition and rephrasing** is helpful by providing evaluation of the reliability and validity of requirements. It could potentially help discover insufficiency's. **Scenario building/Elaboration** is helpful for elicitation of richer, and more detailed, requirements. It also stimulates consideration of circumstances and consequences. **Counterargument(s)** help validates requirements by generating disconfirming evidence. Lastly, it also forces the consideration of unique viewpoints. Furthermore, applying prompt in general is also found to be beneficial. It provides structure and focus to requirements elicitation, and it is applicable to a broad range of development environments.

2.3.1 Success criteria

Distanont et al. [35] provide a comprehensive overview of requirements engineering challenges. However, there are two challenges in particular that seem to be prevalent in requirement elicitation research: the influence of ambiguity [30], [36], [37] and domain knowledge [33], [38], [39]. The importance of both of those factors is highlighted in the work of Distanont et al. [35]. Understanding how to deal with such factors may be helpful, for example when trying to understand such complexities in LLMs. Therefore, both these challenges will be discussed in greater detail.

Ambiguity

When an analyst is conducting an interview, there is a knowledge transfer between the interviewer and the interviewee [35]. Ambiguity is one of the major obstacles that can arise in communication that can influence the transferring of knowledge [30], [36], [37] When something, expressed in natural language, can be interpreted in more ways than one, it is considered ambiguous. When something is expressed by the customer, and not understood properly by the analyst, there is subconscious disambiguation [40]. This phenomenon occurs, for example, when someone reads a document and thinks to have understood it because it did not appear to be ambiguous [41]. According to the work of Ferrari et al. [30], there are several types of ambiguity that can occur in requirement elicitation interviews. **Unclarity** refers to a situation where the interviewer is unable to give an interpretation to the information that the interviewee expresses. This can be because the information is expressed using domain jargon, or unfamiliar language. **Multiple understanding** is a type of ambiguity where the analyst is able to give more than one interpretation to the information that is expressed by the interviewee. An example of this could be the statement "I want to have a modern design". In the case of designing an app, this could mean that the app needs to look modern, or that the technologies in the back-end of the app need to be modern. Both are valid interpretation, but it does not become clear which one is meant. **Incorrect disambiguation** occurs when the analyst interprets a statement by the interviewee in the wrong way. In other words, the analyst assigns the wrong interpretation to what is expressed by the customer. Taking the previous example, it means that the analyst would assume one interpretation, while the other is the one meant by the interviewee. Such ambiguity can "occur" both detected and undetected. This means that it is either noticed and addressed during the interview, or that it becomes clear later on. **Correct disambiguation** occurs similarly to incorrect disambiguation in the sense that it is a statement made by the interviewee, that has more than one possible meaning. However, in the case of correct disambiguation, it is clear to the analyst what the interviewee meant with the statement, and the correct requirement can be elicited without further interruption.

Furthermore, the work of Berry et al [42], mentions two concepts that are related to ambiguity. **Generality**, occurs when a sentence needs extra explanation in order to make sense. For example, "The system should be fast.". Does this mean that the server the software should run on should be fast, or that the software itself needs to be optimized to be fast? **Vagueness:** An expression is considered vague if there are cases where it's challenging to determine the truth value. In other words, it occurs in cases where it is unclear whether the expression is true or false. This happens when the criteria for satisfying the expression are not precisely defined, or when the language used in the expression is ambiguous. For example, "The system should be fast.". It cannot be determined when such an expression holds true or not, since it is vague what is meant by fast.

Domain knowledge

It could be that business analysts have domain knowledge before starting the elicitation process. In general, it is assumed that this has a positive effect on the requirements engineering process [38]. Another example can be found in the work of Browne and Rogich [33], where different prompting schemes for requirement elicitation interviews are compared. In their work, they compare context-dependent and context-independent interviews. In this case, the context refers to the domain that is discussed in the interview. It was found that context-dependent prompting schemes were more powerful than the context-independent ones. However, in order to construct such a prompting scheme, the analyst needs to have significant domain knowledge to succeed. Furthermore, they have come up with a methodology to create prompts for interviews, independent of the skill of the analyst. Generally, it fosters communication and it creates shared understanding in terms of the needs of the customer. However, to some extent, prior domain knowledge can also have a negative impact. In the work of Wiley [39], domain knowledge was found to inhibit creative problem solving to a certain extent. Subjects were asked to perform a creative problem solving task, with the goal to produce something, with the task being easy enough for less skilled subjects to participate. Having domain knowledge turned out to not only bias the first attempt, it also causes to narrow down the search space when trying to come up with a solution. Furthermore, it was found that novices, with little domain knowledge, were more flexible in their way of thinking and came closer to finding the correct solution.

To summarize, interviews play a very important role in the elicitation of requirements. However, this does not mean they are always successful. This section has introduced the concept of requirements elicitation interviews, and some important shortcomings.

2.3.2 Conversation types

Next to interviews, there are other types of requirements elicitation conversations, from which requirements can be elicited. Zhang [43] defines the following conversational requirements elicitation techniques: "interviews", "workshop/focus group", and "brainstorming". Interviews have been discussed before, but what about the other two? When conducting a workshop/focus group, you bring stakeholders together to have a short, and focused, session to come up with, or review, high level aspects of the system to be [44]. In a brainstorming session, stakeholders are brought together to discuss idea more rapidly. Through quickly discussing ideas, and creating a broad list of ideas. it fosters a more creative way of thinking, that may lead to new ideas that may not have come up otherwise [45].

What these conversational elicitation techniques share is that they are commonly used, and that people usually are willing to talk about their work in a conversational setting [43]. This means that they are effective techniques to use to try and elicit non-tacit requirements.

However, such techniques are quite labor intensive [44]. Various challenges arise, such as:

scheduling meetings, producing notes/transcripts, analysing notes/transcripts. Furthermore, getting all the stakeholders in the same place at the same time, may prove to be challenging.

2.4 Large Language Models

2.4.1 Introduction

Language Models (LMs) learn to predict a word given a certain context, the previous sentence or in some cases the following sentence for example [46]. Furthermore, they are capable of understanding, and generating, natural language. LMs have the transformative ability to predict the next word or sequence of words, as well as the ability to generate new text, given a certain input [47]. **Large Language Models** (LLMs) are an advanced version of LMs. The word 'large' refers to the fact that they have many more parameters, or variables, to learn from the data that they are trained on. This makes them more advanced because by being able to learn a lot more from the training data, the performance across natural language tasks improves significantly [48]. In Table 2.2, some high level differences show the comparison between machine learning, deep learning, and LLMs.

Table 2.2: A comparison between Traditional ML, Deep learning, and LLMs [49].

Comparison	**Traditional ML**	**Deep learning**	**LLMs**
Training Data Size	Large	Large	Very large
Feature Engineering	Manual	Automatic	Automatic
Model Complexity	Limited	Complex	Very Complex
Interpretability	Good	Poor	Poorer
Performance	Moderate	High	Highest
Hardware Requirements	Low	High	Very High

As mentioned in Chapter 1, the use of such models is being researched in different sectors, as well as in RE. This section will go over the different types of LLMs, prompt engineering for LLMs, and protocols for leveraging LLMs, in order to get an understanding of the concepts that influence the output of an LLM.

2.4.2 Types of LLMs

When it comes to LLMs, there are two types that are most prevalent, the Generative Pre-trained Transformer (GPT) language representation model, and the Bidirectional Encoder Representations from Transformers (BERT) language representation model [50]. This section serves to provide a high level understanding of these concepts. Because the aim of this research is not to build a LLM, highly technical details will be simplified. The following section will go into further detail on how the BERT and GPT language representation model work. It is important to understand that both models use a transformer as the foundation. A visualization of the architecture of a transformer can be found in Figure 2.2.

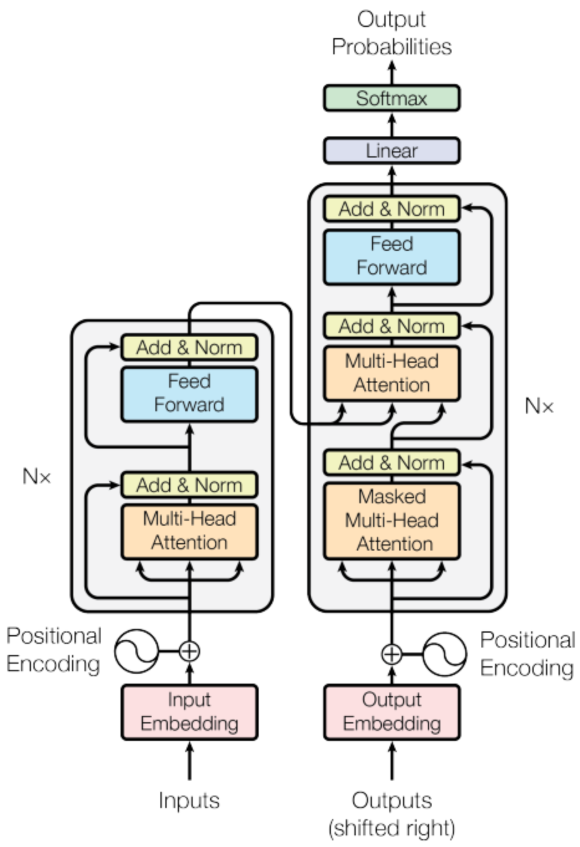


Figure 2.2: A visualization of the transformer architecture [51]

2.4.3 BERT

The BERT model first processes the input it is given. By breaking down textual input into tokens, which can be words, sub-words, or even characters, the input is represented in tokens. These tokens are then converted into a numerical vector, also called embeddings, which allow the model to understand the meaning, and relationship, between words [52]. There are three types of embeddings that a BERT model uses: positional, segment, and token embeddings. Positional embeddings are given to each token in the input, this makes it so that the model knows the position of a word in a sentence. Segment embeddings are embeddings of the first and second sentence of an input, which allow the BERT model to distinguish between. BERT makes this distinction to know to which segment of the input a token belongs. This can also be done with other sentence pairs. Lastly, token embeddings are the embeddings of the words, or sub-words, as mentioned before. Together, this forms an embedding scheme that offers a lot of knowledge about the input. The BERT language representation model is different from others because it is bi-directional. This means that it does not only take into account the left context, but also the right context [47]. This means that when evaluating a certain word, it takes into account the words before it **and** the words after it. Unlike a GPT model, that only takes the following words into account. This difference is visualized in Figure 2.3. In the visualization the following is represented: the yellow elements E are the input embeddings, the blue elements Trm are the transformer blocks, and the green elements T are the output tokens.

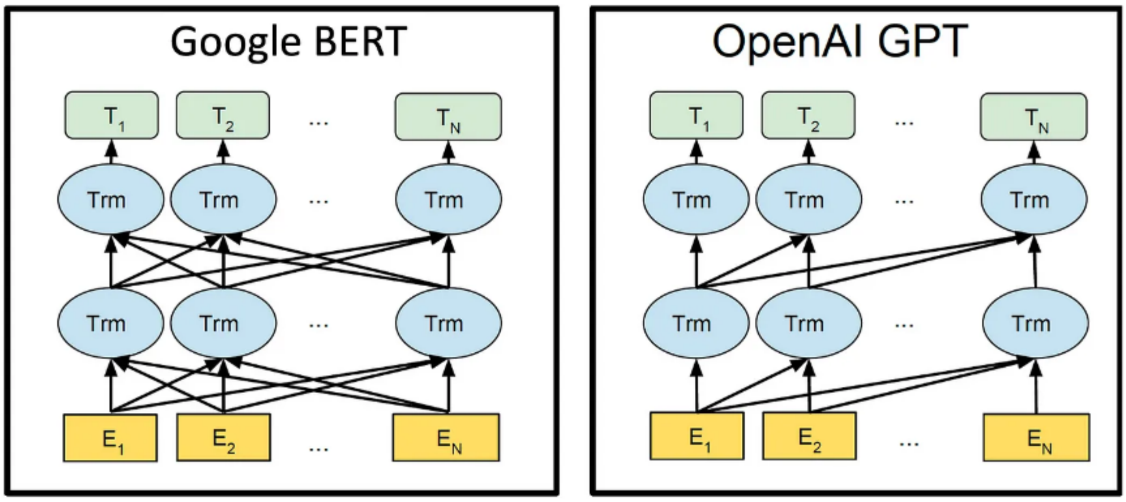


Figure 2.3: A visualization of BERT's bi-directional versus GPT's unidirectional nature[47].

After the creation of the embedding scheme, the model performs a Masked Language Modelling (MLM) task [26], [47], [52]. This means that certain tokens from the input are masked. It is up to the model to predict the masked tokens based on the context around the masked token. It masks 15% of the words of the input. That 15% is made up of 80% masked words, 10% randomized words, and 10% retained words [52]. This is done so that token representation for the "non-masked" words is not affected too much. The next step is Next Sentence Prediction (NSP). This is a similar task to the word prediction task, except that this task predicts the next sentence instead of a single word [52]. This step is conducted in order to give the model an understanding of the relation between the sentences that the input contains. In order to accomplish this, half of the inputs consist of pairs where the second sentence follows the first sentence in the original document. Meanwhile, the remaining 50% of inputs are formed by pairs where the second sentence is selected randomly from the document, with the assumption that the randomly chosen second sentence is unrelated to the first sentence.

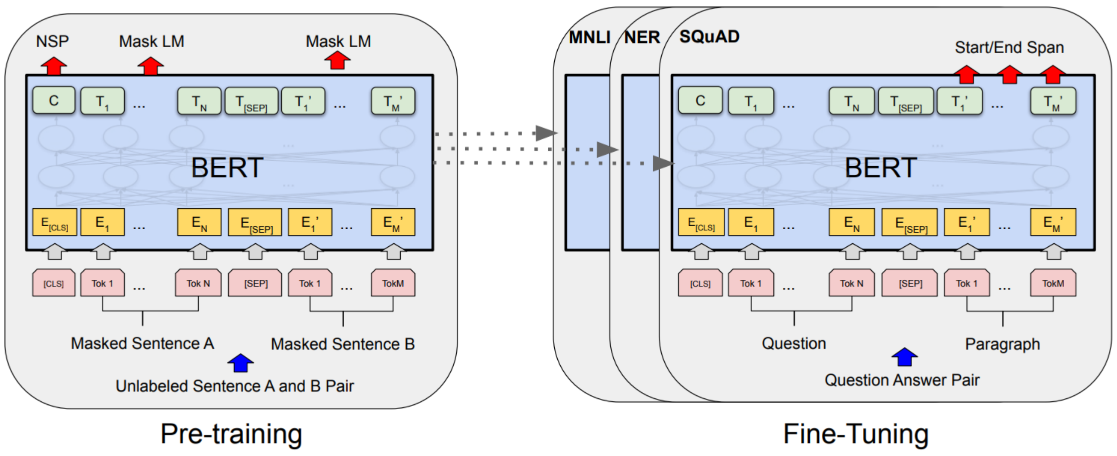


Figure 2.4: A visualization of the BERT architecture [47]

Together, these steps form the pre-training of the BERT model. Broadly speaking, pre-training means training the model to understand the way language is represented in the input it is given [53]. In this case, input means a large corpus of text where the text is not labeled. A visualization of the pre-training architecture can be found in Figure 2.4.

After the fine-tuning phase, the adapted BERT model is ready to be deployed for inference on new data. During inference, the model takes in raw text input and processes it, utilizing the learned representations and task-specific parameters acquired during fine-tuning.

For text classification tasks, the fine-tuned BERT model applies softmax activation to the output layer to produce probabilities for each class, indicating the likelihood of the input belonging to each category. In question answering tasks, the model utilizes its understanding of contextual information to identify the most relevant segments of text containing the answer.

Under the hood, the fine-tuned BERT model computes these predictions through a process called forward propagation. This involves passing the input text through the neural network layers, where each layer applies a series of mathematical transformations to the input data, gradually transforming it into representations that are increasingly tailored to the specific task at hand. Finally, the output layer produces the desired output.

Once the predictions are generated, they are typically post-processed to improve their readability or usability. For instance, in question answering tasks, the model may output the start and end positions of the answer span within the input text, which can then be extracted and presented to the user in a more intuitive format.

2.4.4 GPT

A GPT model is an autoregressive language, which uses deep learning to create natural language [54]. What makes a GPT model different from the aforementioned BERT model, is that a transformer is used in the architecture of the model [51], which the explanation below will be based on. The visualization of the transformer architecture, as explained below, can be seen in Figure 2.2.

Similarly to the BERT model, the first steps for processing an input, is embedding the input and encoding the position of the tokens. The transformer part consists of two stacks, the encoder stack which process the input, and the decoder stack which generates the output. The encoder stack consists of two layers, the multi-head self-attention mechanism, and the position-wise fully connected feed-forward network. The decoder layer is similar, but with an additional layer where it performs multi-head attention. In simple terms, the multi-head self attention part allows the model to determine the importance of the different parts of the input sequence when it processes all the tokens within the input. The mechanism runs multiple linear projections of the input, in parallel. Doing it, ensures that the model can capture the different relationships between the tokens. After the attention mechanism has been leveraged, the input goes through a position-wise feed-forward network. This means that the input tokens are all processed independently from one another. After the input has gone through both the attention mechanism and the position-wise feed-forward network, for both the encoder and the decoder layer, the outputs are normalized. The second to last step is the extra attention mechanism that the output of the encoder layer goes through. Lastly, the output is generated. This is done by taking the output of the decoder and linearly transforming them into probabilities for the next token. Then based on those probabilities the output is produced.

What makes a GPT model stand apart from other models, is the attention mechanism [55]. This helps the model understand the relationships between words that are far apart. Furthermore, the transformer is built in a modular fashion. For example, the attention mechanism and the position-wise feed-forward network, are both sub-layers to the encoder and decoder stacks. This means that it is easier to modify these modules of the GPT model. Furthermore, the attention mechanism allows for differentiating between the important and lesser important words in an input. Lastly, the parallel nature of the transformer makes a GPT model a lot faster in terms of time it takes to compute the inputs it is given.

2.4.5 Prompt engineering

Prompt engineering is a flexible, and intuitive way for users to interact with a large language model [56]. In other words, a prompt refers to a specific instruction or question that you give to a language model, directing its actions to produce the desired results. On the very surface, a prompt consists of a few elements [56], the instruction, the context, the input data, and the output indicator. The **instruction** is for example the task you want the LLM to perform. This guides the behavior of the model and it directs the model to towards the output the user wants. Second, the **context** refers to extra information that the user gives the model to provide background knowledge. This helps the response be more accurate. The **input data** refers to the question or input the model needs to process. This is a core element to the prompt. Lastly, the **output indicator** refers to the format the user wants the response to be in.

There are more sophisticated prompt types, with each their own characteristics. A distinction can be made here between hard prompts and soft prompts. Hard prompt are created either manually or automatically, in natural language, and soft prompt are vector representations [57]

Soft prompts

Adjusting soft prompts is also referred to as prompt tuning [57]. Prompt tuning is a technique proposed as a simplification for adapting language models to downstream tasks [58]. In prompt tuning, the entire pre-trained model is frozen, meaning its parameters remain fixed. Instead of fine-tuning all model parameters as in traditional model tuning, prompt tuning only allows for a small number of additional tunable tokens, denoted as "k," to be added to the input text for each downstream task.

These additional tokens, referred to as a "soft prompt," are appended to the input text and are trainable during the adaptation process. By doing so, the model can capture task-specific information while still benefiting from the pre-trained knowledge encoded in the frozen model. This approach enables prompt tuning to leverage the signal from a full labeled dataset, thus outperforming few-shot prompts and narrowing the performance gap with traditional model tuning.

One of the key advantages of prompt tuning is that it allows for the reuse of a single pre-trained model for all downstream tasks. This contrasts with traditional model tuning, which requires maintaining separate copies of the model for each task. Therefore, prompt tuning retains the efficiency benefits of frozen models while achieving competitive performance with model tuning [58].

Hard prompts

In the scope of this research, the focus will be on natural language (hard) prompts. Hard prompts are easily understandable pieces of text that can be a added before an input. An example of this is: "Rewrite the following paragraph to be shorter: [input]" Furthermore, the work of Gu et al. [57], defines four sub-types of hard prompts, task instruction prompting, in-context learning, retrieval-based prompting, and chain-of-thought computing.

Task instruction prompting When using such a prompt, the LLM is given an natural language input that describes a desired output [59]. Furthermore, the input consists of instructions and resources which can be used to give the model a better context, and guide the model towards the desired output. An example of this is "Create a shopping list for dinner. Include ingredients for spaghetti.". In this case the instruction is to create a shopping list, and the addition of including ingredients for spaghetti is a resource to guide towards the desired output.

In-context learning This type of prompting relies on the LLM to learn from examples [60]. The LLM is given a few examples that together form a context. Then, a piece of one of the given example is taken together with a question to form a prompt. The LLM is expected to take the examples, or rather the context, into account and decipher a pattern. That pattern is then used in order to make the correct prediction when producing the output. An example of this could be that a LLM is fed five haiku's, and comes up with a new haiku. The context helps define the grammatical structure of a haiku, and the prompt gives the LLM the topic of the haiku. Through subjecting the model to a series of interconnected examples and/or prompts, the in-context learning technique enhances its performance in comprehending and generating responses [61].

Retrieval-based prompting means that the prompts that are being used for a certain task that a user might want a LLM to perform, is retrieved from a retrieval base [57], [62]. The retrieval base consists of pairs of inputs and responses that have been established before. When working with retrieval-based prompts, the LLM is given context by making it aware of the contents of the retrieval base that match with a certain similarity to the task the user wants to perform. Whatever examples are chosen by the prompt retriever, are used in order to give the LLM context. This is similar to few-shot prompting in the sense that the LLM is made aware of a certain context through examples. However, it is something different than few-shot learning. In the case of retrieval-based prompting, the output is generated with previous answers and responses. In the case of few-shot learning, the LLM is given examples of something, and then asked to do a similar task. In this case only a similar output is given to the LLM, and not the input, as is the case with retrieval-based prompting.

Chain-of-thought computing, refers to a method in which the LLM is given instructions in a iterative manner [61]. Throughout the process of using this method, the focus narrows, and the context is being enriched constantly. By "building" a conversation this way, the LLM is able to build a more coherent understanding of the context, and is able to keep up with an evolving context.

2.4.6 Patterns

White et al. [63], state that prompt patterns, to be used with LLMs, are similar to software patters used in software engineering. In both cases, patterns offer a reusable solution to a specific problem. While prompt patterns are used for generating output from LLMs, they both offer a systematic approach to solving challenges. Prompt patterns offers a systematic approach to customize the output of LLMs, and interact with LLMs.

White et al [63] introduce an approach to represent prompt patterns. An overview of this can be found in Table 2.3.

Table 2.3: An approach to capture prompt patterns systematically [63].

Category	Description
Name and classification	The prompt pattern name uniquely identifies the pattern and ideally indicates the problem that is being addressed.
Intent and context	describes the problem the prompt pattern solves and the goals it achieves. The most direct translation of software pattern structure to prompt patterns is the naming, intent, motivation, and sample code. The structure and classification, however, although named similarly, require more adaptation. should ideally be independent of any domain, though domain-specific patterns may also be documented with an appropriate discussion of the context where the pattern applies.
Motivation	The motivation provides the rationale for the problem and explains why solving it is important. The motivation is explained in the context of users interacting with a conversational LLM and how it can improve upon users informally prompting the LLM in one or more circumstances. Specific circumstances where the improvements are expected are documented.
Structure and key ideas	The structure describes the fundamental contextual information, as a series of key ideas, that the prompt pattern provides to the LLM. These ideas are similar to “participants” in a software pattern. The contextual information may be communicated through varying wording (just as a software pattern can have variations in how it is realized in code), but should have fundamental pieces of information that form a core element of the pattern.
Example implementation	This example demonstrates how the prompt pattern is worded in practice.
Consequences	This summarizes the pros and cons of applying the pattern and may provide guidance on how to adapt the prompt to different contexts.

In addition to the overview of how to structure prompt patterns, White et al. provide an initial overview of prompt classifications and corresponding prompt patterns [63]. Since this is a rather extensive list, a few interesting ones will be highlighted rather than listing them all.

- The **meta language creation pattern** is intended to be used to learn the LLM to understand another "language", for example shorthand notation used in graphs. The LLM understanding such a language can be useful if a user wants it to perform a domain specific task, or if it allows a user to express things more unambiguously. Such a pattern could be useful for the domain model derivation task. By making the LLM learn a shorthand notation for the relationships between a model, it would become easier to leverage the LLM to answer questions about a domain model.

- The **question refinement pattern** aims to leverage the LLM in order to refine the questions that the user asks the LLM, in order to generate a potentially better output. If the user lacks domain knowledge, or is not able to come up with the right words for a question, the LLM might be able to do suggestions of what could be meant by the user. By implementing such suggestions in the prompt, the output the LLM provides could be of better quality/more fitting, than the output based on the original question. This could be beneficial to the domain model derivation task when the modeler runs into less explicit modelling decisions, the LLM could possibly provide a solution based on output of a refined question.

- The **alternative approaches pattern** is intended to help users take another perspective away from the solution that they would normally choose. More often than not humans suffer from cognitive bias, which may lead them to opting for a solution that may not be optimal given the situation. This pattern aims at dissolving such cognitive biases by providing the user alternative solutions for solving a problem. It may be a handy addition to have more than one possible solution for a certain problem when making domain models. Having multiple perspectives on how to handle a certain modelling decision could lead to better domain models.

Other work, more specifically on requirements engineering, allows the user to leverage an LLM to identify ambiguity in a requirements specification [27]. It can be used to review a requirements specification document, or parts of it, to identify any potential vagueness or ambiguity. Such vagueness/ambiguity may also appear in requirements elicitation conversation transcripts, therefore this pattern could also be useful for this research.

2.4.7 Zero-shot, one-shot, few-shot

The concept of zero-shot learning refers to classifying instances on which the LLM has not been trained [64]. As mentioned before, a LLM is trained on a large set of data, in order to identify relationships between the words and sentences in the input [51]–[53]. However, in practice it occurs rather often that a LLM is asked about something that it has not been trained on [64]. While this is not really an issue for certain tasks, it can be an issue for others. In the case of visual recognition, it has been found that the fewer examples a LLM has, the more difficult it becomes to recognize something [65].

With one-shot learning, the assumption is made that there is only one example of each category available [66]. In other words, it means that of each category that an LLM is trained on, there is only one example of that category that the LLM knows. For example, if an LLM is trained on animals, and the animal in the task is a panda, the LLM is only trained on one example of a panda.

In few-shot learning, a LLM is trained on a few examples [67]. It can learn to perform new tasks based on the prior training of the model, together with the few examples it is provided. This allows the LLM to learn the more rare cases, and decreases the need for large supervised training data sets.

2.5 Domain modelling

2.5.1 Introduction

Domain modelling for a system or a software development task is about creating domain descriptions [68]. The creation of domain models serves as a method to generate an abstract understanding of the "world" in which the system to be will operate. It allows to reason, and validate, assumptions about the domain. Furthermore, it could allow analysts to reuse requirements within a domain.

A domain model generally consist of two parts [69]. First, the domain specific rules, terminology, and notions, that describe the taxonomy of the domain, and the specific rules and principles that apply to it. Things such as functions, data types, and concepts are included in this. Second, the domain model consists of the context model. This part of the model describes the general properties of the systems environment. Elements such as actors, users, software systems, and physical systems are included in this part of the model. Altogether, this captures all the required knowledge about the problem domain in which the system to-be is going to operate, which allows for capturing the requirements for the system to-be. Domain-specific models have proven to be an essential ingredient in the creation of automated tools, because of they allow tractable reasoning [70].

"Ultimately, the domain model is a collection of knowledge about the application domain at an adequate level of abstraction—including the use of modeling techniques where useful."

Manfred Broy

2.5.2 Modelling languages

According to the International Requirements Engineering Board (IREB), there are various different modelling languages/notations, that can be used to model the domain [71]. Important things to consider when deciding on which model type to use are, "What is the purpose of the model?", "Which requirements need to be modelled?", and "Who is the audience to which to model should cater?". Based on these, and other, considerations, the IREB mentions modelling languages that could be used to model requirements. To model systems of a highly reactive nature, UML state machine diagrams can be used. Such models an be supplemented by including use case diagrams, or activity diagrams. For systems that are not reactive in nature, for example loan application software, diagrams that allow modelling complex information structures should be used. IREB suggests using UML class diagrams. In order to model more process-oriented requirements, Business Process Management Notation (BPMN) is suggested. For both these types of models, state machine diagrams can be included in order to model the reactive parts of the domain.

Overall, universal modelling approaches, such as UML, are used very often [69].

2.5.3 Model quality

In similar work [72], the quality of model is assessed using the work of Lindland, Sindre, and Sølvberg [73]. There are three types of quality in the framework that they propose.

- Syntactic quality

- Semantic quality
- Pragmatic quality

Syntactic quality refers to the correct structure, grammar, and syntax of a conceptual model. A syntactically sound model ensures that the elements and relationships are represented in a manner that adheres to the modeling language's rules and guidelines. **Semantic quality** deals with the meaning and interpretation of the elements and relationships within a conceptual model. A semantically rich model accurately represents the real-world entities and their relationships, ensuring that the model conveys a meaningful and correct understanding. **Pragmatic quality** focuses on the alignment of the conceptual model with the intended purpose and goals for which the model is created. A pragmatically effective model serves its purpose, providing relevant and valuable insights for the targeted stakeholders and/or developers. In summary, assessing the quality of conceptual models involves evaluating syntactic correctness, semantic accuracy, and pragmatic effectiveness. A well-crafted conceptual model not only adheres to the formal rules of the modeling language (syntactic quality), but also accurately represents the real-world entities and their relationships (semantic quality), and serves the intended purpose effectively (pragmatic quality).

In the work of Rumbaugh and Blaha [74], criteria are given which a modelling element should, or should not meet. There are criteria for elements such as, classes, associations, and attributes. Whilst all of them are relevant, a short example will be provided in Table 2.4 that shows exclusion criteria for candidate classes.

Table 2.4: Exclusion criteria for classes in UML diagrams [74].

Criteria	**Explanation**
Redundant classes	Eliminate redundant classes by favoring the most descriptive name when two classes express the same concept.
Irrelevant classes	Remove irrelevant classes that have little relevance to the problem at hand. Consider the context, as a class may be important in another scenario.
Vague classes	Refine vague classes by ensuring specificity. Classes with ill-defined boundaries or broad scopes should be avoided.
Attributes	Transform names primarily describing individual objects into attributes.
Operations	Differentiate between classes and operations. Classes should represent intrinsic natures, while operations should be modeled separately if they have distinct features.
Roles	Ensure class names reflect intrinsic nature rather than roles played in associations.

Implementation constructs	Be cautious with implementation constructs, removing implementation elements from the analysis model.
Derived classes	Minimize derived classes unless essential, marking them with a preceding slash ('/') in the class name.

UML State Machine Diagrams

State Machine Diagrams are one of the most useful, most widely used, UML diagrams within the field of RE [75]. Such a diagram can be created to show the behavior of a single class. It can also show the behavior of an object between different use cases. In terms of software modelling, UML State Machine Diagrams have been found to be used a lot for code generation [76]. A drawback of State Machine Diagrams is, that they are not suited to describe behavior related to collaboration between objects [77].

In a more detailed manner, State Machine Diagrams, show the different states that a system, or a component of a system can be in [78]. Furthermore, it also captures the transition between states, where each state represents a condition or mode of operation. There are events that might occur, that determine the state of the part that is modelled in the diagram. There are conditions (guards), that are related to these events. These guards act as decision points on whether a certain transition will take place or not. Lastly, actions represent activities that are related with transitions. Actions are performed when an event triggers a transitions. They can for example represent changes in the internal state of the system, or other behavior that may be relevant.

Similar to the ontological mapping of object oriented concepts to UML class diagram elements, a framework has been made that maps UML state diagram concepts to requirements artifacts [79]. Requirements artifacts, such as a business object lifecycle or a business rule, are mapped to a state machine and a constraint. An overview of how the requirements artifacts are mapped to state machine diagram concepts, can be found in Table 2.5.

Table 2.5: The mapping of requirements artifacts to state machine diagram concepts [79].

Requirements artifact	**UML concept**
Business object lifecycle	State Machine
Business process	Activity
Business task	Action
Business rule	Constraint, Guard
Usage scenario	Activity, Interaction
GUI navigation schema	State Machine

UML Class Diagrams

When it comes to the development, analysis, and design of software systems, UML is is the most popular modelling language to use [80]. Research often focused on the translation from natural language requirements into UML class diagrams [81], [82]. Furthermore, automating the generation of UML class diagram from natural language requirements has been research often [80], [83], [84]. It is important to understand that the UML modelling language family consists of 14 types of diagrams, as of version 2.4, and that they can both cover structure and behavior [69].

Some automated techniques are able to generate UML elements and models [84]. However, others require consistent human intervention and interaction, to be able to come up with UML models. Overall, it became clear that there is currently not a solution that is able to generate all the UML model elements, e.g. class names, operations, relationships etc.

UML modelling guidelines have been established, in order to improve the quality of created models. It seems relevant to include guidelines on UML modelling, in order to be somewhat able to asses the quality of a UML model.

Ontological guidelines have been established for the use of association classes [85]. An overview of the established guidelines can be found in Table 2.6.

Table 2.6: Ontological guidelines for the use of association classes in UML diagrams [85].

Category	Description
Mutual properties must be represented as attributes of association classes	In UML, association class attributes correspond to slots in links connecting objects. The concept of mutual property aligns with these link slots, as their values describe connections between ontological entities. The association class attributes collectively embody mutual properties. It's important to note that the association class is a structural container without inherent ontological significance.
An association class must not possess methods or operations	Association classes, representing sets of mutual properties rather than substantial things, lack the capacity for methods or operations. Only actual things, not their properties, can engage in action and interaction ontologically.
An association class must possess at least one attribute	An association class without attributes would represent an empty set of mutual properties. Such a class is ontologically meaningless
An association class must not be associated with another class	In ontology, association classes with attributes representing mutual properties should not be linked to other association classes. In UML terms, this means an association class must not own properties that serve as member ends in other associations.

An association class must not participate in generalization relationships	In UML, properties and associations can be generalized and specialized, but in ontology, mutual properties cannot be generalized. Therefore, association classes with attributes representing sets of mutual properties should not be generalized in ontology.
Association classes model mutual properties from the same interaction among class instances. Different interactions with distinct sets of mutual properties should be represented using separate association classes.	Previous research, proposes that mutual properties arise from interactions between entities. We refer to sets of mutual properties resulting from the same interaction event as concurrent mutual properties. Each association class and its attributes represent a set of concurrent properties. Therefore, distinct association classes and their attributes should be employed when mutual properties are not concurrent.

Furthermore, ontological concepts have been mapped to domain modelling concepts. In previous work, researchers have tried to come up with a foundation for ontology based object-oriented domain modelling [86]. This is interesting because it matches ontological concepts to objects used in the UML class diagram. This is relevant for the second phase of the research. An overview of the ontological mappings can be found in Table 2.7

Table 2.7: The mapping of ontological concepts to the modelling objects [85].

Category	**Description**
Thing	Object
Property	Attribute
Intrinsic property	Attribute of ‘ordinary’ class
Mutual property	Attribute of association class
Emergent property	Class attribute
Functional schema	Class
Natural kind	Set of objects (extension of object-class)
Composition	Aggregation/Composition
Re-classification	Object creation
	Object destruction
	Object identifier
	Association

BPMN

Business Process Model and Notation, or BPMN, is a modelling language that is based on flow-charts [87]. It can be used to describe different scenarios that define business processes. Furthermore, it can be used to identify user tasks, that may or may not be computer supported, which can be linked to use cases. In recent years, BPMN has become one of the most applied general purpose modelling languages, in the business process management discipline.

"Such business process models enable the common understanding of the kind of business activities underlying a business process, including their mutual relationships."

Koschmider and Reijers

Since business process modelling can be done for all sorts of domains, e.g. healthcare, software development, retail, considerable effort has been put into researching extensions of the BPMN language [88]. The BPMN language allows for the extension of the language with domain specific elements [89]. Through the "extension by addition" mechanism, the definition and integration of domain specific elements is made possible, without harming the validity of the core BPMN concepts. For domain modelling, the BPMN language is mostly used for capturing the process related parts of the domain [71], and the structure and coordination of the domain [90].

When taking a process oriented perspective on requirements engineering, BPMN is often the go-to modelling language [71], [91]. Intrigila et al., provide an extension to the BPMN language that mitigates the risk of ambiguous- and incomplete requirements [91]. They provide a solution where data properties are annotated within the model, by means of constraints. These constraints are the pre-and post-conditions that the activities within the business processes must satisfy. This easy way of adding extra behavioral, and data, properties, to a BPMN model, foster communication between stakeholders.

Chapter 3

Research method

3.1 Introduction

This chapter will serve as an explanation of the systematic framework that is chosen, to investigate and address the research questions and objectives outlined in Chapter 1. It serves as the blueprint for the research design, data collection, and analysis processes, by offering a comprehensive overview to understanding the rigor and validity of the study. By elaborating on the intricacies of the chosen research approach, this chapter aims to provide a transparent overview of how the practical part of this research will be conducted.

3.2 Research philosophy and type

This research will take a pragmatic approach. This means that it will be more flexible, to focus more on the usefulness and applicability of the findings. Furthermore, there is still a big need for research into the use of LLMs, and how to use them effectively, as has become clear in Chapter 1. Continuing, this research will be inductive, taking a bottom-up approach. Since there are no real proven theories when it comes to domain model derivation by using LLMs, it makes sense to adopt a bottom-up approach. This means an experiment will be conducted to gather data and make observations, which then will be analyzed to see what can be learned from it.

3.3 Research design

The research design is partly based on the case study protocol, as presented by Maimbo and Pervan [92]. It offers a Case Study Protocol (CSP), that helps build the design of the research. Furthermore, it elaborates on the procedures, the research instrument(s), and the data analysis. While this research may not be a true case study, building the research design based on such a framework, allows for structuring the practical part of this research in a rigorous manner. Each of the parts defined in the CSP, will be elaborated upon, creating a solid overview of the protocol for this research.

3.3.1 Preamble

Confidentiality and data storage are very important. Personal information and all other kinds of personal information that might occur, will be treated with confidentiality. This means that they will only be shared in an anonymized manner, and that the principal investigator is the only one with direct access to all that information. Data will always be stored in a closed off environment that is not accessible by anyone from the outside. If, and when, this research will be published, all data related to participants will all be presented in an anonymized manner. Furthermore, the

Ethics and Privacy Quick Scan of the Utrecht University Research Institute of Information and Computing Sciences was conducted, and can be found in Appendix A. It classified this research as low-risk with no fuller ethics review or privacy assessment required.

3.3.2 General

The research aims, and the relevance of the research, are elaborated upon in Chapter 1. The goal of the primary experiment, is to gather quantitative data to analyze, in order to answer the research questions. The data that is used for the experiment comes from a dataset in which each case is about a different software system to be built. Each case contains interview transcripts and documentation, that is used to create a domain with. For each case, a manually made domain model is created, without any pre-existing knowledge of the case, but only based on the two transcripts (one per interview) and a vision document that the stakeholders prepared for the interviewees prior to the interviews. This domain model should try to capture the data structure of the system to be built, as much as possible. After the manually made model has been established, ChatGPT will be asked, through several fine-tuned prompts, to also create a domain model for the same case. After a model is generated for the case, both models are compared on their modelling components. This comparison results in a set of quantitative results. These results are used to compare the human made models, with the models generated by ChatGPT.

3.3.3 Procedure

Start-Up Phase The procedure is the same for each case. After creating and improving the prompts, and making sure the results are somewhat similar to what is expected, the experiment can commence. This step is unstructured and can depend on what the goal is of what one might want to accomplish. In the case of this research, it is meant to refine the prompts, and to get a better understanding of ChatGPT's capability to model a domain.

Modelling After this initial start-up phase, the procedure for the primary experiment starts. For each of the cases, first the vision document is studied, followed by transcripts one and two. For each of these parts notes will be gathered, that can be used throughout the manual modelling process. Furthermore, it might be used in order to refresh the memory of the researcher during the quantification of the results. After going through the transcripts and the vision document, a domain model is created in Draw.io. After this manually created model is established, ChatGPT is used to generate a domain model. By applying the prompts, as can be found in Appendix B in Section B.6, ChatGPT gives an output in PlantUML code. This code is then entered into the online PlantUML web server, which is able to visualize the model specified in the code. All intermediate steps, answers, and models, are saved in the same document as the manually created model. This ensures that all information belonging to each case is stored together.

Quantification After both models have been created the models can be compared, and the results of the modelling parts will be quantified. To do so, a quantification method has been established. All cases are put side by side, in order for easy comparison. Then all components are marked step by step. Each entity, attribute, relationship, and cardinality, is analyzed and marked as matching with a certain scenario. The various scenarios that have been established are based on the types of knowledge, and the modeler. In Figure 3.1, the foundation of the scenarios can be found. Something can (not) be modelled by the human only, by ChatGPT only, or by both.

Furthermore, the distinction can be made between the types of knowledge that an element can be modelled upon. There is direct knowledge, meaning that an element can be traced back to the transcripts or the vision document, domain knowledge, meaning something cannot be traced back to the domain model but can reasonably be assumed to be a fitting choice, or missing/wrong knowledge, meaning that something is clearly missing from the model or something that cannot reasonably be assumed to be a fitting choice for the model. On these foundations, scenarios have been established, an overview of them can be found in Appendix C.

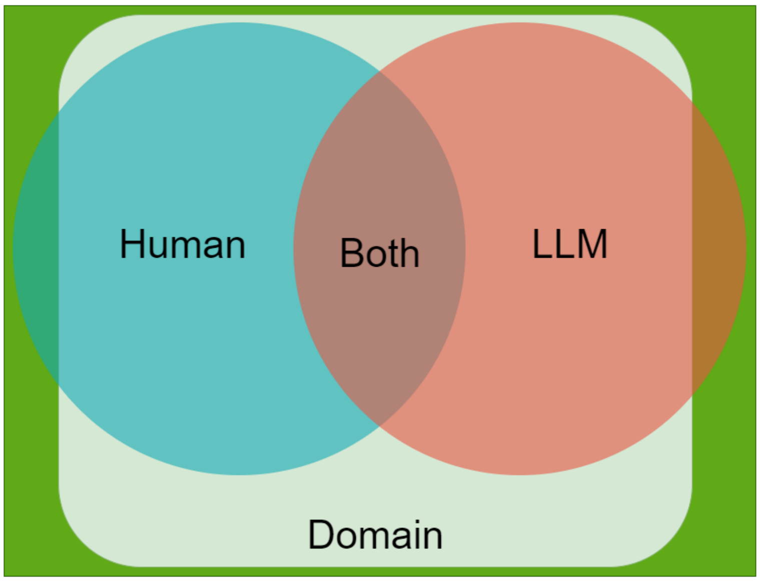


Figure 3.1: A Venn diagram showing the foundation on which the knowledge types and scenarios are based.

Scoring For each case, the elements are scored individually. Meaning that each case has four tables, one for each modelling element, where each individual entity, for example, is matched with a scenario. Going through all the cases, this will result in a sheet for each case with four tables. In essence, each table counts the number of occurrences for each scenario. These numbers are then combined into various tables, in order to highlight for example the performance of the human, or the types of knowledge that the LLM has relied on in order to model a certain case.

Analysis After the aforementioned steps have been taken, the results will be analyzed. This is done by performing statistical tests on the data, in order to test the hypotheses that have been established in.

The entire process has been visualized in Figure 3.2.

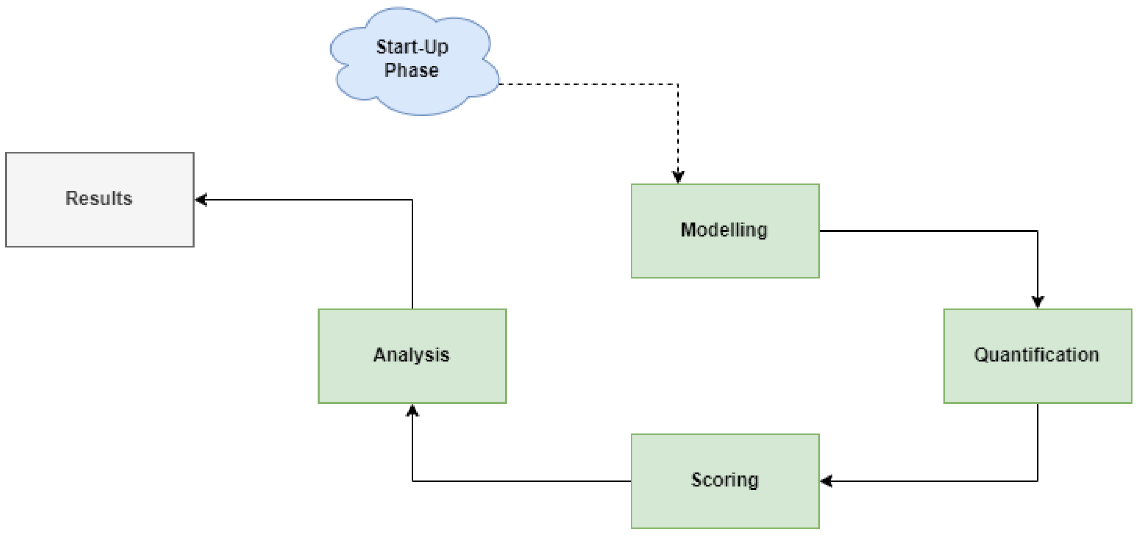


Figure 3.2: An overview of experiment procedure.

3.3.4 Research Instrument

The research instrument for this project is the quantification method. There were ready-made solutions that can be used in order to do this experiment. The method relies on the interview transcripts and the vision document, in order for the domain models to be created. These are all the same for each case. Each case consists of two interview transcripts in the range of 30 minutes to 1 hour, and 1 vision document. This provides an equal foundation for both the human modeler, as well as the LLM. The scenarios are specular, meaning that they are the same for both the human and the LLM. This part is also easily adjustable, in case any unforeseen issues arise. The Excel sheet allows for the easy creation of tables, and keeping track of all the statistics for each individual case. An overview of the scenarios can be found in Appendix C, and the Excel sheet can be found in Appendix B, section Section B.7.

3.3.5 Considerations

One of the choices that needed to be considered was the use of ChatGPT. There are many different LLMs out there, as has become clear in Chapter 2. However ChatGPT was deemed an appropriate choice. Prior use of ChatGPT, as well as the familiarity of the researcher with ChatGPT, have led to the choice of using it. Furthermore, ChatGPT is able to handle documents, which is required to perform the experiment. Lastly, it also allows to upload images for analysis, which is deemed a useful feature.

3.3.6 Data analysis guidelines

There are three different aspects that are going to be analyzed: the agreement between the human and the LLM, the usable element creation of both the human and the LLM, and the unusable element creation of both the human and the LLM. The usable and unusable element creation is tested by performing the Wilcoxon signed-rank test. The Wilcoxon signed-rank test is a non-parametric statistical test, used to compare two related samples or repeated measurements on a single sample to assess whether their population mean ranks differ. This is useful because it cannot be assumed that the data is distributed normally. The test works by calculating the differences between paired observations, ranking the differences, and then analyzing the ranks of the positive and negative differences. By focusing on the ranks rather than the raw data, the Wilcoxon signed-rank test is able to handle skewed distributions and outliers. This makes the test useful for comparing means, and handling changes in the central tendency of the data.

Agreement The agreement between the human and the LLM can only be measured using the modelling elements that are modelled based on direct knowledge, because both parties can only agree on elements that exist in either the transcripts/documentation. Therefore only the occurrences of the scenarios relation to direct knowledge, are taken into account. To measure the agreement, the percentage of the amount of elements that both the human and the LLM have modelled based on direct knowledge, is used. This is done for each of the modelling elements separately, as well as for all the elements combined.

Usable element creation Usable element creation, means the amount of elements based on both direct knowledge and domain knowledge, are compared between the human and the LLM. This is done to highlight the differences in usable/correct element creation when modelling. To measure this, a Wilcoxon Signed-Rank Test is performed for each of the modelling elements, as well as all elements combined.

Unusable element creation Unusable element creation, means the amount of elements based missing/wrong knowledge, are compared between the human and the LLM. This is done to highlight the differences in unusable/wrong/missing element creation when modelling. To measure this, a Wilcoxon Signed-Rank Test is performed for each of the modelling elements, as well as all elements combined.

Chapter 4

Results

4.1 Introduction

To get a better understanding of the capabilities of ChatGPT to generate domain models, and to see how that compares to manually created domain models, initial experimentation has been done. The term initial experimentation refers to a process of free-form experimentation in order to narrow the scope, and define boundaries. This meant narrowing what would be possible to research, for example is it possible to compare more than one LLM or is it possible to compare multiple LLMs. Defining the boundaries was about setting a limit for what would be considered acceptable; does a model need to be near perfect, or should it simply be a decent model. This section will elaborate more on this process, and the insights that have been gained from going through this process. By showing the progression, and elaborating on the improvements that were made with each iteration, the foundation of further research in this thesis becomes clear. The initial experimentation can be divided into three parts, start, traceability, and normalization.

Each part will be elaborated upon in the following section.

4.2 Initial Experimentation

4.2.1 Start

The first part of the initial experimentation process served to try and get an understanding of what ChatGPT could do when asked to create a model based on provided transcripts. By using a minimal prompt, and attaching the first one of two transcripts of a case, ChatGPT generated PlantUML code that could be inserted into a modelling tool. By adjusting the first prompt, also the information of the second transcript was given to ChatGPT. The first prompt was written as follows:

```
    "The following txt document contains an interview between
business analysts and client.  They are discussing a software sys-
tem to be built.  I want you to create a UML class diagram,
in PlantUML code, that describes the data structure (database) of
this system to be built.  Based on the conversation provided
in the txt file.".  (file included)
```

When extending the model with information from the second transcript, it was stated explicitly that the model needed to be extended based on that information. With knowledge of both the transcripts, ChatGPT manages to give a PlantUML code as output, which was then put into the

PlantUML web server demo. This visualized the generated code as a UML class diagram model, as can be seen in Figure 4.1.

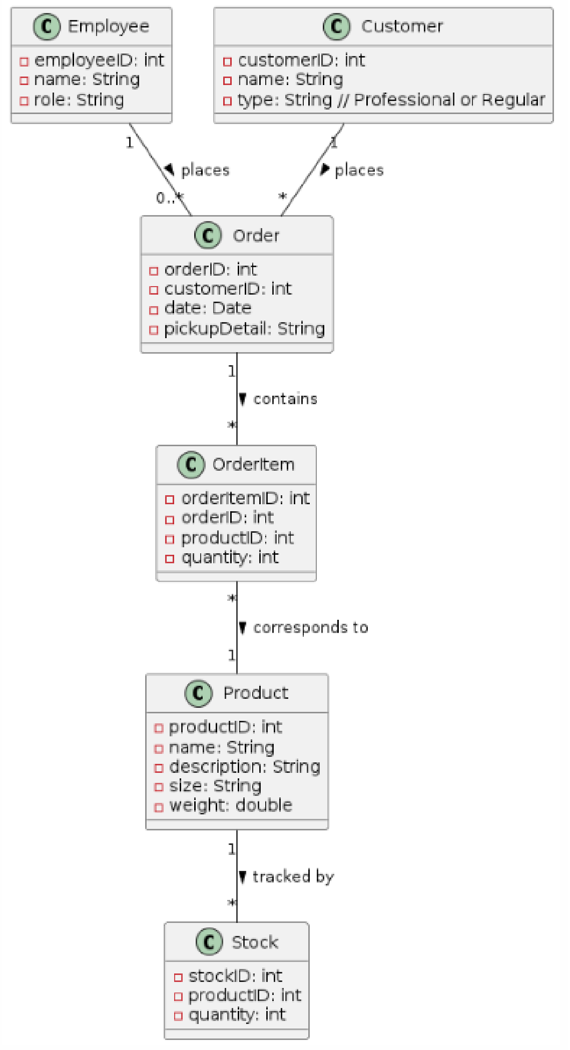


Figure 4.1: A generated model made with the initial prompt.

At this stage the model is not yet specific enough, and there are still mistakes in the model. For example, the relationship between the employee and order is wrong. The way that it is modelled now means that the employee orders products. It should be an employee handles an order. Furthermore, the cardinalities suggest that an order should always consist of multiple orderItems. This is not always the case, for example when a customer order just one item. The goal is to be able to generate a domain model that captures the data structure for a system to-be. One of the things that can still be changed to do so, is to for example model employee and customer as one class, user. At this stage, such improvements were made by adding small prompts, in order to fix noticeable issues. This is an example of such a prompt:

```
    "Keep in mind the Plant UML code you provided in the previous
answer.  Now change that code so that an order contains a product.
With the product class containing a stock level."
```

It has also not become clear what parts of the model are based on which type of knowledge. These types of knowledge are elaborated upon in Chapter 3. ChatGPT can for example make things up as well as use knowledge in the transcripts. In order to get a better idea of the modelling decisions that ChatGPT makes, the next iteration focused on traceability.

4.2.2 Traceability

Traceability in this case refers to getting an understanding on what knowledge ChatGPT makes certain modelling decisions. Does ChatGPT base everything on the transcripts, does it make up things based on it's own knowledge of such systems, or does it make things up that do not make sense? To try and understand this better, the experimentation evolved to a more structured approach. A domain model was made by the primary researcher based on the transcripts, extra documentation (the vision document), and domain knowledge outside of the two aforementioned sources. After that was done, improved prompts were used in order to generate a model. Then afterwards, both the model could be compared. Now not only ChatGPT's own elaboration of it's output could be used, but also a human model. The structure of the generation process remained the same, one prompt with the first transcript attached, and one prompt with the second transcript attached. The first prompt is as follows:

```
The following txt document contains an interview between business
analysts and client.  They are discussing a software system to be
built.

I want you to model a UML class diagram, in PlantUML code,
that describes the data structure (database) of this system to be
built.  Based on the conversation provided in the txt file.

When you process this prompt, build/extend on the previous
PlantUML code you've provided.

Please take into account the following context when modelling.
The following context is split up into three parts, context on mod-
elling the right classes, context on modelling the right associa-
tions (relationships), and context on keeping the right attributes.
All this context needs to be taken into account.

Now discard unnecessary and incorrect classes according to the fol-
lowing criteria.  Adjust you modelling choices accordingly.

(Example of a guideline) Redundant classes.  If two classes
express the same concept, you should keep the most descriptive
name.  For example, although Customer might describe a person tak-
ing an airline flight, Passenger is more descriptive.  On the
other hand, if the problem concerns contracts for a charter airline,
Customer is also an appropriate word, since a contract might
involve several passengers.  ATM example.  Customer and User
are redundant; we retain Customer because it is more descriptive.

guideline 2
   guideline N

Now discard unnecessary and incorrect associations, using the fol-
lowing criteria.  Adjust your modelling choices accordingly.  As-
sociations between eliminated classes.  If you have eliminated one
of the classes in the association, you must eliminate the associ-
ation or restate it in terms of other classes.  ATM example.  We
can eliminate Banking network includes cashier stations and ATMs,
ATM dispenses cash, ATM prints receipts, Banks provide software,
Cost apportioned to banks, System provides recordkeeping, and Sys-
tem provides security.

guideline 2
   guideline N

Eliminate unnecessary and incorrect attributes with the following
criteria.  Adjust your modelling choices accordingly.

Objects.  If the independent existence of an element is important,
rather than just its value, then it is an object.
For example, boss refers to a class and salary is an attribute.
The distinction often depends on the application.
For example, in a mailing list city might be considered as an at-
tribute, while in a census City would be a class with
many attributes and relationships of its own.  An element that has
features of its own within the given application is a class.

guideline 2
   guideline N

These are some further rules that your output should adhere to.

   • Make sure that there are not isolated classes, i.e.  a class
     should always be related to at least one other class.

   • Make sure that for each of the relationships proper cardinal-
     ities are included.

   • Make sure that attributes are ordered the following way.
```

attribute: type, with attribute being the attribute and type being the datatype.

When you have analyzed the document and have created the output UML code, I want you to elaborate on each component in the model, for the following elements:

- For each of the individual classes explain if it came from the document directly or whether it came from your own knowledge base.
- For each of the individual attributes explain if it came from the document directly or whether it came from your own knowledge base.
- For each individual association (relationship) class explain if it came from the document directly or whether it came from your own knowledge base.
- For each individual cardinality explain if it came from the document directly or whether it came from your own knowledge base.

Please give me the following number as well, to show you have fully taken into account the whole prompt 1465. (requesting a number as a check)

The second prompt is similar, and is like this:

Prompt transcript 2

The following txt document contains an interview between business analysts and client. They are discussing a software system to be built.

I want you to model a UML class diagram, in PlantUML code, that describes the data structure (database) of this system to be built. Based on the conversation provided in the txt file.

Keep in mind that this is a follow up interview to the other file I provided earlier. So when you process this prompt, build/extend on the previous Plant UML code you've provided.

Please take into account the following context when modelling. The following context is split up into three parts, context on modelling the right classes, context on modelling the right associations (relationships), and context on keeping the right attributes. All this context needs to be taken into account.

(Same guidelines as the previous prompts) Please give me the following number as well, to show you have fully taken into account the whole prompt 1959. (requesting a different number as a check)

The improved prompts gave a better model as output. The following examples highlight the improvements: there is no redundancy when it comes to users, there is only the customer, and other users are left outside of the scope. This makes it less prone to mistakes, since there is only a single relationship going to each of the classes. Furthermore, the entities that have been modelled are all directly related to the transcripts. This shows that the model has improved. Lastly, all relationships are named and have corresponding cardinalities. In earlier versions this was not always the case. The most important change that was made, is the inclusion of background information and guidelines, on how to properly create a domain model. By including this, the modelling guidelines that are provided to ChatGPT are similar, even across different chats. For all the prompts that included either a transcripts, or the vision document, these guidelines where provided. This way all the modelling is done based on the same guidelines. The final model for one of the cases can be seen in Figure 4.3. The manually made model can be found in Figure 4.2.

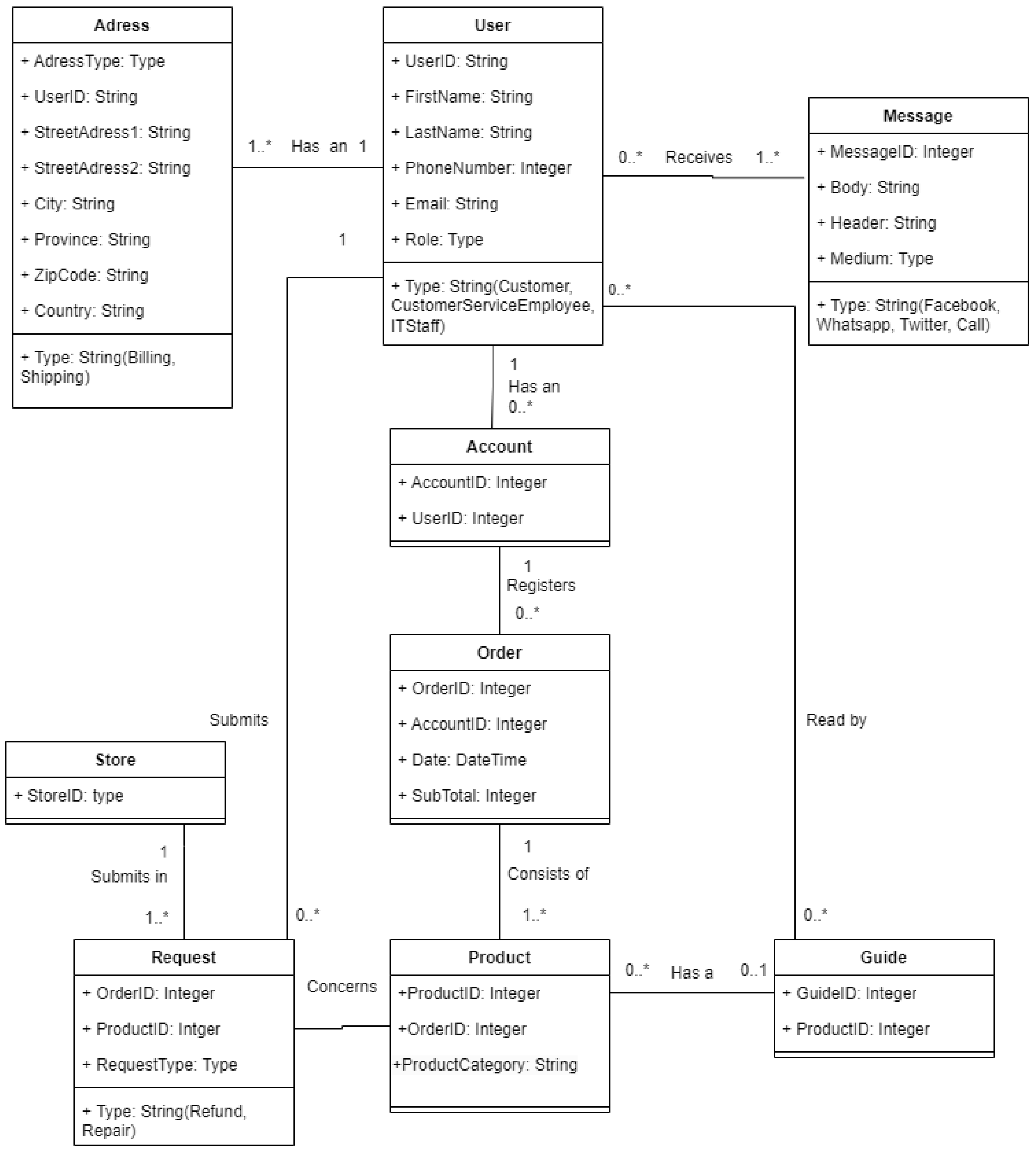


Figure 4.2: A manually made domain model to use for comparison.

In Figure 4.3, the generated model can be found. This model was generated by using the improved prompts as found in Appendix B in Section B.6.

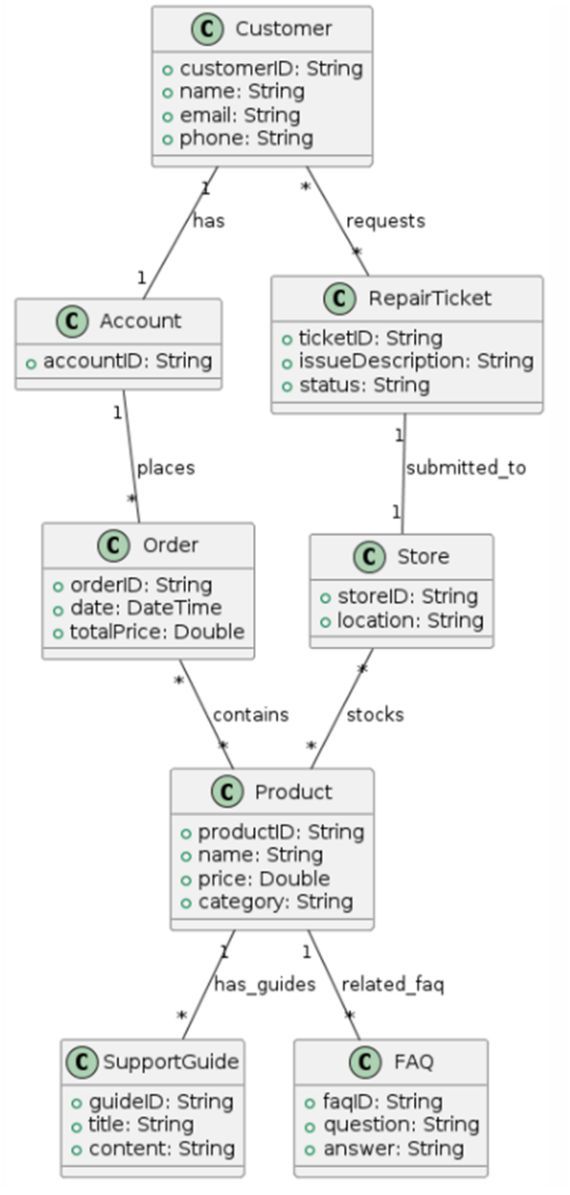


Figure 4.3: A generated domain model.

Upon comparing both models, the primary issue with the generated model was that it did not reflect the data structure for a system to-be sufficiently. In professional use, a single user class is modelled, and roles are added to that class in order to differentiate the customer and the employee for example. Another example is the inclusion of the order, stock, and orderItem classes. In order to simplify the model, these could be abstracted, for example, by adding a stockLevel attribute to the product class. Ideally a generated model should be of the quality that it could be uploaded into a modelling tool and used straight away, with adjustments only to be made because of taste or case specific differences. As of this point in the initial experimentation, generated models were not on that level of quality yet. They could be useful as a reminder of what the interview was about, but they were not useful as a starting point of modelled data structure. This means that the models were not of a quality that was directly usable to implement into a modelling tool, without significant intervention of a human to revise the model. To try and get closer to such a model. The next step was to normalize the models to represent a data structure better.

4.2.3 Normalization

By adding two extra prompts, the output generated by the improved prompts handling the first and second transcript, evolved into something resembling the domain model of a data structure better. A third prompt was also added in, in order to include the information from the vision document. The prompt for including the vision document in similar to the previous two prompts, and is as follows:

```
    The attached document contains a vision statement for a system
to be built.  It contains the wishes of the customer of the soft-
ware company who is going to build the system.

I want you to model a UML class diagram, in PlantUML code, that
describes the data structure (database) of this system to be
built.  Based on the vision document provided in the file.

Please take into account the following context when modelling.
The following context is split up into three parts, context on
modelling the right classes, context on modelling the
right associations (relationships), and context on keeping the
right attributes.
All this context needs to be taken into account.
(Same guidelines as the previous prompts)

    Please give me the following number as well, to show you
have fully taken into account the whole prompt 1234.
```

This prompt is applied before the two previous prompt mentioned before, and makes use of the vision document that is analyzed. After the prompt for the vision document, and the prompts for the two transcripts are applied, there are two further normalization prompts that are applied after the documents have been handled. The first one of these has a set of guidelines on how to normalize models to be more uniform. For example, making sure that the naming conventions are the same across the models, or making sure the right data types are included. The second prompt addresses some of the remaining issues, like forgetting to model cardinalities, or not naming relationships.

normalization prompt 1 `I want you to refine the previously generated PlantUML code.`

```
My goal, and what you should try and give me as output, is to have
the code of model that captures the data structure and the appli-
cation logic of the system discussed in the previously analyzed
interview transcripts.
I want the model you provide to be usable as the blueprint for a
```

database, that can you used during the software development process.

To do so, I will give you guidelines.
The guidelines aim to ensure that developers set up high quality data models.
High quality means: Logical, Consistent, Well structured and, Unambiguous.
I want you to consider all of the following categories of modelling guidelines.

- Data Modelling: The first step is making a distinction between the following:
  Strong entities, Weak entities, Link tables, and inheritance tables.
  An entity is strong when its existence does not depend on the existence of any other entity in a database.
  For example, sales_order, or sales_invoice.
  A weak entity depends on a strong(er) entity for its existence.
  For example, sales_order_line, or sales_invoice_line.

  A link table maps two or more tables together by referencing the primary key (PK) of each data table.
  In effect, it contains a number of foreign keys (FK), each in a many-to-one relationship from the link table to the individual data tables.
  The PK of the link table is typically composed of the FK columns themselves. For example: customer_product, linking customer, and product. Or, company_group_company linking company_group, and company. Sometimes there are multiple types of entities which have certain attributes or relations in common. Using "sub-type" tables is a simple way to implement table inheritance in SQL Server.
  For example: Company, with customer, supplier, debtor, and creditor.

  In the following section Strong and Weak entities form the basis of many of the Guidelines.

- Strong Entity:
  A Strong entity, has 1 primary key column ,and does not have foreign key columns in the primary key.

- Weak Entity:

1. A weak entity, has more than 1 primary
   key column.
2. The primary key columns are ordered
   from strong to weak.
3. The last primary key column is not
   a foreign key, the other primary key columns are.
4. Has exactly 1 primary column more
   than his 'parent'.
5. Has the name of its 'parent' plus
   an addition.

- Link Table:

1. Link tables have all primary key
   columns from both source tables.
2. All primary columns are also foreign
   keys
3. Have the name of one table combined
   with the name of the other table, if another name is more
   suitable, then this table should probably be a Strong
   or Weak Entity.

- Inheritance Table (One to One Relationship
  (1 : 0..1)):

1. Target table has his own name. This
   means that his name is not derived from the 'parent'
   entity.
2. The number of primary key columns is
   equal to the 'parent'.
3. All primary columns are also
   foreign keys.

- Foreign Key Relation:

1. Each foreign column has its own reference,
   so if a table has 5 foreign key columns, there
   must also be 5 references, and a column may be
   part of multiple references.
2. The database check must be enabled
   for every reference, there are situations
   where the check is not allowed, for example
   with a reference to a view. Then the check can be
   turned off.

- Recursive Relation:
  Last foreign key column has a new name: the column does not have a derived name.

- All entities:

  1. The name must be self-explanatory
  2. Names singular
  3. Names lowercase
  4. No abbreviations
     (a) Unless platform limits are exceeded
  5. Divide a name into small words (subnames). Place an underscore between the words.
     (a) Good: sales_order_line
     (b) Not good: salesorder_line, or salesorderline
  6. No meta information in names

- Columns:

  1. The name must be self-explanatory
  2. Names lowercase
  3. No meta information in names
  4. Divide a name into small words (subnames). Place between the words an underscore.
  5. No abbreviations, except no, and id.
  6. No table name in the column for non-key columns

- Primary Keys:

  1. Name primary key column is table name + _id.
  2. Type of column is preferably an identity with an INT as the data type. For tables with more than 2 billion expected rows or data mutations, use BIGINT. Only primary key columns that are not foreign keys may be an identity column.

- Foreign Keys:
  1. The referring foreign key column has the same name as the primary key of the parent table. If the corresponding reference has an addition, then this addition must also be added in front of the column name.
- Domains:
  1. No DTTP in the name, except XML, DATE, or IMAGE
  2. No length in the name
  3. No meta information in name
  4. Primary key columns which are not foreign key columns have the same domain name as the column.
- Datatypes:
  1. Use DATETIME2 instead of DATETIME
  2. Use NVARCHAR instead of VARCHAR, unless performance is a key factor
  3. Use INT or BIGINT for identities
  4. Use NUMERIC for numbers with digits after the decimal point
  5. Don't use CHAR, FLOAT, NCHAR without a specific reason

When you have analysed the document and have created the output UML code, I want you to elaborate on each component in the model, for the following elements:

- For each of the individual classes explain if it came from the document directly or whether it came from your own knowledge base.
- For each of the individual attributes explain if it came from the document directly or whether it came from your own knowledge base.
- For each individual association (relationship) class explain if it came from the

document directly or whether it came from your own knowledge base.

- For each individual cardinality explain if it came from the document directly or whether it came from your own knowledge base.

normalization prompt 2 I am not quite satisfied with the model (the PlantUML code) output you've just provided. I will give you a list of things I want you to change, please incorporate all items on that list, and provide met with the improved PlantUML code and elaborate on all the changes you've made.

1. Remove all isolate classes (classes that are not connected to any other class, OR classes that are connected to other classes but are separate from another group of connected classes that is bigger)

2. Make sure to abstract classes as much as possible. For example if you model a customer and 2 types of employees, model them as one class called user. All three, albeit different, are users of the system. Make sure that it makes sense to do this. In some cases the customer may not interact with/use the system, but still be important to model. In such a case a customer should be modelled as a separate class, and not as a user.

3. An address should be a class not an attribute, a user can have more than one address, for example a billing address and a shipping address. Important: Only in cases where it makes sense to have two different addresses, for example with an online store where you order things, it makes sense to have two different addresses. Make sure to think about this.

4. Make sure that all the relationships are named properly, and all the cardinalities make sense.

5. Make sure to not mess with
   other parts of the model that are good already.

6. Make sure that each
   of the classes that you model are filled with all the
   possible attributes
   that make sense for that class. Give me as many attributes
   as possible,
   within reason.

Give me the sum of all the different
things in the list that I have provided, to show that you've
incorporated all the changes that I want you to make
to the model.

To reiterate the what I want you
to do once more, make sure you improve the model (PlantUML code)
you most recently provided, based on the things I have provided in
the list above.

When you have analyzed the document
and have created the output UML code, I want you to elaborate
on each component in the model, for the following elements:

- For each of the individual
  classes explain if it came from the document directly or
  whether
  it came from your own knowledge base.

- For each of the individual
  attributes explain if it came from the document directly or
  whether it
  came from your own knowledge base.

- For each individual association
  (relationship) class explain if it came from the document
  directly or whether it came from your own knowledge base.

- For each individual cardinality
  explain if it came from the document directly or whether it
  came from your own knowledge base.

Good luck!

In order to illustrate the evolving nature of the domain throughout the generation process, a full example will be provided. The process to get to the final model was as follows: run the first three prompts with the accompanying vision document and transcripts attached, run the two normalization prompts, insert the final PlantUML code into the PlantUML web server demo. The visualization of this final output was used to compare with the manually made model.

The difference between the model generated only with the improved prompts, and the model after the normalization prompts have been applied, is quite remarkable. With the normalization prompts, the goal was to improve the generated output, to be closer to a model that would be usable in professional use. E.g. a model that would need less human intervention, before it would be sufficiently representing a data-structure of a system-to-be. In Figure 4.3 and in Figure 4.4, a direct comparison can be seen between the models with Figure 4.3 not having been generated with the normalization prompts, and Figure 4.3 generated with the normalization prompts. It becomes clear that for example the customer class, has been abstracted to the user class. This is a more common phenomenon in professional use. This is a good practice because it simplifies the data structure, and mitigates the risk for redundancy. For example instead of keeping separate tables for each type of user, only one type of users column is needed. By creating an extra row in the user table, for example a role column, you'd already be able the distinguish the different types of users. Furthermore, it can be seen that the classes have been extended by the addition extra attributes.

4.2.4 Prompts

In the previous sections of this chapter, the prompts have been mentioned. Throughout the initial experimentation phase, the prompts have been improved through various iterations. The initial approach to creating a model, was to use prompts to build a model, e.g. using 5 prompts to create a model of the first transcripts, and using 5 prompts to improve the model based on the information in the second prompt. The 5 prompts formed a restriction in order to limit the amount human intervention. The goal is to semi-automate the modelling process after all. The issue with this approach is that it is different for each time you create a model. This meant that there was still a lot of human intervention needed. Instead of writing prompts specifically tailored to each of the cases, the prompts were improved to a point where they would be suitable for all different cases, as far as possible. These prompts included guidelines. In the prompts as shown above, only a few examples have been mentioned, the full guidelines can be found in Appendix D. Lastly, the normalization prompts were applied. These prompts, which also consists of data model normalization guidelines, further enhanced the domain model that the LLM created. Overall, the initial experimentation phase has iteratively improved the prompts. The approach that was taken in the end, was to include guidelines for modelling and normalization. This ensured that the LLM modelled according to the same guidelines, even across different domains.

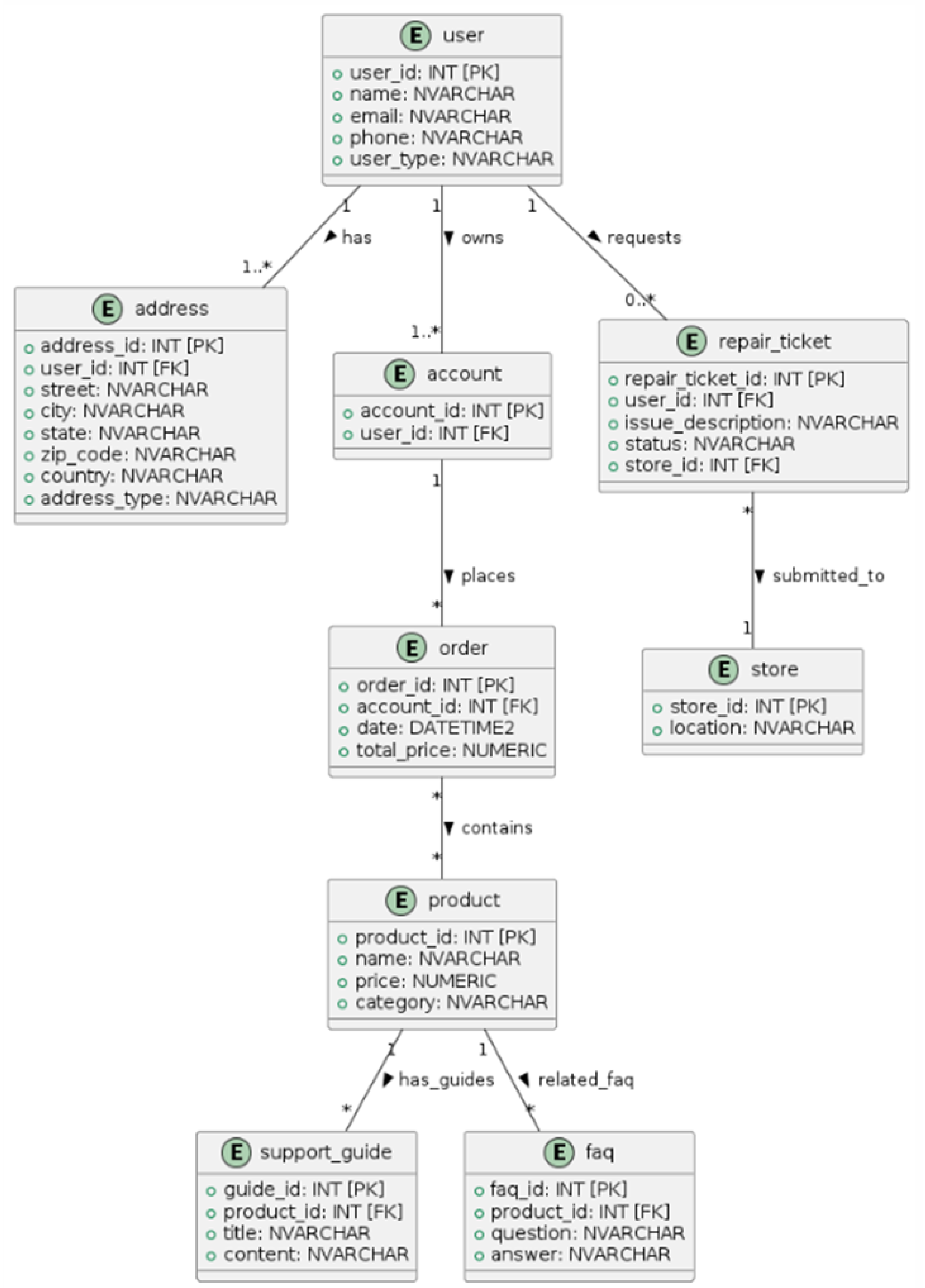


Figure 4.4: A generated domain model using the improved prompts and the normalization prompts.

4.3 Primary experiment

4.3.1 Introduction

This section presents the results of the primary experiment of this research. Individual cases and scores will not be elaborated upon in great detail. All the raw data can be found in Appendix B in Section B.8. It will however go over several distinct perspectives, and showing their results.

This chapter is divided into five distinct sections. First, it covers the results on the four modelling elements: Entities, Attributes, Relationships, and Cardinalities. Lastly, the overall results are presented.

Each section will feature a visualization of the related data. In each of the sections hypothesis will be answered, comparing the performance of the human, and the LLM. In addition to comparing the differences in performance, the agreement between the human and the LLM is analyzed.

There is one important change that was made after the conduction of the experiment. The scenarios, as defined in Chapter 3, were made so that there was a distinction between elements that should have been modelled, for example entities that are direct knowledge, and for example attributes based on domain knowledge. When it comes to comparing performance, it is not fair to include elements that are not mentioned directly in the transcripts/documentation. How can either party be expected to model something if it is not explicitly mentioned? However, these scenarios are included when comparing element creation, since that is about the amount of elements each party creates.

4.3.2 Hypotheses

These are the hypotheses that have been established. It is important to note that the hypotheses are generalized, and will be answered for each modelling element individually. E.g. hypothesis 1 will be answered for each element, and will be denoted as follows, H1E: (Entities), and so on.
Null hypotheses:

$H1_0$: There is substantial agreement between the human and the LLM.

$H2_0$: The human models significantly more usable elements than the LLM.

$H3_0$: The LLM models significantly more unusable elements, than the human.

In H1, substantial means 50% agreement or more.

Alternative hypotheses:

$H1_1$: There is no substantial agreement between the human and the LLM.

$H2_1$: The human does not model significantly more usable elements than the LLM.

$H3_1$: The LLM does not model significantly more unusable elements, than the human.

4.3.3 Agreement H1

To measure the agreement between the human modeler and ChatGPT, only the direct knowledge will be taken into account. This choice was made because we want to measure the agreement based on those inputs that both the human modeler and ChatGPT had access to: the transcripts and the documentation.

For each of the modelling elements, the overall agreement is measured as a percentage. This percentage stands for the percent of modelling elements that both the human modeler, as well as the LLM, have modelled. This is then also converted into an average across all 10 cases. In Table 4.1, the agreement percentages per case can be found. The raw data for each of the modelling elements can be found in Table 4.2, Table 4.3, and Table 4.4.

When it comes to cardinalities, it is not possible to a meaningful analysis. There were no cases where cardinalities were modelled based on direct knowledge. This would mean that the human and the LLM would have 100% agreement. Because of the lack of usable cases, cardinalities are not taken into account any further, when it comes to analyzing the agreement.

Table 4.1: The percentage of agreement per modelling element.

Case/Category	Entities %	Attributes %	Relationships %
1: CRM	57.14	66.67	100.00
2: CRM	66.67	null	50.00
3: CRM	72.73	50.00	100.00
4: CRM	55.56	100.00	100.00
5: Market Place	42.86	null	null
6: ERP	37.50	null	null
7: Market Place	75.00	100.00	null
8: Customer App	55.56	null	null
9: Subscription service	54.55	null	null
10: ERP	50.00	66.67	null
Combined Average	**56.76**	**76.67**	**87.50**
Standard Deviation	**11.42**	**20.00**	**21.65**

Table 4.2: The number of entities based on direct knowledge for each case.

Case/Category	Human	Both	LLM	Total	Agreement %
1: CRM	3	4	0	7	57.14
2: CRM	3	6	0	9	66.67
3: CRM	2	8	1	11	72.73
4: CRM	4	5	0	9	55.56
5: Market Place	4	3	0	7	42.86
6: ERP	5	3	0	8	37.50
7: Market Place	6	6	0	8	75.00
8: Customer App	4	5	0	9	55.56
9: Subscription service	5	6	0	11	54.55
10: ERP	3	4	1	8	50.00
Average	**3.5**	**5.0**	**0.2**	**8.7**	**56.76**
Std. Dev.	**1.02**	**1.48**	**0.40**	**1.35**	**11.42**

Table 4.3: The number of attributes based on direct knowledge for each case.

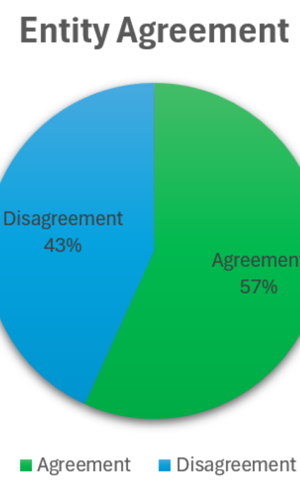
Case/Category	Human	Both	LLM	Total	Agreement
1: CRM	1	2	0	3	66.67
2: CRM	0	0	0	0	null
3: CRM	2	2	0	4	50.00
4: CRM	0	5	0	5	100.00
5: Market Place	0	1	0	1	null
6: ERP	1	0	0	1	null
7: Market Place	0	2	0	2	100.00
8: Customer App	9	0	0	9	null
9: Subscription service	2	0	0	2	null
10: ERP	1	2	0	3	66.67
Average	**1.60**	**1.30**	**0.10**	**3.00**	**76.67**
Std. Dev.	**2.58**	**1.55**	**0.30**	**4.43**	**20.00**

Table 4.4: The number of relationships based on direct knowledge for each case.

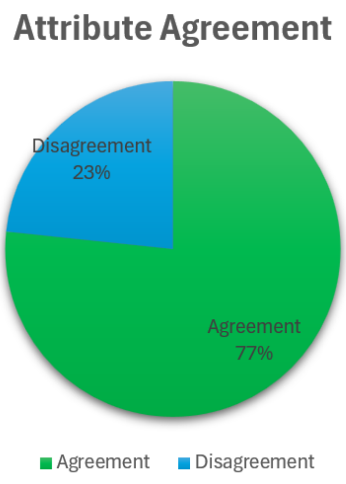
Case/Category	Human	Both	LLM	Total	Agreement
1: CRM	0	2	0	2	100.00
2: CRM	1	1	0	2	50.00
3: CRM	0	2	0	2	100.00
4: CRM	0	3	0	3	100.00
5: Market Place	0	0	0	0	null
6: ERP	0	0	0	0	null
7: Market Place	0	0	0	0	null
8: Customer App	0	0	0	0	null
9: Subscription service	0	0	0	0	null
10: ERP	0	0	0	0	null
Average	**0.10**	**0.80**	**0.00**	**0.90**	**87.50**
Std. Dev.	**0.30**	**1.08**	**0.00**	**1.14**	**21.65**

From what can been seen in Table 4.1, the overall average agreement, for modelling entities, is 56.76%. This indicates that $H1E\ 0$ is accepted, and that there is a substantial agreement between the human and the LLM, when modelling entities. A visualization of this can be found in Figure 4.5.

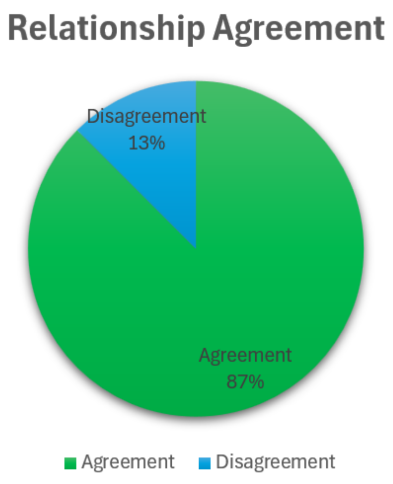
Furthermore, Table 4.1 shows that there is an average agreement of 76.67%, when it comes to modelling attributes. This indicates that $H1A\ 0$ is accepted and the alternative hypothesis is rejected. This means there is a substantial agreement between the human and the LLM, when it comes to modelling attributes. A visualization of this can be found in Figure 4.5.



(a) Agreement and disagreement when modeling entities



(b) Agreement and disagreement when modeling attributes



(c) Agreement and disagreement when modeling relationships

Figure 4.5: Visualizations showing the agreement and disagreement between the human and the LLM when modeling (a) entities, (b) attributes, and (c) relationships.

Continuing, Table 4.1 also shows that there is an average agreement of 76.67%, when it comes to modelling relationships. This indicates that H1*R* 0 is accepted, and the alternative hypothesis is rejected. This means that there is a substantial agreement between the human and the LLM, when it comes to modelling relationships. A visualization of this can be found in Figure 4.5.

Given the averages for each modelling element, *E:* 56.75%, *A:* 76.67%, and *R:* 87.5%, the overall agreement between the human modeler and ChatGPT, is 73.64%. This means that H1*O* 0 is accepted and the alternative hypothesis is rejected. This means that there is a substantial agreement between the human and the LLM. A visualization of this can be found in Figure 4.6.

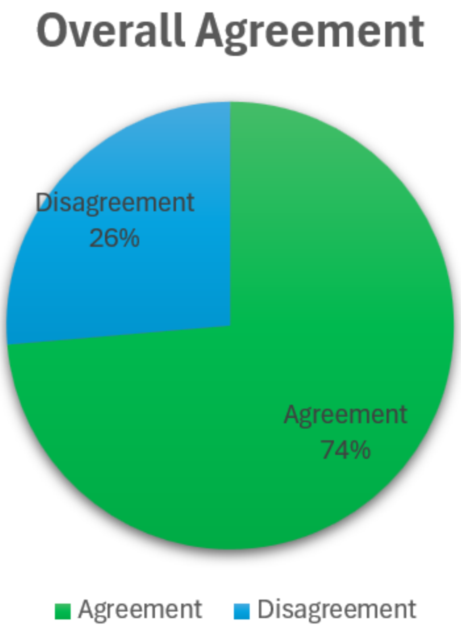


Figure 4.6: A visualization showing the overall agreement and disagreement, between the human and the LLM, when modelling elements.

It is important to note that for attributes and relationships, a smaller amount of cases could be taken into account. Since no agreement or disagreement could be measured. Similarly, the cardinalities have not been taken into consideration. Because in the dataset, there were no cases were any agreement, or disagreement, could be distinguished.

4.3.4 Usable element creation H2

Next to the agreement between the human and the LLM, the usable element creation can be compared. With usable element creation, all the elements modelled using direct knowledge, and domain knowledge, are meant. By comparing the usable element creation between the human and the LLM, it becomes clear what possible differences are between the human and the LLM. Furthermore, it may nuance other results. For example, if it turns out to be the case that the LLM makes a lot more mistakes, but it also creates a lot more elements, it may be an explanation for the amount of mistakes the LLM makes.

For each of the modelling elements, as well as overall, a Wilcoxon Signed-Rank test was done. The Wilcoxon signed-rank test is a non-parametric statistical test, used to compare two related samples or repeated measurements on a single sample to assess whether their population mean ranks differ. This is useful because it cannot be assumed that the data is distributed normally.

The test works by calculating the differences between paired observations, ranking the differences, and then analyzing the ranks of the positive and negative differences. By focusing on the ranks rather than the raw data, the Wilcoxon signed-rank test is able to handle skewed distributions and outliers.

Entities A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating usable entities. In Figure 4.7, a visualization of the distribution of usable entities created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for the human was 8.7 with a standard deviation of 1.25, while the mean score of the LLM was 7.4 with a standard deviation of 1.65. The test results indicated that there were 6 instances where human scores were higher, 2 instances where LLM scores were higher, and 2 ties.

The test statistics showed a Z-value of -1.995 and a p-value of 0.046. Since the p-value is less than 0.05, H2*E* 0 is accepted, and the alternative hypothesis is rejected. This suggests that the human creates a significantly higher number usable entities than the LLM, in this context.

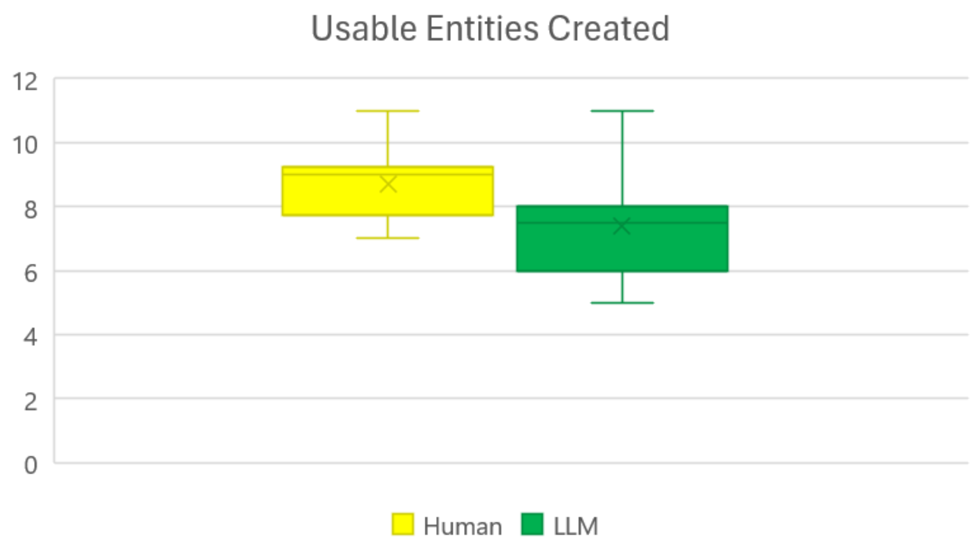


Figure 4.7: A visualization of the distribution of usable entities modelled.

Attributes A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating usable attributes. In Figure 4.8, a visualization of the distribution of usable attributes created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for humans was 4.0150 with a standard deviation of 0.73485, while the mean score for the LLM was 2.9050 with a standard deviation of 0.60421. The test results indicated that there were 8 instances where human scores were higher, 1 instance where LLM scores were higher, and 1 tie.

The test statistics revealed a Z-value of -2.192 and a p-value of 0.028. Since the p-value is less than 0.05, H2*A* 0 is accepted, and the alternative hypothesis is rejected. This suggests that the human creates significantly more usable attributes than the LLM, in this context.

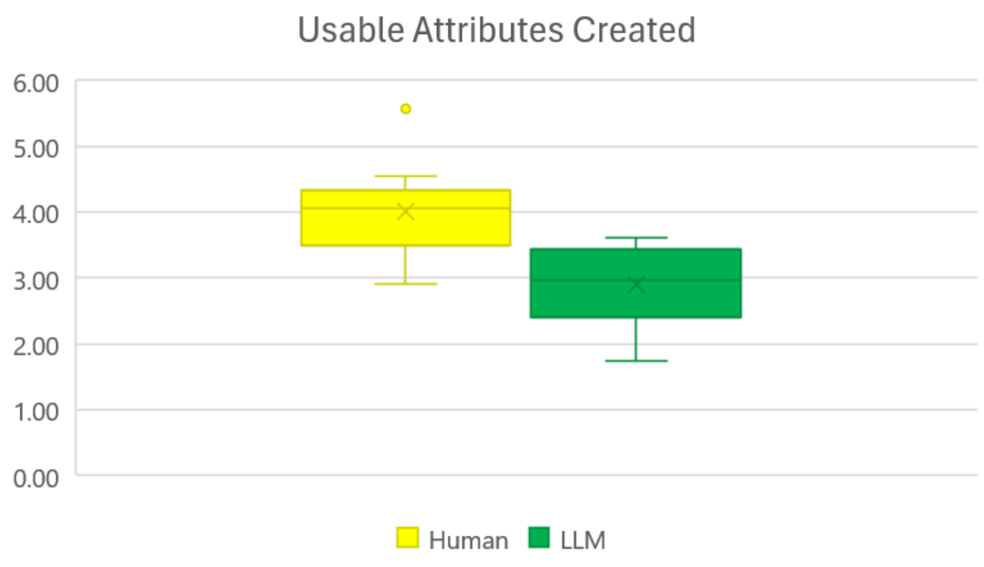


Figure 4.8: A visualization of the distribution of usable attributes modelled.

Relationships A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating usable relationships. In Figure 4.9, a visualization of the distribution of usable relationships created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for humans was 0.9290 with a standard deviation of 0.29153, while the mean score for the LLM was 0.4840 with a standard deviation of 0.09755. The test results indicated that there were 9 instances where human scores were higher and 1 instance where the LLM score was higher, with no ties.

The test statistics revealed a Z-value of -2.601 and a p-value of 0.009. Since the p-value is less than 0.05, H2*R* 0 is accepted, and the alternative hypothesis is rejected. This suggests that the human creates significantly more usable relationships than the LLM, in this context.

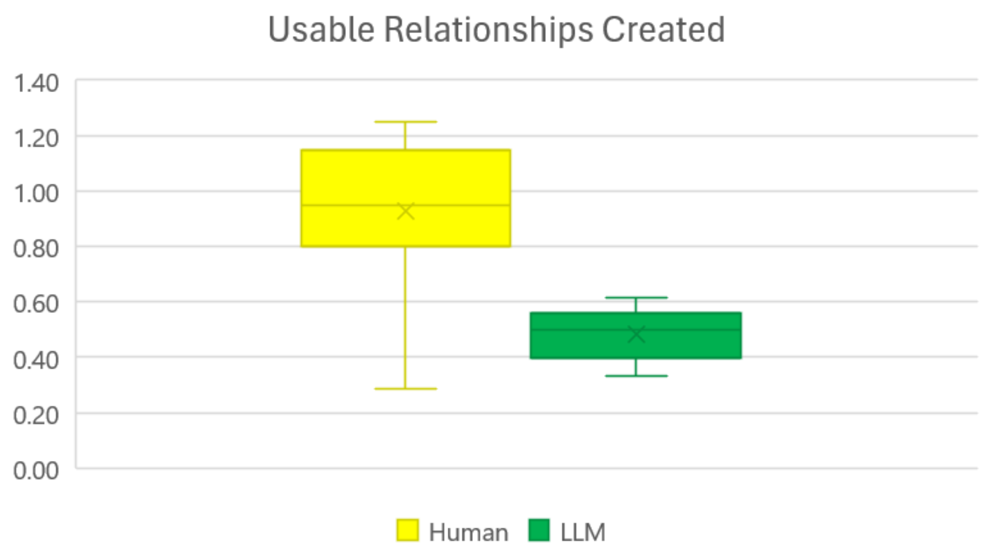


Figure 4.9: A visualization of the distribution of usable relationships modelled.

Cardinalities A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating usable cardinalities. In Figure 4.10, a visualization of the distribution of usable cardinalities created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for humans was 0.8730 with a standard deviation of 0.33609, while the mean score for the LLM was 0.3420 with a standard deviation of 0.12017. The test results indicated that there were 9 instances where human scores were higher and 1 instance where the LLM score was higher, with no ties.

The test statistics revealed a Z-value of -2.701 and a p-value of 0.007. Since the p-value is less than 0.05, H2C 0 is accepted, and the alternative hypothesis is rejected. This suggests that the human creates significantly more usable cardinalities than the LLM, in this context.

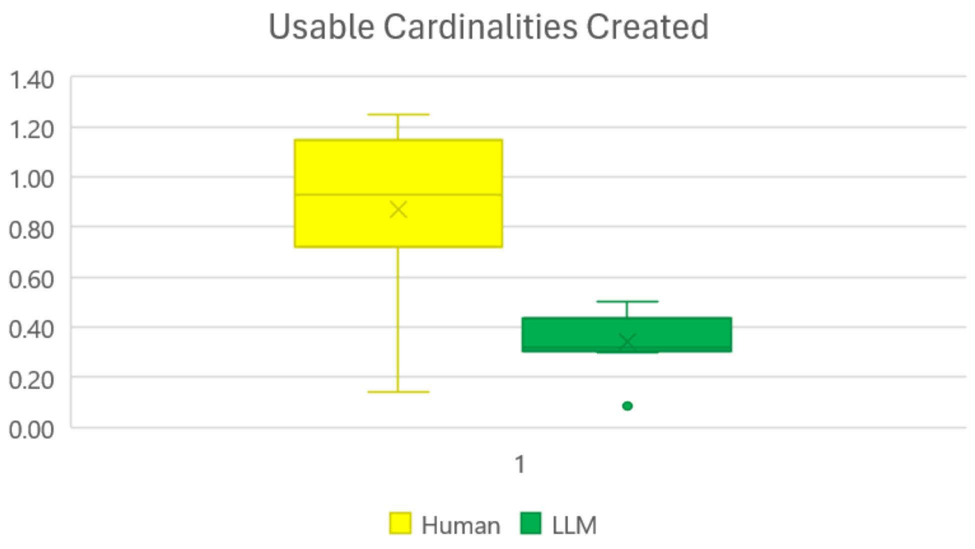


Figure 4.10: A visualization of the distribution of usable cardinalities modelled.

Overall A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating usable elements. In Figure 4.11, a visualization of the distribution of usable elements created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for humans was 14.5160 with a standard deviation of 1.61719, while the mean score for the LLM was 11.1290 with a standard deviation of 1.92374. The test results indicated that there were 8 instances where human scores were higher and 2 instances where LLM scores were higher, with no ties.

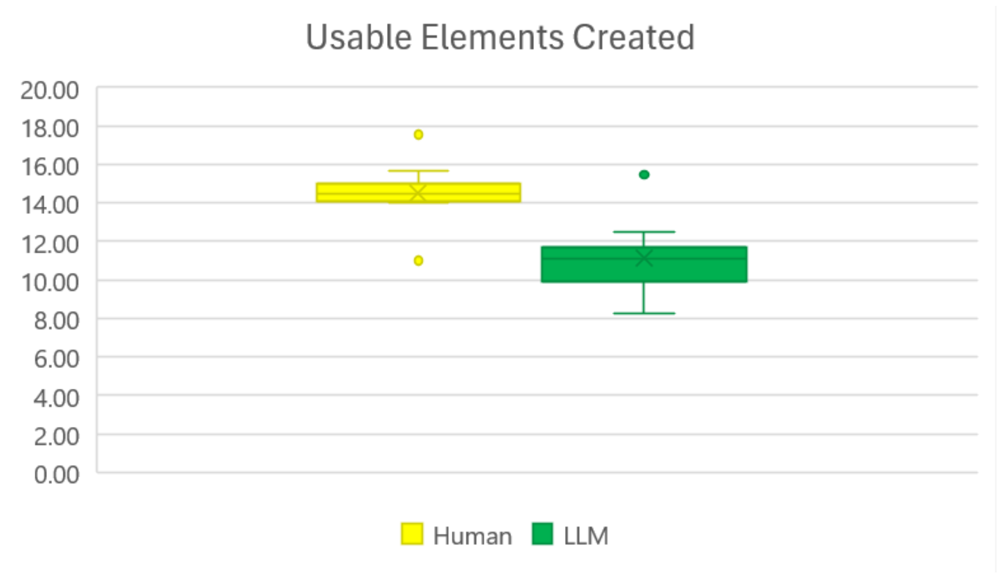


Figure 4.11: A visualization of the distribution of usable elements created.

The test statistics revealed a Z-value of -2.497 and a p-value of 0.013. Since the p-value is less than 0.05, H2O 0 is accepted, and the alternative hypothesis is rejected. This suggests that the human creates significantly more usable elements than the LLM, in this context.

4.3.5 Unusable element creation H3

Finally, the unusable element creation be compared. By taking the numbers of missing knowledge for each of the cases, and comparing the scores of the human and the LLM, it becomes clear who has made more mistakes. For each of the modelling elements, as well as overall, a Wilcoxon Signed-Rank test was done.

Entities A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating unusable entities when modelling. In Figure 4.12, a visualization of the distribution of the unusable entities created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for humans was 0.200 with a standard deviation of 0.42164, while the mean score for the LLM was 4.200 with a standard deviation of 1.31656. The test results indicated that in all instances, the LLM scores were higher than human scores, with no ties. The test statistics revealed a Z-value of -2.842 and a p-value of 0.004. Since the p-value is less than 0.05, H3E 0 is accepted, and the alternative hypothesis is rejected. This suggests that the LLM does make significantly more mistakes when creating entities than the human, in this context.

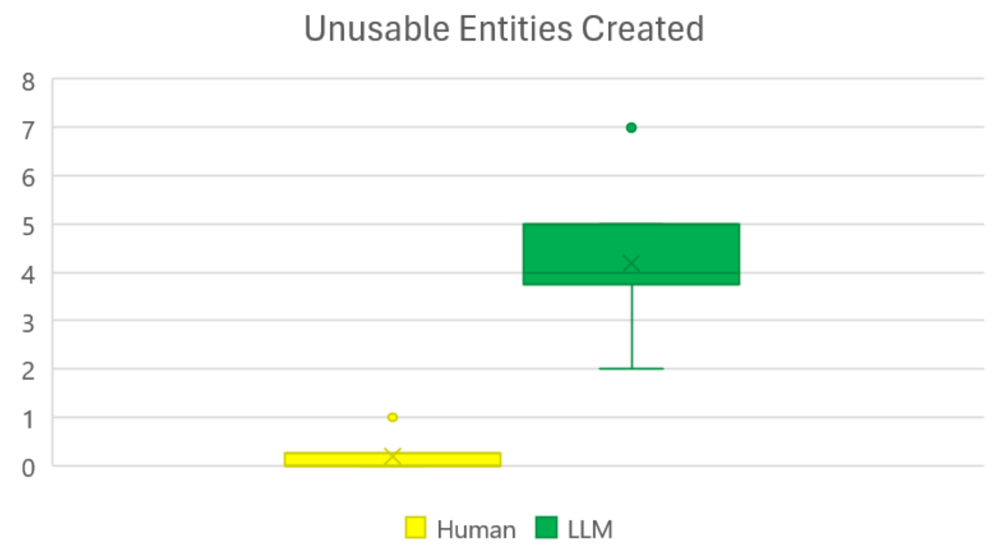


Figure 4.12: A visualization of the distribution of unusable elements created, between the human and the LLM, when modelling entities.

Attributes A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating unusable attributes when modelling. In Figure 4.13, a visualization of the distribution of the unusable attributes created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for humans was 0.0490 with a standard deviation of 0.06385, while the mean score for the LLM was 0.2520 with a standard deviation of 0.29442. The test results indicated that there were 2 instances where human scores were higher, 7 instances where LLM scores were higher, and 1 tie. The test statistics revealed a Z-value of -1.838 and a p-value of 0.066. Since the p-value is greater than 0.05, H3*A* 0 is rejected, and the alternative hypothesis is accepted. This suggests that the LLM does not make significantly more mistakes when creating attributes than the human, in this context.

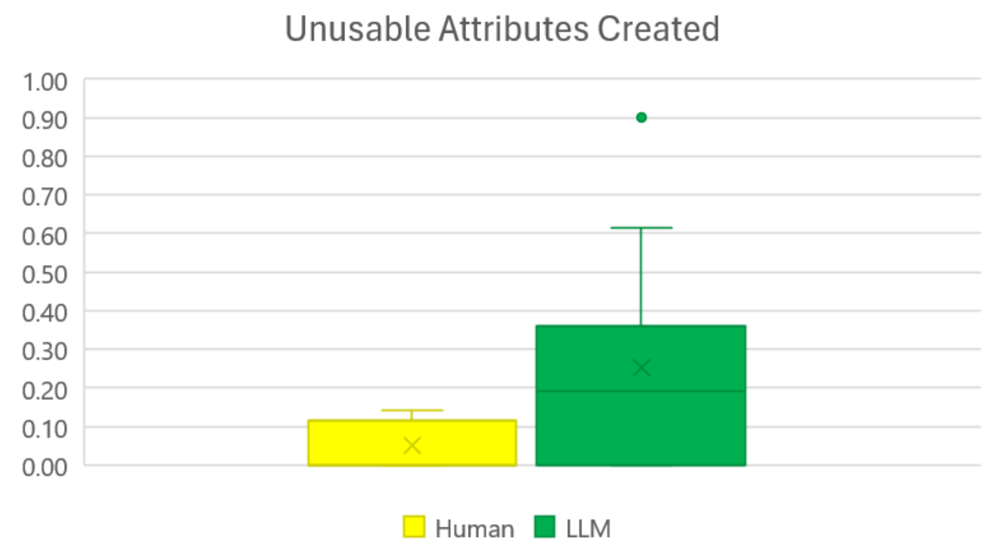


Figure 4.13: A visualization of the distribution of unusable elements created, between the human and the LLM, when modelling attributes.

Relationships A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating unusable relationships when modelling. In Figure 4.14, a visualization of the distribution of the unusable relationships created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for humans was 0.0130 with a standard deviation of 0.04111, while the mean score for the LLM was 0.0830 with a standard deviation of 0.19551. The test results indicated that there was 1 instance where human scores were higher, 2 instances where LLM scores were higher, and 7 ties. The test statistics revealed a Z-value of -1.069 and a p-value of 0.285. Since the p-value is greater than 0.05, H3R 0 is rejected, and the alternative hypothesis is accepted. This suggests that the LLM does not make significantly more mistakes when creating relationships than the human, in this context.

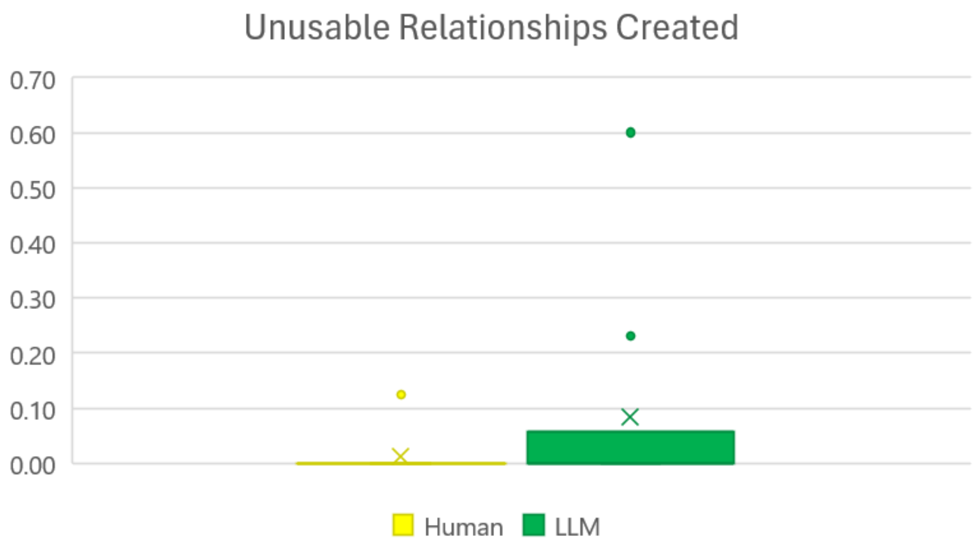


Figure 4.14: A visualization of the distribution of unusable elements created, between the human and the LLM, when modelling relationships.

Cardinalities A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating unusable cardinalities when modelling. In Figure 4.15, a visualization of the distribution of the unusable cardinalities created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for humans was 0.0640 with a standard deviation of 0.07321, while the mean score for the LLM was 0.1770 with a standard deviation of 0.15755. The test results indicated that there was 1 instance where human scores were higher, 7 instances where LLM scores were higher, and 2 ties. The test statistics revealed a Z-value of -2.100 and a p-value of 0.036. Since the p-value is less than 0.05, H3C 0 is accepted, and the alternative hypothesis is rejected. This suggests that the LLM does make significantly more mistakes when creating cardinalities than the human, in this context.

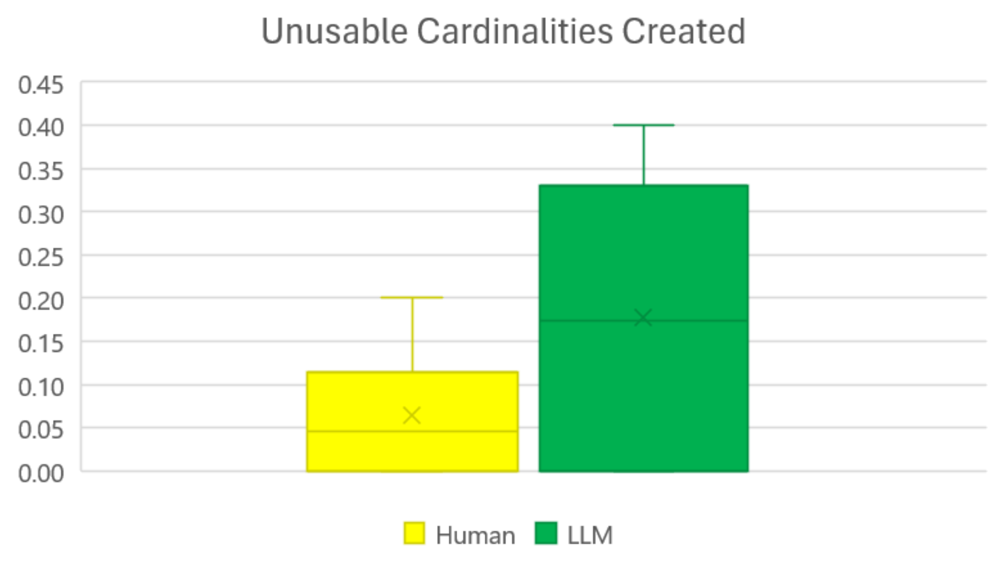


Figure 4.15: A visualization of the distribution of unusable elements created, between the human and the LLM, when modelling cardinalities.

Overall A Wilcoxon Signed-Rank test was conducted, in order to compare the performance of the human and the LLM, when it comes to creating unusable elements when modelling. In Figure 4.16, a visualization of the distribution of the unusable elements created, by the human and the LLM, can be found. The descriptive statistics showed that the mean score for humans was 0.3250 with a standard deviation of 0.44826, while the mean score for the LLM was 4.7110 with a standard deviation of 1.20585. The test results indicated that in all instances, the LLM scores were higher than human scores, with no ties. The test statistics revealed a Z-value of -2.803 and a p-value of 0.005. Since the p-value is less than 0.05, H3*O* 0 is accepted, and the alternative hypothesis is rejected. This suggests that the LLM does make significantly more mistakes when creating elements than the human, in this context.

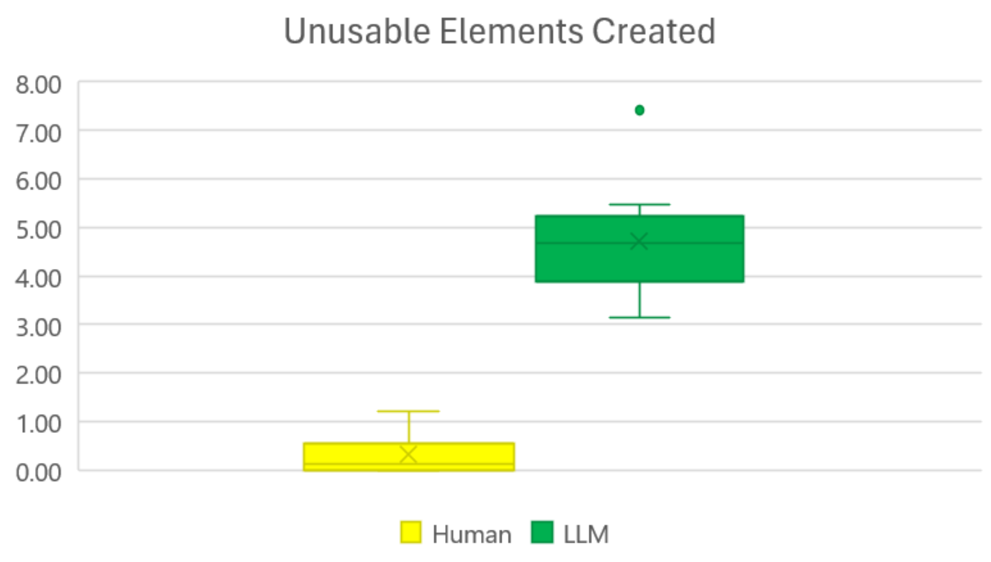


Figure 4.16: A visualization of the distribution of unusable elements created, between the human and the LLM, when modelling.

Chapter 5

Discussion

5.1 Discussion

5.1.1 Key findings

The statistical tests that were performed on the data suggest that there are significant differences in the performance of humans and the LLM when deriving domain models from requirements elicitation conversations.

The analysis identifies that the human modeler created significantly more usable entities, attributes, relationships, and overall usable elements compared to the LLM. Specifically, the human had higher mean scores and lower standard deviations, indicating more consistent performance. The Wilcoxon Signed-Rank tests consistently indicate the the human outperforms the LLM.

Furthermore, the LLM was found to make significantly more mistakes in creating unusable elements. The statistical test indicate that the LLM made significantly more mistakes when it came to modelling unusable entities and cardinalities. It also indicated that there was no significant difference when it came to modelling unusable attributes and relationships. Overall, the statistical test indicates that the LLM made more mistakes than the human.

These results highlight some of the current limitations of ChatGPT, when the prompts as stated earlier are applied, in deriving domain models from requirement elicitation conversations. A positive highlight is the agreement between the human and the LLM, when it comes to modelling entities, attributes, and relationships. It is however a bit more nuanced, since the cardinalities could not be taken into account. Furthermore, the attributes and relationships percentages are not based on all 10 cases, on account of some cases not having occurrences of elements modelled based on direct knowledge. Overall, the results seem to suggest that the LLM is not yet capable to create quality domain models without a human being in the loop.

5.1.2 Interpretation & Validity

The results will be further interpreted. For each of the subsections of the primary experiment, a more in depth interpretation of the results will be given. In addition to interpreting the results, threats to the validity of the research will also be taken into consideration. This gives a more nuanced perspective of the results.

Because of the exploratory nature of this research, there is no substantial foundation of other research projects that have conducted a similar experiment. It is therefore rather difficult to say whether these results are similar to previous work. Such comparisons are therefore not made in the sections below.

Agreement

What becomes apparent from the results in Chapter 4, is that there is a substantial agreement between the human and the LLM, when it comes to modelling entities, attributes and relationships. Despite the substantial agreement between the human and the LLM when it comes to modelling entities, attributes, and relationships, there is still a significant amount of work that needs to be done. This means that a human would have to remain closely involved with the generation process, and possibly iteratively improve on the outputs, in order to get a more agreed upon domain model. This is consistent with current work [27].

One of the main factors that might have affected the agreement between the human and the LLM, is the "strictness" of the researcher, when it comes to matching a certain modelling element with a scenario. In the case of this research, elements did not need to match exactly, meaning that for example a phoneNumber attribute is equal to a telephone attribute. Another factor that might have influenced the amount of agreement is the abstraction level of the model itself. If for example the human and the LLM model on a different level of abstraction, it might occur that certain attributes for example, are not modelled by the LLM.

Usable element creation

In Chapter 4, it became apparent from the statistical tests that the human creates more usable elements than the LLM. One of the explanations of why this could be the case, has to do with the creative aspect of domain modelling. It is definitely up to the human modeler to decide how elaborate a model is going to be. One modeler might choose to model everything that comes up, while another modeler might strictly constrain themselves to what originates directly in the transcripts/documentation. While in the case of this research a similar level of detail has been applied for all cases, changing the modeler and conducting the same experiment would definitely yield different results.

Unusable Element creation

In Chapter 4, it became apparent from the statistical tests that the LLM makes significantly more mistakes when modelling entities and cardinalities. However, the fact that the LLM makes significantly more mistakes, shows that in order to have generated models be closer to a human made model, the human still needs to be involved more to correct during the generation process. The need for the human to remain involved with the LLM, is something that has become apparent from previous research [27]. Whilst there is no significant difference in mistakes made when modelling attributes and relationships, the LLM does make significantly more mistakes overall, than the human.

A big factor, that changes the results if it where changed, are the prompts that one could try and use when generating the domain models. This goes for all the results mentioned above. The possibilities for prompts are almost endless. The results are therefore definitely dependent on the prompts that someone might choose to use. A specific example of such case is the inclusion of the guidelines for domain modelling. These could be exchanged for other guidelines in future work. Doing so would most certainly influence the generated models.

5.1.3 Recommendations for implementation

The findings from this study highlight the potential of LLMs in supporting domain model derivation from requirements elicitation conversations. Practically, it might be possible to apply the findings in several ways:

Focusing human effort LLMs can serve as valuable tools for business analysts and domain modelers by generating initial drafts of domain models. These preliminary models can significantly reduce the time and effort required for the manual creation of models, allowing human experts to focus on refining and validating the models. This hybrid approach combines the strengths of both human intuition and machine efficiency, leading to a quicker initial model, and more focus on model refinement.

Training The results of the experiment indicate that the model generated by the LLM are mostly not of a similar quality as the models created by the human. An interesting implementation of the approach, like the one taken in this thesis, is to use it when training people to model a domain. By letting students or business analysts work with ChatGPT for example, they not only familiarize themselves with using a LLM to create domain models, but they also learn to recognize mistakes and modelling patterns. This could possibly lead to a better understanding of domain modelling.

Tool integration Software development tools and platforms can incorporate LLMs to assist in the early stages of system design. Integrating LLM support into a modelling software for example, might lead to a more efficient and less error-prone domain model creation process. This can be particularly useful in areas where rapid iteration and prototyping are essential, such as an agile development environment.

5.1.4 Summary

This thesis explored the potential of LLMs in supporting the derivation of domain models from requirements elicitation conversations. The key findings indicate that while LLMs can assist in generating initial domain models, they are not yet capable of matching human performance in terms of usable and unusable element creation. Specifically, the human created more usable elements, while the LLM made more mistakes.

The experiment that was conducted highlighted significant differences in performance between the human and the LLM, showing the necessity of human involvement in refining and validating the models generated by LLMs. The effectiveness of LLMs is also heavily influenced by the quality of prompts used. Furthermore, the impact that the human modeler has on the models it produces, also influences the results.

Several limitations were noted, including the volatile nature of LLMs, generalizability issues due to specific datasets, ethical considerations, and resource constraints. Despite these limitations, the modelling approach could be useful when applied into a practical setting, such as initial model generation to reduce manual effort, training tools for domain modeling, and integration into modelling tools.

Future research directions include improving prompt engineering, validating results in practical settings, addressing privacy concerns, and exploring multi-modal inputs to enhance the reliability and applicability of LLMs in domain model derivation.

Chapter 6

Conclusion

In this chapter the conclusion of the thesis will be given. The results and the discussion of the results, have given an in depth look at the results. What will follow is a more high level, more general view of what the findings of this research mean. The relations between these key findings, with the research aims and questions, will be summarized. Furthermore, the value and contributions of these key findings will be discussed. Continuing, the limitation of this research will be addressed, and recommendation for future research will be given.

6.1 Overall findings in relation to the research aims

6.1.1 Research aims and questions

This thesis aimed to explore the potential of LLMs in supporting the derivation of domain models from requirements elicitation conversations. The primary research question was: *"What is the potential of Large Language Models in supporting the derivation of domain models from requirements elicitation conversations, and what factors influence their effectiveness in this context?"*

To address this overarching question, the study focused on several sub-questions:

- **SQ1:** What are the best practices for designing an LLM-based approach that supports domain model derivation from requirement elicitation conversation transcripts?
- **SQ2:** What is the effectiveness of the LLM compared to the human, when deriving domain models from requirement elicitation conversation transcripts?
- **SQ3:** What is the similarity between the LLM and the human, when deriving domain models from requirement elicitation conversation transcripts?

Through performing the experiment, several best practices have been identified. (SQ1) One of the most important ones is the inclusion of modelling guidelines in the prompts. By adding modelling guidelines, the LLM is consistently modelling elements in a way as defined by the human. This decreases the need for human intervention. This also goes for the use of the normalization prompts. The normalization prompts are there to "normalize" the generated output towards a certain desired outcome. For example, enforcing the use of a certain naming convention, or specifying the data types. By adding in these prompts, the generated domain model can be adjusted in way that the human prefers.

In terms of effectiveness, the LLM was found to be less effective than the human. When it comes to usable element creation, the LLM was found to create a significantly lower amount of usable entities, attribute, relationships, and cardinalities.(SQ2) While this does not necessarily mean that the usable element creation of the LLM is bad, it shows that there is still a considerable need for human intervention if the generated models need to be of the same quality.

Continuing, the results show that the LLM makes significantly more mistakes than the human. The LLM creates a significantly higher amount of unusable entities and cardinalities. This further underlines the point made earlier; there is still a considerable amount of human intervention needed if the generated models are to be of a similar quality than that of the LLM. (SQ2)

In terms of similarity, the results showed that there was a substantial agreement between the human and the LLM, when it came to modelling entities, attributes, and relationships.(SQ3) This means that modelled more than 50% of the elements similarly. However, not all 10 cases could be taken into account for attributes and relationships, and the cardinalities did not have a usable amount of data to be taken into account. This means that although the results indicate a substantial agreement between the human and the LLM, further human intervention is needed to increase the agreement.

The answer to the main research question; LLMs demonstrate potential in supporting the derivation of domain models from requirements elicitation conversations, by producing models comparable to those created by human analysts. Initial experimentation shows that the effectiveness of an LLM is influenced by factors such as the quality of prompt engineering, and the type of learning approach used (zero-shot, one-shot, or few-shot), but further research needs to be conducted to properly assess the effect. Despite being less effective when it comes to creating usable elements, and creating more unusable elements, the LLM shows potential to be of value when used in a semi-automated approach. Continuing, the similarity between the human and the LLM, when deriving domain models, seems to suggest that the generated models are of a sufficient quality to use them as a starting point. Overall, LLMs hold the potential to enhance efficiency and accuracy in deriving domain models, if properly managed.

6.1.2 Broader implications

The results of this study have broader implications for the field of RE. The demonstrated potential of LLMs in supporting domain model derivation tasks highlights their potential value as a tool for improving efficiency and accuracy in the domain modelling process. By identifying factors that influence the effectiveness of LLMs in this context, this research could contribute to the further research into the application of LLMs in the field of RE.

6.2 Contributions

The contributions that this research has to the field are not yet proven in practice. Due to the exploratory nature of this research, the results have only been validated in a "laboratory" type setting, meaning a controlled environment and not in practice. It cannot definitively say that its results show that there are significant improvements, that can immediately be applied in practice. It does however confirm the existing notion that a LLM, when applied to a task, will most likely need human intervention in order to be of a similar quality as a human performed task.

Despite this, it manages to contribute to the field by opening up this research direction. In the wake of the rapid developments in LLM research, it has made a start on research what potential LLMs may have in the field of RE. It has shown that the LLM capable of performing the domain model derivation task. Despite not being significantly better, and even being worse in several aspects, it does look promising. It would seem that more research this direction, would have the potential to make a more significant impact in how a business analyst performs the domain modelling task.

6.3 Limitations

Despite trying to be a thorough as possible when conducting research, there are as limitations that constrain the project. These are the limitations that need to be kept in mind when assessing the value of the outcome of this thesis.

The volatile nature of LLMs The field of LLMs is of a volatile nature, with new techniques and models emerging frequently. The continuous advancements make it challenging to capture the most recent innovations within the scope of this thesis. Consequently, the findings may not fully capture the latest breakthroughs in LLMs, potentially limiting the relevance of the thesis's outcomes. Meaning that the conclusions drawn here might need to be revisited as newer LLMs and techniques are developed and validated in future research.

Generalizability The empirical validation, of the approach that has been used in this thesis, may prove to be difficult. The experiment relies on specific datasets that may not be representative of all potential use cases and contexts. The dataset is limited in the sense that it only has two interview transcripts and one vision document per case. The interviews were also performed by different people from case to case, meaning that the interview style and questions vary greatly. Factors such as these, as well as varying datasets, evolving LLM architectures, and diverse application contexts, may affect the extent to which the findings can be generalized to other scenarios. It is important to acknowledge that the practical implications derived from this study may be context-dependent and might not be directly applicable to different organizational practices or settings.

Ethical considerations The ethical implications associated with the use of LLMs present another significant limitation. In practice, it might very well be the case that certain conversation or conversation topics, are not suited/allowed, to be uploaded to a third party such as ChatGPT. In practice, companies are often concerned with privacy issues and handling sensitive data. Despite trying to be as thorough as possible, it may turn out to be impossible to implement the approach as used in this thesis, due to these concerns. This could potentially limit the usefulness of the findings in ethically sensitive contexts.

Resource constraints This research is limited due to resource constraints, including time and access to more varied datasets. While efforts have been made to conduct an as thorough as possible experiment, these constraints may limit the depth of analysis in certain aspects. The scope of the study does not cover all potential use cases, and is not able to compare the performance of different LLMs. This means that this research is by no means able to provide a definitive answer as to whether or not LLMs should be used to support the domain model creation process. It can however highlight whether or not it could be a potential direction for further research.

Validation The validation of LLM generated models was conducted under specific experimental conditions, which may not reflect all scenarios in practice. The models still require refinement to correct inaccuracies and misinterpretations, particularly in complex relationships and cardinalities. While the results show some potential of LLMs to generate preliminary models, the outputs often needs substantial human intervention to be practical and accurate. This indicates that while LLMs can be potentially valuable in supporting domain model derivation, they are not yet reliable enough to replace human modelers entirely.

6.4 Recommendations for future research

There are several opportunities for future research that could help understand the potential of LLMs in supporting the domain model derivation task. As has become clear from the research goals, as well as from the literature discussed in Chapter 1, and Chapter 3, there is a large knowledge gap in the field of LLMs. This thesis has taken an even deeper dive into an application of LLMs in the field of RE.

Longitudinal study Another recommendation for future research would be to have a more longitudinal experiment set-up, in which a LLM can be trained on the domain model derivation task. This could improve the LLM's ability to recognize important information in the transcripts/documentation, potentially improving the amount of direct knowledge that the LLM is able to derive.

Prompt engineering enhancement The thesis has shown that the effectiveness of LLMs is heavily influenced by the quality of the prompts used. Future research could focus on developing more sophisticated prompt engineering techniques to improve the accuracy and reliability of the models created by LLMs. Points of interest could be developing prompt patterns specifically for deriving domain models, trying to establish efficient and fast feedback loops, or deriving domain models from visual inputs.

Validating results in a practical setting The current findings are based on specific datasets and experimental conditions that are different from a practical setting. Future research could try to validate these findings across a broader range of datasets, including different domains and more complex systems. This will help in understanding the generalizability of the results and it could possibly identify any domain-specific challenges that may arise.

Addressing privacy and ethics concerns Given the ethical implications of using LLMs, particularly concerning privacy and sensitive data, future studies could try to identify approaches that could eliminate such concerns. Future research could try to focus on establishing guidelines for the responsible use of LLMs, when dealing with potentially sensitive data, such as requirements elicitation conversations. Doing so might help reduce the reluctance of companies when it comes to adopting LLMs in their processes.

Exploring multi modal inputs Future research can also explore the use of multi modal inputs that can be processed by the LLM, such as textual requirements, diagrams, and user feedback through dictation, or images. Researching such an approach could lead to more comprehensive and accurate domain models.

Types of LLMs There are many different types of LLMs, as has become clear in Chapter 2. For future research, it might be worth to consider other types of LLMs. An example would be to try another widely available one, such as Microsoft copilot. Another idea would be to focus on open LLMs. Open LLMs are made accessible to the public for use, modification, and further development. The open nature of such an LLM could improve the transparency of the processes going behind the scenes. Furthermore, such an LLM could be used to collaborate across multiple research projects, potentially allowing to iteratively improving it's outputs.

By addressing these areas, future research might be able to overcome current limitations and help further advance the application of LLMs in RE.

Bibliography

[1] D. Zowghi and C. Coulin, “Requirements elicitation: A survey of techniques, approaches, and tools”, *Engineering and managing software requirements*, pp. 19–46, 2005.

[2] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, “Extracting domain models from natural-language requirements: Approach and industrial evaluation”, in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, 2016, pp. 250–260.

[3] T. Spijkman, F. Dalpiaz, and S. Brinkkemper, “Requirements elicitation via fit-gap analysis: A view through the grounded theory lens”, in *Advanced Information Systems Engineering*, M. La Rosa, S. Sadiq, and E. Teniente, Eds., Cham: Springer International Publishing, 2021, pp. 363–380, ISBN: 978-3-030-79382-1.

[4] T. Spijkman, B. Winter, S. Bansidhar, and S. Brinkkemper, “Concept extraction in requirements elicitation session recordings: Prototype and experimentation”, in *REFSQ Workshops*, 2021. [Online]. Available: `https://api.semanticscholar.org/CorpusID:235467673`.

[5] T. Spijkman, F. Dalpiaz, and S. Brinkkemper, “Back to the roots: Linking user stories to requirements elicitation conversations”, in *2022 IEEE 30th International Requirements Engineering Conference (RE)*, 2022, pp. 281–287. DOI: `10.1109/RE54965.2022.00042`.

[6] T. Spijkman, X. de Bondt, F. Dalpiaz, and S. Brinkkemper, “Summarization of elicitation conversations to locate requirements-relevant information”, in *Requirements Engineering: Foundation for Software Quality*, A. Ferrari and B. Penzenstadler, Eds., Cham: Springer Nature Switzerland, 2023, pp. 122–139, ISBN: 978-3-031-29786-1.

[7] E. Insfrán, O. Pastor, and R. Wieringa, “Requirements engineering-based conceptual modelling”, *Requirements Engineering*, vol. 7, no. 2, pp. 61–72, Jun. 2002, ISSN: 1432-010X. DOI: `10.1007/s007660200005`. [Online]. Available: `https://doi.org/10.1007/s007660200005`.

[8] C. Rolland and N. Prakash, “From conceptual modelling to requirements engineering”, *Annals of Software Engineering*, vol. 10, no. 1, pp. 151–176, Nov. 2000, ISSN: 1573-7489. DOI: `10.1023/A:1018939700514`. [Online]. Available: `https://doi.org/10.1023/A:1018939700514`.

[9] L. M. Cysneiros, J. C. S. do Prado Leite, and J. de Melo Sabat Neto, “A framework for integrating non-functional requirements into conceptual models”, *Requirements Engineering*, vol. 6, no. 2, pp. 97–115, Jun. 2001, ISSN: 1432-010X. DOI: 10.1007/s007660170008. [Online]. Available: https://doi.org/10.1007/s007660170008.

[10] D. Richards, “Merging individual conceptual models of requirements”, *Requirements Engineering*, vol. 8, no. 4, pp. 195–205, Nov. 2003, ISSN: 1432-010X. DOI: 10.1007/s00766-002-0158-5. [Online]. Available: https://doi.org/10.1007/s00766-002-0158-5.

[11] E.-M. Schön, J. Thomaschewski, and M. J. Escalona, “Agile requirements engineering: A systematic literature review”, *Computer Standards & Interfaces*, vol. 49, pp. 79–91, 2017, ISSN: 0920-5489. DOI: https://doi.org/10.1016/j.csi.2016.08.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548916300708.

[12] A. De Lucia and A. Qusef, “Requirements engineering in agile software development”, *Journal of emerging technologies in web intelligence*, vol. 2, no. 3, pp. 212–220, 2010.

[13] K. Elghariani and N. Kama, “Review on agile requirements engineering challenges”, in *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, 2016, pp. 507–512. DOI: 10.1109/ICCOINS.2016.7783267.

[14] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, “A systematic literature review on agile requirements engineering practices and challenges”, *Computers in Human Behavior*, vol. 51, pp. 915–929, 2015, Computing for Human Learning, Behaviour and Collaboration in the Social and Mobile Networks Era, ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2014.10.046. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S074756321400569X.

[15] R. Saini, G. Mussbacher, J. L. Guo, and J. Kienzle, “Towards queryable and traceable domain models”, in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020, pp. 334–339. DOI: 10.1109/RE48521.2020.00044.

[16] R. Saini, G. Mussbacher, J. L. C. Guo, and J. Kienzle, “Automated traceability for domain modelling decisions empowered by artificial intelligence”, in *2021 IEEE 29th International Requirements Engineering Conference (RE)*, 2021, pp. 173–184. DOI: 10.1109/RE51729.2021.00023.

[17] S. Arulmohan, M.-J. Meurs, and S. Mosser, “Extracting domain models from textual requirements in the era of large language models”, in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2023, pp. 580–587. DOI: 10.1109/MODELS-C59198.2023.00096.

[18] W. C. Lena Waizenegger Brad McKenna and T. Bendz, “An affordance perspective of team collaboration and enforced working from home during covid-19”, *European Journal of Information Systems*, vol. 29, no. 4, pp. 429–442, 2020. DOI: 10.1080/0960085X.2020.1800417. [Online]. Available: https://doi.org/10.1080/0960085X.2020.1800417.

[19] W. Standaert, S. Muylle, and A. Basu, "Business meetings in a postpandemic world: When and how to meet virtually", *Business Horizons*, vol. 65, no. 3, pp. 267–275, 2022, ISSN: 0007-6813. DOI: https://doi.org/10.1016/j.bushor.2021.02.047. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007681321000665.

[20] J. Li *et al.*, "Recent advances in end-to-end automatic speech recognition", *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2022.

[21] W. Han, Z. Zhang, Y. Zhang, *et al.*, *Contextnet: Improving convolutional neural networks for automatic speech recognition with global context*, 2020. arXiv: 2005.03191 [eess.AS].

[22] L. Yan, L. Sha, L. Zhao, *et al.*, "Practical and ethical challenges of large language models in education: A systematic scoping review", *British Journal of Educational Technology*, Aug. 2023, ISSN: 1467-8535. DOI: 10.1111/bjet.13370. [Online]. Available: http://dx.doi.org/10.1111/bjet.13370.

[23] B. Meskó and E. J. Topol, "The imperative for regulatory oversight of large language models (or generative ai) in healthcare", *npj Digital Medicine*, vol. 6, no. 1, p. 120, Jul. 2023, ISSN: 2398-6352. DOI: 10.1038/s41746-023-00873-0. [Online]. Available: https://doi.org/10.1038/s41746-023-00873-0.

[24] K. S. Cheung, "Real estate insights unleashing the potential of chatgpt in property valuation reports: The "red book" compliance chain-of-thought (cot) prompt engineering", *Journal of Property Investment & Finance*, vol. ahead-of-print, no. ahead-of-print, Jan. 2023, ISSN: 1463-578X. DOI: 10.1108/JPIF-06-2023-0053. [Online]. Available: https://doi.org/10.1108/JPIF-06-2023-0053.

[25] D. Russo, *Navigating the complexity of generative ai adoption in software engineering*, 2023. arXiv: 2307.06081 [cs.SE].

[26] D. Luitel, S. Hassani, and M. Sabetzadeh, "Improving requirements completeness: Automated assistance through large language models", *arXiv preprint arXiv:2308.03784*, 2023.

[27] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, *Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design*, 2023. arXiv: 2303.07839 [cs.SE].

[28] A. M. HICKEY and A. M. DAVIS, "A unified model of requirements elicitation", *Journal of Management Information Systems*, vol. 20, no. 4, pp. 65–84, 2004. DOI: 10.1080/07421222.2004.11045786. eprint: https://doi.org/10.1080/07421222.2004.11045786. [Online]. Available: https://doi.org/10.1080/07421222.2004.11045786.

[29] A. Sutcliffe and P. Sawyer, "Requirements elicitation: Towards the unknown unknowns", in *2013 21st IEEE International Requirements Engineering Conference (RE)*, 2013, pp. 92–104. DOI: 10.1109/RE.2013.6636709.

[30] A. Ferrari, P. Spoletini, and S. Gnesi, “Ambiguity and tacit knowledge in requirements elicitation interviews”, *Requirements Engineering*, vol. 21, no. 3, pp. 333–355, Sep. 2016, ISSN: 1432-010X. DOI: 10.1007/s00766-016-0249-3. [Online]. Available: https://doi.org/10.1007/s00766-016-0249-3.

[31] K. L. McGraw and K. Harbison-Briggs, *Knowledge acquisition: Principles and guidelines*. Prentice-Hall, Inc., 1989.

[32] J. C. Wetherbe, “Executive information requirements: Getting it right”, *MIS Quarterly*, vol. 15, no. 1, pp. 51–65, 1991, ISSN: 02767783. [Online]. Available: http://www.jstor.org/stable/249435 (visited on 12/06/2023).

[33] G. J. Browne and M. B. Rogich, “An empirical investigation of user requirements elicitation: Comparing the effectiveness of prompting techniques”, *Journal of Management Information Systems*, vol. 17, no. 4, pp. 223–249, 2001. DOI: 10.1080/07421222.2001.11045665. eprint: https://doi.org/10.1080/07421222.2001.11045665. [Online]. Available: https://doi.org/10.1080/07421222.2001.11045665.

[34] M. G. Pitts and G. J. Browne, “Improving requirements elicitation: An empirical investigation of procedural prompts”, *Information Systems Journal*, vol. 17, no. 1, pp. 89–110, 2007. DOI: https://doi.org/10.1111/j.1365-2575.2006.00240.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-2575.2006.00240.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2575.2006.00240.x.

[35] A. Distanont, H. Haapasalo, M. Vaananen, J. Lehto, *et al.*, “The engagement between knowledge transfer and requirements engineering”, *International Journal of Management, Knowledge and Learning*, vol. 1, no. 2, pp. 131–156, 2012.

[36] A. Ferrari, P. Spoletini, and S. Gnesi, “Ambiguity cues in requirements elicitation interviews”, in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 2016, pp. 56–65. DOI: 10.1109/RE.2016.25.

[37] A. Ferrari, P. Spoletini, B. Donati, D. Zowghi, and S. Gnesi, “Interview review: Detecting latent ambiguities to improve the requirements elicitation process”, in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 400–405. DOI: 10.1109/RE.2017.15.

[38] I. Hadar, P. Soffer, and K. Kenzi, “The role of domain knowledge in requirements elicitation via interviews: An exploratory study”, *Requirements Engineering*, vol. 19, no. 2, pp. 143–159, Jun. 2014, ISSN: 1432-010X. DOI: 10.1007/s00766-012-0163-2. [Online]. Available: https://doi.org/10.1007/s00766-012-0163-2.

[39] J. Wiley, “Expertise as mental set: The effects of domain knowledge in creative problem solving”, *Memory & Cognition*, vol. 26, no. 4, pp. 716–730, Jul. 1998, ISSN: 1532-5946. DOI: 10.3758/BF03211392. [Online]. Available: https://doi.org/10.3758/BF03211392.

[40] D. C. Gause and G. M. Weinberg, “Exploring requirements”, *Dorset House*, p. 249, 1989.

[41] S. F. Tjong, M. Hartley, and D. M. Berry, “Extended disambiguation rules for requirements specifications.”, in *WER*, 2007, pp. 97–106.

[42] D. Berry, E. Kamsties, and M. M. Krieger, “From contract drafting to software specification: Linguistic sources of ambiguity”, 2003.

[43] Z. Zhang, “Effective requirements development-a comparison of requirements elicitation techniques”, *Software quality management XV: software quality in the knowledge society*, pp. 225–240, 2007.

[44] J. Goguen and C. Linde, “Techniques for requirements elicitation”, in *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, 1993, pp. 152–164. DOI: `10.1109/ISRE.1993.324822`.

[45] D. Leffeingwell and D. Widrig, *Managing software requirements. a use case aproach*, 2003.

[46] B. Min, H. Ross, E. Sulem, *et al.*, “Recent advances in natural language processing via large pre-trained language models: A survey”, *ACM Comput. Surv.*, vol. 56, no. 2, Sep. 2023, ISSN: 0360-0300. DOI: `10.1145/3605943`. [Online]. Available: `https://doi.org/10.1145/3605943`.

[47] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: `1810.04805 [cs.CL]`.

[48] M. Chen, J. Tworek, H. Jun, *et al.*, *Evaluating large language models trained on code*, 2021. arXiv: `2107.03374 [cs.LG]`.

[49] Y. Chang, X. Wang, J. Wang, *et al.*, *A survey on evaluation of large language models*, 2023. arXiv: `2307.03109 [cs.CL]`.

[50] E. M. Bender, T. Gebru, A. McMillan-Major, and M. Mitchell, “On the dangers of stochastic parrots: Can language models be too big?”, in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT ’21, Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 610–623, ISBN: 9781450383097. DOI: `10.1145/3442188.3445922`. [Online]. Available: `https://doi.org/10.1145/3442188.3445922`.

[51] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need”, in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[52] N. Sabharwal and A. Agrawal, “Bert algorithms explained”, in *Hands-on Question Answering Systems with BERT: Applications in Neural Networks and Natural Language Processing*. Berkeley, CA: Apress, 2021, pp. 65–95, ISBN: 978-1-4842-6664-9. DOI: `10.1007/978-1-4842-6664-9_4`. [Online]. Available: `https://doi.org/10.1007/978-1-4842-6664-9_4`.

[53] Z. Liu, W. Lin, Y. Shi, and J. Zhao, “A robustly optimized bert pre-training approach with post-training”, in *Chinese Computational Linguistics*, S. Li, M. Sun, Y. Liu, *et al.*, Eds., Cham: Springer International Publishing, 2021, pp. 471–484, ISBN: 978-3-030-84186-7.

[54] L. Floridi and M. Chiriatti, “Gpt-3: Its nature, scope, limits, and consequences”, *Minds and Machines*, vol. 30, no. 4, pp. 681–694, Dec. 2020, ISSN: 1572-8641. DOI: 10.1007/s11023-020-09548-1. [Online]. Available: https://doi.org/10.1007/s11023-020-09548-1.

[55] B. Ghojogh and A. Ghodsi, “Attention mechanism, transformers, bert, and gpt: Tutorial and survey”, 2020.

[56] Y. Zhou, A. I. Muresanu, Z. Han, *et al.*, *Large language models are human-level prompt engineers*, 2023. arXiv: 2211.01910 [cs.LG].

[57] J. Gu, Z. Han, S. Chen, *et al.*, *A systematic survey of prompt engineering on vision-language foundation models*, 2023. arXiv: 2307.12980 [cs.CV].

[58] B. Lester, R. Al-Rfou, and N. Constant, *The power of scale for parameter-efficient prompt tuning*, 2021. arXiv: 2104.08691 [cs.CL].

[59] A. Efrat and O. Levy, *The turking test: Can language models understand instructions?*, 2020. arXiv: 2010.11982 [cs.CL].

[60] Q. Dong, L. Li, D. Dai, *et al.*, *A survey on in-context learning*, 2023. arXiv: 2301.00234 [cs.CL].

[61] J. Wei, X. Wang, D. Schuurmans, *et al.*, “Chain-of-thought prompting elicits reasoning in large language models”, in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 24 824–24 837. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

[62] O. Rubin, J. Herzig, and J. Berant, *Learning to retrieve prompts for in-context learning*, 2022. arXiv: 2112.08633 [cs.CL].

[63] J. White, Q. Fu, S. Hays, *et al.*, *A prompt pattern catalog to enhance prompt engineering with chatgpt*, 2023. arXiv: 2302.11382 [cs.SE].

[64] W. Wang, V. W. Zheng, H. Yu, and C. Miao, “A survey of zero-shot learning: Settings, methods, and applications”, *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019, ISSN: 2157-6904. DOI: 10.1145/3293318. [Online]. Available: https://doi.org/10.1145/3293318.

[65] X. Sun, J. Gu, and H. Sun, “Research progress of zero-shot learning”, *Applied Intelligence*, vol. 51, no. 6, pp. 3600–3614, Jun. 2021, ISSN: 1573-7497. DOI: 10.1007/s10489-020-02075-7. [Online]. Available: https://doi.org/10.1007/s10489-020-02075-7.

[66] C. Liu, C. Xu, Y. Wang, L. Zhang, and Y. Fu, “An embarrassingly simple baseline to one-shot learning”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2020.

[67] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a few examples: A survey on few-shot learning”, *ACM Comput. Surv.*, vol. 53, no. 3, Jun. 2020, ISSN: 0360-0300. DOI: 10.1145/3386252. [Online]. Available: https://doi.org/10.1145/3386252.

[68] B. Nuseibeh and S. Easterbrook, "Requirements engineering: A roadmap", in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00, Limerick, Ireland: Association for Computing Machinery, 2000, pp. 35–46, ISBN: 1581132530. DOI: 10.1145/336512.336523. [Online]. Available: https://doi.org/10.1145/336512.336523.

[69] M. Broy, "Domain modeling and domain engineering: Key tasks in requirements engineering", in *Perspectives on the Future of Software Engineering: Essays in Honor of Dieter Rombach*, J. Münch and K. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 15–30, ISBN: 978-3-642-37395-4. DOI: 10.1007/978-3-642-37395-4_2. [Online]. Available: https://doi.org/10.1007/978-3-642-37395-4_2.

[70] H. B. Reubenstein and R. C. Waters, "The requirements apprentice: Automated assistance for requirements acquisition", *IEEE Transactions on Software Engineering*, vol. 17, no. 3, p. 226, 1991.

[71] T. Cziharz, P. Hruschka, S. Queins, and T. Weyer, "Handbuch der anforderungsmodellierung nach ireb standard", *Karlsruhe: IREB InternationalRequirementsEngineeringBoard eV*, 2014.

[72] A. González, S. España, M. Ruiz, and Ó. Pastor, "Systematic derivation of class diagrams from communication-oriented business process models", in *Enterprise, Business-Process and Information Systems Modeling*, T. Halpin, S. Nurcan, J. Krogstie, *et al.*, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 246–260, ISBN: 978-3-642-21759-3.

[73] O. Lindland, G. Sindre, and A. Solvberg, "Understanding quality in conceptual modeling", *IEEE Software*, vol. 11, no. 2, pp. 42–49, 1994. DOI: 10.1109/52.268955.

[74] J. Rumbaugh and M. Blaha, *Object-Oriented Modeling and Design with UML*, 1st. Upper Saddle River, NJ: Prentice Hall, 1999, ISBN: 0-13-015920-4.

[75] M. Kossmann, M. Odeh, A. Gillies, and C. Ingamells, "11.1. 3 'tour d'horizon'in requirements engineering-areas left for exploration", in *INCOSE International Symposium*, Wiley Online Library, vol. 17, 2007, pp. 1737–1757.

[76] M. Savary-Leblanc, X. Le-Pallec, and S. Gérard, "Understanding the need for assistance in software modeling: Interviews with experts", *Software and Systems Modeling*, May 2023, ISSN: 1619-1374. DOI: 10.1007/s10270-023-01104-6. [Online]. Available: https://doi.org/10.1007/s10270-023-01104-6.

[77] M. Fowler, *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2018.

[78] S. M. White, "Modeling a system of systems to analyze requirements", in *2009 3rd Annual IEEE Systems Conference*, 2009, pp. 83–89. DOI: 10.1109/SYSTEMS.2009.4815777.

[79] D. Silingas and R. Butleris, "Uml-intensive framework for modeling software requirements", in *Proceedings of the 14th International Conference on Information and Software Technologies*, 2008, pp. 334–342.

[80] R. Sharma, P. K. Srivastava, and K. K. Biswas, “From natural language requirements to uml class diagrams”, in *2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, 2015, pp. 1–8. DOI: 10.1109/AIRE.2015.7337625.

[81] A. Abdalazeim and F. Meziane, “A review of the generation of requirements specification in natural language using objects uml models and domain ontology”, *Procedia Computer Science*, vol. 189, pp. 328–334, 2021, AI in Computational Linguistics, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2021.05.102. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921012266.

[82] V. B. R. Vidya Sagar and S. Abirami, “Conceptual modeling of natural language functional requirements”, *Journal of Systems and Software*, vol. 88, pp. 25–41, 2014, ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2013.08.036. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121213002379.

[83] O. S. Dawood *et al.*, “From requirements engineering to uml using natural language processing–survey study”, *European Journal of Industrial Engineering*, vol. 2, no. 1, pp–44, 2017.

[84] E. A. Abdelnabi, A. M. Maatuk, and M. Hagal, “Generating uml class diagram from natural language requirements: A survey of approaches and techniques”, in *2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA*, 2021, pp. 288–293. DOI: 10.1109/MI-STA52233.2021.9464433.

[85] P. Bera and J. Evermann, “Guidelines for using uml association classes and their effect on domain understanding in requirements engineering”, *Requirements Engineering*, vol. 19, no. 1, pp. 63–80, Mar. 2014, ISSN: 1432-010X. DOI: 10.1007/s00766-012-0159-y. [Online]. Available: https://doi.org/10.1007/s00766-012-0159-y.

[86] J. Evermann and Y. Wand, “Ontology based object-oriented domain modelling: Fundamental concepts”, *Requirements Engineering*, vol. 10, no. 2, pp. 146–160, May 2005, ISSN: 1432-010X. DOI: 10.1007/s00766-004-0208-2. [Online]. Available: https://doi.org/10.1007/s00766-004-0208-2.

[87] H. M. Arne J. Berre Shihong Huang and H. Alibakhsh, “Teaching modelling for requirements engineering and model-driven software development courses”, *Computer Science Education*, vol. 28, no. 1, pp. 42–64, 2018. DOI: 10.1080/08993408.2018.1479090. eprint: https://doi.org/10.1080/08993408.2018.1479090. [Online]. Available: https://doi.org/10.1080/08993408.2018.1479090.

[88] R. Braun and H. Schlieter, “Requirements-based development of bpmn extensions: The case of clinical pathways”, in *2014 IEEE 1st International Workshop on the Interrelations between Requirements Engineering and Business Process Management (REBPM)*, 2014, pp. 39–44. DOI: 10.1109/REBPM.2014.6890734.

[89] L. J. R. Stroppi, O. Chiotti, and P. D. Villarreal, "Extending bpmn 2.0: Method and tool support", in *Business Process Model and Notation*, R. Dijkman, J. Hofstetter, and J. Koehler, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 59–73, ISBN: 978-3-642-25160-3.

[90] M. W. A. Steen, M. E. Iacob, M. M. Lankhorst, *et al.*, "Service modelling", in *Agile Service Development: Combining Adaptive Methods and Flexible Solutions*, M. Lankhorst, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 59–94, ISBN: 978-3-642-28188-4. DOI: `10.1007/978-3-642-28188-4_4`. [Online]. Available: `https://doi.org/10.1007/978-3-642-28188-4_4`.

[91] B. Intrigila, G. Della Penna, and A. D'Ambrogio, "A lightweight bpmn extension for business process-oriented requirements engineering", *Computers*, vol. 10, no. 12, p. 171, 2021.

[92] G. Pervan and H. Maimbo, "Designing a case study protocol for application in is research", in *Proceedings of the Ninth Pacific Asia Conference on Information Systems*, ser. Proceedings of the Ninth Pacific Asia Conference on Information Systems, PACIS, 2005, pp. 1281–1292. [Online]. Available: `http://hdl.handle.net/20.500.11937/45440`.

Appendix A

Ethics and privacy quick scan

Response Summary:

Section 1. Research projects involving human participants

P1. Does your project involve human participants? This includes for example use of observation, (online) surveys, interviews, tests, focus groups, and workshops where human participants provide information or data to inform the research. If you are only using existing data sets or publicly available data (e.g. from Twitter, Reddit) without directly recruiting participants, please answer no.

- Yes

Recruitment

P2. Does your project involve participants younger than 18 years of age?

- No

P3. Does your project involve participants with learning or communication difficulties of a severity that may impact their ability to provide informed consent?

- No

P4. Is your project likely to involve participants engaging in illegal activities?

- No

P5. Does your project involve patients?

- No

P6. Does your project involve participants belonging to a vulnerable group, other than those listed above?

- No

P8. Does your project involve participants with whom you have, or are likely to have, a working or professional relationship: for instance, staff or students of the university, professional colleagues, or clients?

- Yes

P9. Is it made clear to potential participants that not participating will in no way impact them (e.g. it will not directly impact their grade in a class)?

- Yes

Informed consent

PC1. Do you have set procedures that you will use for obtaining informed consent from all participants, including (where appropriate) parental consent for children or consent from legally authorized representatives? (See suggestions for information sheets and consent forms on the website.)

- Yes

PC2. Will you tell participants that their participation is voluntary?

- Yes

PC3. Will you obtain explicit consent for participation?

- Yes

PC4. Will you obtain explicit consent for any sensor readings, eye tracking, photos, audio, and/or video recordings?

- Yes

PC5. Will you tell participants that they may withdraw from the research at any time and for any reason?

- Yes

PC6. Will you give potential participants time to consider participation?

- Yes

PC7. Will you provide participants with an opportunity to ask questions about the research before consenting to take part (e.g. by providing your contact details)?

- Yes

PC8. Does your project involve concealment or deliberate misleading of participants?

- No

Section 2. Data protection, handling, and storage

The General Data Protection Regulation imposes several obligations for the use of **personal data** (defined as any information relating to an identified or identifiable living person) or including the use of personal data in research.

D1. Are you gathering or using personal data (defined as any information relating to an identified or identifiable living person)?

- Yes

High-risk data

DR1. Will you process personal data that would jeopardize the physical health or safety of individuals in the event of a personal data breach?

- No

DR2. Will you combine, compare, or match personal data obtained from multiple sources, in a way that exceeds the reasonable expectations of the people whose data it is?

- No

DR3. Will you use any personal data of children or vulnerable individuals for marketing, profiling, automated decision-making, or to offer online services to them?

- No

DR4. Will you profile individuals on a large scale?

- No

DR5. Will you systematically monitor individuals in a publicly accessible area on a large scale (or use the data of such monitoring)?

- No

DR6. Will you use special category personal data, criminal offense personal data, or other sensitive personal data on a large scale?

- No

DR7. Will you determine an individual’s access to a product, service, opportunity, or benefit based on an automated decision or special category personal data?

- No

DR8. Will you systematically and extensively monitor or profile individuals, with significant effects on them?
- No

DR9. Will you use innovative technology to process sensitive personal data?
- No

Data minimization

DM1. Will you collect only personal data that is strictly necessary for the research?
- Yes

DM4. Will you anonymize the data wherever possible?
- Yes

DM5. Will you pseudonymize the data if you are not able to anonymize it, replacing personal details with an identifier, and keeping the key separate from the data set?
- Yes

Using collaborators or contractors that process personal data securely

DC1. Will any organization external to Utrecht University be involved in processing personal data (e.g. for transcription, data analysis, data storage)?
- No

International personal data transfers

DI1. Will any personal data be transferred to another country (including to research collaborators in a joint project)?
- No

Fair use of personal data to recruit participants

DF1. Is personal data used to recruit participants?
- No

Participants' data rights and privacy information

DP1. Will participants be provided with privacy information? (Recommended is to use as part of the information sheet: For details of our legal basis for using personal data and the rights you have over your data please see the University's privacy information at www.uu.nl/en/organisation/privacy.)
- Yes

DP2. Will participants be aware of what their data is used for?
- Yes

DP3. Can participants request that their personal data be deleted?
- Yes

DP4. Can participants request that their personal data be rectified (in case it is incorrect)?
- Yes

DP5. Can participants request access to their personal data?
- Yes

DP6. Can participants request that personal data processing is restricted?
- Yes

DP7. Will participants be subjected to automated decision-making based on their personal data with an impact on them beyond the research study to which they consented?
- No

DP8. Will participants be aware of how long their data is being kept for, who it is being shared with, and any safeguards that apply in case of international sharing?
- Yes

DP9. If data is provided by a third party, are people whose data is in the data set provided with (1) the privacy information and (2) what categories of data you will use?
- Yes

Using data that you have not gathered directly from participants

DE1. Will you use any personal data that you have not gathered directly from participants (such as data from an existing data set, data gathered for you by a third party, data scraped from the internet)?
- Yes

DE2. Will you use an existing dataset in your research?
- Yes

DE3. Do you have permission to do so from the owners of the data set?
- Yes

DE4. Have the people whose data is in the data set consented to their data being used by other researchers and/or for purposes other than that for which that data set was gathered?
- Yes

DE5. Are there any contractual conditions attached to working with or storing the data from DE2?
- No

DE6. Does your project require access to personal data about participants from other parties (e.g., teachers, employers), databanks, or files?
- No

DE9. Does the project involve collecting personal data from websites or social media (e.g., Facebook, Twitter, Reddit)?
- No

Secure data storage

DS1. Will any data be stored (temporarily or permanently) anywhere other than on password-protected University authorized computers or servers?
- No

DS4. Excluding (1) any international data transfers mentioned above and (2) any sharing of data with collaborators and contractors, will any personal data be stored, collected, or accessed from outside the EU?

- No

Section 3. Research that may cause harm

Research may cause harm to participants, researchers, the university, or society. This includes when technology has dual-use, and you investigate an innocent use, but your results could be used by others in a harmful way. If you are unsure regarding possible harm to the university or society, please discuss your concerns with the Research Support Office.

H1. Does your project give rise to a realistic risk to the national security of any country?

- No

H2. Does your project give rise to a realistic risk of aiding human rights abuses in any country?

- No

H3. Does your project (and its data) give rise to a realistic risk of damaging the University's reputation? (E.g., bad press coverage, public protest.)

- No

H4. Does your project (and in particular its data) give rise to an increased risk of attack (cyber- or otherwise) against the University? (E.g., from pressure groups.)

- No

H5. Is the data likely to contain material that is indecent, offensive, defamatory, threatening, discriminatory, or extremist?

- No

H6. Does your project give rise to a realistic risk of harm to the researchers?

- No

H7. Is there a realistic risk of any participant experiencing physical or psychological harm or discomfort?

- No

H8. Is there a realistic risk of any participant experiencing a detriment to their interests as a result of participation?

- No

H9. Is there a realistic risk of other types of negative externalities?

- No

Section 4. Conflicts of interest

C1. Is there any potential conflict of interest (e.g. between research funder and researchers or participants and researchers) that may potentially affect the research outcome or the dissemination of research findings?

- No

C2. Is there a direct hierarchical relationship between researchers and participants?

- No

Section 5. Your information.

This last section collects data about you and your project so that we can register that you completed the Ethics and Privacy Quick Scan, sent you (and your supervisor/course coordinator) a summary of what you filled out, and follow up where a fuller ethics review and/or privacy assessment is needed. For details of our legal basis for using personal data and the rights you have over your data please see the University's privacy information. Please see the guidance on the ICS Ethics and Privacy website on what happens on submission.

Z0. Which is your main department?
- Information and Computing Science

Z1. Your full name:
Sander van Nifterik

Z2. Your email address:
s.w.b.vannifterik@students.uu.nl

Z3. In what context will you conduct this research?
- As a student for my master thesis, supervised by::
  Fabiano Dalpiaz

Z5. Master programme for which you are doing the thesis
- Business Informatics

Z6. Email of the course coordinator or supervisor (so that we can inform them that you filled this out and provide them with a summary):
f.dalpiaz@uu.nl

Z7. Email of the moderator (as provided by the coordinator of your thesis project):
g.wagenaar@uu.nl

Z8. Title of the research project/study for which you filled out this Quick Scan:
Exploring the Potential of Large Language Models in Supporting Domain Model Derivation from Requirements Elicitation Conversations

Z9. Summary of what you intend to investigate and how you will investigate this (200 words max):
I want to investigate the potential that Large Language Models may have in supporting the derivation of domain models from requirement elicitation conversation transcripts. To do so I want to ask students to participate in an experiment where they are asked to perform a modelling assignment, based on a transcript of a conversation where a database structure is being discussed. By comparing the models that they create with an anwer model, and by interviewing them to ask about their experience, I hope to get and understanding to what degree large language models can be useful. The conversations will be fictional, and the names mentioned in the audio recordings can be pseudonymized. The assignment itself will produce a (part of) domain model, which is based on those fictitious conversations. Lastly, he participants will be interviewed to ask them about the modelling experience.

Z10. In case you encountered warnings in the survey, does supervisor already have ethical approval for a research line that fully covers your project?
- Not applicable

Scoring

- Privacy: 0
- Ethics: 0

Appendix B

Online Appendix

B.1 Elaboration

Due to the large amount of external materials related to this project, the choice has been made to use an online appendix. In the online appendix all related materials can be found and accessed. All materials are stored in the folder "Online Appendix Thesis 6847064".

In the following section, all the components of the online appendix will be elaborated upon.

B.2 Experiment Cases

In the folder experiment cases, all the materials used for the primary experiment of this thesis can be found. Inside, folders for each of the 10 cases can be found. The content of each of the case folder can be found in Table B.1.

Table B.1: Experiment cases elaboration

Filename	Description
Case x Files x	In this file the manual modelling notes for each of the three documents are captured, as well as the responses given by ChatGPT. Furthermore, the final model generated by ChatGPT, as well as all the intermediate ones, can be found.
case x sidebyside	This is a PNG of the manually created model side by side with the ChatGPT generated model. It has been marked and used for the primary experiment.
model x png	This is a PNG of the manually created model for ease of use, while executing the experiment.
model x	This is the Draw.io file used to manually model the domain model for the case.
REGxx Doc	This is the vision document for the case.
REGxx-1	This file contains the transcript of the first interview of the case.
REGxx-2	This file contains the transcript for the second interview of the case.

B.3 Initial experimentation

In this folder, the files corresponding to the initial experimentation phase can be found. It has an Excel sheet with an early attempt at capturing quantitative data, and a Word file containing the initial prompts and outcomes.

B.4 Normalization

The normalization folder contains the files corresponding to the normalization experimentation phase. It features Word files, containing both manual modelling notes as well as ChatGPT responses and models, corresponding to files 09 and 10 of the dataset. Furthermore, it features the corresponding Draw.io files belonging to each of the two file sets.

B.5 Normalization testing

The files in this folder relate to the testing done with the normalization prompts. Earlier on in the initial experimentation phase, models had been generated for file sets 01, 03, and 04. These files contain answers and models which have been made using the improved prompts and the added normalization prompts. They were used for comparing the improvement between the early prompts, and the new and improved prompts.

B.6 Prompts

In the prompts folder, Word files can be found containing all the prompts that were used during the primary experiment. Prompt vision refers to the prompt to use for the vision document, whilst the prompt 1 and prompt 2 are used in conjunction with the first and second transcript. Finally, normalization prompt 1 and 2 are used, in that order and by themselves, to apply some normalization to the models.

B.7 Quantification

In the folder quantification, all the materials generated during the process of creating the quantification method can be found. A case 10 side by side was used as a model comparison to mark and score. Two versions of the method, as well as two versions of the Excel scoring sheet can be found. Furthermore, a Venn diagram can be found that formed the foundation of the idea for the scenarios that were used in the primary experiment.

B.8 Quantified Results

In the quantified results folder, the Excel sheet that holds the quantified data gathered during the primary experiment, can be found. This Excel sheet contains the tables in which all the modelling elements, entities, attributes, relationships, and cardinalities, have been matched to scenarios. Furthermore, it contains a page where all the numbers have been gathered together. For example how often a certain scenario has occurred.

B.9 Traceability

In the folder traceability, two iterations of the traceability exercise can be found. This exercise was done in order keep try and get a better understanding of which modelling decisions ChatGPT made based on given documents versus its own knowledge base.

Appendix C

Scenarios

C.1 Elaboration

This is an overview of the scenarios that have been established to use in the quantification method. This is a complete overview of all the options that could be considered. One can choose to add, or remove from this list. Furthermore, it will offer an overview of the shorthand notations that have been used in the scenarios, as well as in the excel sheet used to keep track of the scoring.

C.2 Shorthands

H = Human

LLM = ChatGPT (or any other LLM that someone might use)

DiK = Direct Knowledge

DoK= Domain Knowledge

MiK = Missing Knowledge/forgotten

EK = Extra Knowledge

E = Entity

A = Attribute

R = Relationship

C = Cardinality

C.3 Scenarios

The scenarios have been split into two sets. The first set only applies to the entities, while the second set applies to the attributes, relationships, and cardinalities. This is because if either one of the two parties misses an entity, that should have clearly be modelled, the other party should be penalized for it. While that does not go for the other modelling elements. If either one of the parties misses an attribute, but does not even have the accompanying entity, why should that party be penalized for not having that attribute? The same goes for relationships and cardinalities.

Entities

1. (Both): H models E(x) DiK and LLM models E(x) DiK

2. (Both): H models E(x) DoK and LLM models E(x) DoK

3. (Human): H models E(x)DiK and LLM does not model E(x) MiK

4. (Human): H models E(x)DoK and LLM does not model E(x) EK

5. (Human): H models E(x) MiK
6. (LLM): LLM models E(x)DiK and H does not model E(x) MiK
7. (LLM): LLM models E(x)DoK and H does not model E(x) EK
8. (LLM): LLM models E(x) MiK
9. (Neither): H does not model E(x) MiK and LLM does not model E(x) MiK

Attributes, Relationships, and Cardinalities

1. (Both): H models A(x) DiK and LLM models A(x) DiK
2. (Both): H models A(x) DoK and LLM models A(x) DoK
3. (Human): H models A(x)DiK and LLM does not model A(x) MiK
4. (Human): H models A(x)DiK and LLM does not model A(x) EK
5. (Human): H models A(x)DoK and LLM does not model A(x) EK
6. (Human): H models A(x) MiK
7. (LLM): LLM models A(x)DiK and H does not model A(x) MiK
8. (LLM): LLM models A(x)DiK and H does not model A(x) EK
9. (LLM): LLM models A(x)DoK and H does not model A(x) EK
10. (LLM): LLM models A(x) MiK
11. (Neither): H does not model A(x) MiK and LLM does not model A(x) MiK

In the scenarios above the A for attributes can be swapped with R and C, for all scenarios.

Appendix D

Prompt guideline information

These are the full guidelines that are incorporated in to the prompt applied to the vision document, the first transcript, and the second transcript.

Now discard unnecessary and incorrect classes according to the following criteria. Adjust you modelling choices accordingly.

Redundant classes. If two classes express the same concept, you should keep the most descriptive name. For example, although Customer might describe a person taking an airline flight, Passenger is more descriptive. On the other hand, if the problem concerns contracts for a charter airline, Customer is also an appropriate word, since a contract might involve several passengers. ATM example. Customer and User are redundant; we retain Customer because it is more descriptive. Irrelevant classes. If a class has little or nothing to do with the problem, eliminate it. This involves judgment, because in another context the class could be important. For example, in a theater ticket reservation system, the occupations of the ticket holders are irrelevant, but the occupations of the theater personnel may be important. ATM example. Apportioning Cost is outside the scope of the ATM software. Vague classes. A class should be specific. Some tentative classes may have ill-defined boundaries or be too broad in scope. ATM example. RecordkeepingProvision is vague and is handled by Transaction. In other applications, this might be included in other classes, such as StockSales, TelephoneCalls, or MachineFailures. Attributes. Names that primarily describe individual objects should be restated as attributes. For example, name, birthdate, and weight are usually attributes. If the independent existence of a property is important, then make it a class and not an attribute. For example, an employee's office would be a class in an application to reassign offices after a reorganization. ATM example. AccountData is underspecified but in any case probably describes an account. An ATM dispenses cash and receipts, but beyond that cash and receipts are peripheral to the problem, so they should be treated as attributes. Operations. If a name describes an operation that is applied to objects and not manipulated in its own right, then it is not a class. For example, a telephone call is a sequence of actions involving a caller and the telephone network. If we are simply building telephones, then Call is part of the state model and not a class. An operation that has features of its own should be modeled as a class, however. For example, in a billing system for telephone calls a Call would be an important class with attributes such as date, time, origin, and destination. Roles. The name of a class should reflect its intrinsic nature and not a role that it plays in an association. For example, Owner would be a poor name for a class in a car manufacturer's database. What if a list of drivers is added later? What about persons who lease cars? The proper class is Person (or possibly Customer), which assumes various different roles, such as owner, driver, and lessee. One physical entity sometimes corresponds to several classes. For example, Person and Employee may be distinct classes in some circumstances and redundant in others. From the viewpoint of a company database of employees, the two may be identical. In a government tax database, a person may hold more than one job, so it is important to distinguish Person from Employee; each person can correspond to zero or more instances of employee information. Implementation constructs. Eliminate constructs from the analysis model that are extraneous to the real world. You may need them later during design, but not now. For example, CPU, subroutine, process, algorithm, and interrupt are implementation constructs for most applications, although they are legitimate classes for an operating system. Data structures, such as linked lists, trees, arrays, and tables, are almost always implementation constructs. ATM example. Some tentative classes are really implementation constructs. TransactionLog is simply the set of transactions; its exact representation is a design issue. Communication links can be shown as associations; CommunicationsLine is simply the physical implementation of such a link. Derived classes. As a general rule, omit classes that can be derived from other classes. If a derived class is especially important, you can include it, but do so only sparingly. Mark all derived classes with a preceding slash ('/') in the class name.

Now discard unnecessary and incorrect associations, using the following criteria. Adjust your modelling choices accordingly.

Associations between eliminated classes. If you have eliminated one of the classes in the association, you must eliminate the association or restate it in terms of other classes. ATM example. We can eliminate Banking network includes cashier stations and ATMs, ATM dispenses cash, ATM prints receipts, Banks provide software, Cost apportioned to banks, System provides recordkeeping, and System provides security. Irrelevant or implementation associations. Eliminate any associations that are outside the problem domain or deal with implementation constructs. ATM example. For example, System handles concurrent access is an implementation concept. Real-world objects are inherently concurrent; it is the implementation of the access algorithm that must be concurrent. Actions. An association should describe a structural property of the application domain, not a transient event. Sometimes, a requirement expressed as an action implies an underlying structural relationship and you should rephrase it accordingly. ATM example. ATM accepts cash card describes part of the interaction cycle between an ATM and a customer, not a permanent relationship between ATMs and cash cards. We can also eliminate ATM interacts with user. Central computer clears transaction with bank describes an action that implies the structural relationship Central computer communicates with bank. Ternary associations. You can decompose most associations among three or more classes into binary associations or phrase them as qualified associations. If a term in a ternary association is purely descriptive and has no identity of its own, then the term is an attribute on a binary association. Association Company pays salary to person can be rephrased as binary association Company employs person with a salary value for each Company-Person link. Occasionally, an application will require a general ternary association. Professor teaches course in room cannot be decomposed without losing information. We have not encountered associations with four or more classes in our work. ATM example. Bank computer processes transaction against account can be broken into Bank computer processes transaction and Transaction concerns account. Cashier enters transaction for account can be broken similarly. ATMs communicate with central computer about transaction is really the binary associations ATMs communicate with central computer and Transaction entered on ATM. Derived associations. Omit associations that can be defined in terms of other associations, because they are redundant. For example, GrandparentOf can be defined in terms of a pair of ParentOf associations. Also omit associations defined by conditions on attributes. For example, youngerThan expresses a condition on the birth dates of two persons, not additional information. As much as possible, classes, attributes, and associations in the class model should represent independent information. Multiple paths between classes sometimes indicate derived associations that are compositions of primitive associations. Consortium shares ATMs is a composition of the associations Consortium owns central computer and Central computer communicates with ATMs. Be careful, because not all associations that form multiple paths between classes indicate redundancy. Sometimes the existence of an association can be derived from two or more primitive associations and the multiplicity can not. Keep the extra association if the additional multiplicity constraint is important. For example, in Figure 12.8 a company employs many persons and owns many computers. Each employee is assigned zero or more computers for the employee's personal use; some computers are for public use and are not assigned to anyone. The multiplicity of the AssignedTo association cannot be deduced from the Employs and Owns associations. Although derived associations do not add information, they are useful in the real world and in design. For example, kinship relationships such as Uncle, MotherInLaw, and Cousin have names because they describe common relationships considered important within our society. If they are especially important, you may show derived associations in class diagrams, but put a slash in front of their names to indicate their dependent status and to distinguish them from fundamental associations. ATM example. Bank computer maintains accounts is a statement of action; rephrase as Bank holds account. Association end names. Add association end names where appropriate. For example, in the WorksFor association a Company is an employer with respect to a Person and a Person is an employee with respect to a Company. If there is only one association between a pair of classes and the meaning of the association is clear, you may omit association end names. For example, the meaning of ATMs communicate with central computer is clear from the class names. An association between two instances of the same class requires association end names to distinguish the instances. For example, the association Person manages person would have the end names boss and worker. Qualified associations. Usually a name identifies an object within some context; most names are not globally unique. The context combines with the name to uniquely identify the object. For example, the name of a company must be unique within the chartering state but may be duplicated in other states (there once was a Standard Oil Company in Ohio, Indiana, California, and New Jersey). The name of a company qualifies the association State charters company; State and company name uniquely identify Company. A qualifier distinguishes objects on the "many" side of an association. ATM example. The qualifier bankCode distinguishes the different banks in a consortium. Each cash card needs a bank code so that transactions can be directed to the appropriate bank. Multiplicity. Specify multiplicity, but don't put too much effort into getting it right, as multiplicity often changes during analysis. Challenge multiplicity values of "one." For example, the association one Manager manages many employees precludes matrix management or an employee with divided responsibilities. For multiplicity values of "many" consider whether a qualifier is needed; also ask if the objects need to be ordered in some way. Missing associations. Add any missing associations that are discovered. ATM example. We overlooked Transaction entered on cashier station, Customers have accounts, and Transaction authorized by cash card. If cashiers are restricted to specific stations, then the association Cashier authorized on cashier station would be needed. Aggregation. Aggregation is important for certain kinds of applications, especially for those involving mechanical parts and bills of material. For other applications aggregation is relatively minor and it can be unclear whether to use aggregation or ordinary association. For these other applications, don't spend much time trying to distinguish between association and aggregation. Aggregation is just an association with extra connotations. Use whichever seems more natural at the time and move on. ATM example. We decide that a Bank is a part of a Consortium and indicate the relationship with aggregation.

Eliminate unnecessary and incorrect attributes with the following criteria. Adjust your modelling choices accordingly. Objects. If the independent existence of an element is important, rather than just its value, then it is an object. For example, boss refers to a class and salary is an attribute. The distinction often depends on the application. For example, in a mailing list city might be considered as an attribute, while in a census City would be a class with many attributes and relationships of its own. An element that has features of its own within the given application is a class. Qualifiers. If the value of an attribute depends on a particular context, then consider restating the attribute as a qualifier. For example, employeeNumber is not a unique property of a person with two jobs; it qualifies the association Company employs person. Names. Names are often better modeled as qualifiers rather than attributes. Test: Does the name select unique objects from a set? Can an object in the set have more than one name? If so, the name qualifies a qualified association. If a name appears to be unique in the world, you may have missed the class that is being qualified. For example, departmentName may be unique within a company, but eventually the program may need to deal with more than one company. It is better to use a qualified association immediately. A name is an attribute when its use does not depend on context, especially when it need not be unique within some set. Names of persons, unlike names of companies, may be duplicated and are therefore attributes. Identifiers. OO languages incorporate the notion of an object identifier for unambiguously referencing an object. Do not include an attribute whose only purpose is to identify an object, as object identifiers are implicit in class models. Only list attributes that exist in the application domain. For example, accountCode is a genuine attribute; Banks assign accountCodes and customers see them. In contrast, you should not list an internal transactionID as an attribute, although it may be convenient to generate one during implementation. Attributes on associations. If a value requires the presence of a link, then the property is an attribute of the association and not of a related class. Attributes are usually obvious on many-to-many associations; they cannot be attached to either class because of their multiplicity. For example, in an association between Person and Club the attribute membershipDate belongs to the association, because a person can belong to many clubs and a club can have many members. Attributes are more subtle on one-to-many associations because they could be attached to the "many" class without losing information. Resist the urge to attach them to classes, as they would be invalid if multiplicity changed. Attributes are also subtle on one-to-one associations. Internal values. If an attribute describes the internal state of an object that is invisible outside the object, then eliminate it from the analysis. Fine detail. Omit minor attributes that are unlikely to affect most operations. Discordant attributes. An attribute that seems completely different from and unrelated to all other attributes may indicate a class that should be split into two distinct classes. A class should be simple and coherent. Mixing together distinct classes is one of the major causes of troublesome models. Unfocused classes frequently result from premature consideration of implementation decisions during analysis. Boolean attributes. Reconsider all boolean attributes. Often you can broaden a boolean attribute and restate it as an enumeration [Coad-95].