POLITECNICO DI TORINO

COLLEGIO DI INGEGNERIA GESTIONALE E DELLA PRODUZIONE

Corso di Laurea in Engineering and Management

Tesi di Laurea

Large Language Models per la generazione di Requisiti

Relatore
prof. Riccardo Coppola

Candidato
Pablo Federico Vega Carrazan
Matricola: 307042

Anno accademico 2023-2024

Sommario

Throughout the last decades, technology has had a very big impact in humans' life, both positively and negatively, depending on the point of view you prefer to see it. Even though technology may have some negative impacts, related for example to certain types of human jobs that now a days may be substituted with technology, we can say that most of the technological advancements have showed to be of great utility for humans. Everything around us is starting to change with an impacting high dynamism, and software development is very important in order to understand this changes.

Moreover, with the recent apparition of the so called artificial intelligence, everything is becoming even more dynamic, and now, humans do not only have on their hands the opportunity to develop software, or execute any kind of technological-related task, but we also have the possibility to automate this tasks, allowing ourselves to do our work in a more efficient way. Efficiency is a word which relates the outcome of the work we can do; with the time we need to spend to do it in the right way. This is exactly the point of study of this investigation.

Large Language Models, like ChatGPT, are extremely powerful tools that are already helping millions of people to work efficiently, throughout a widely diverse world of topics. For the subject of our study, we want to show how can LLM's help human users, with different kind of knowledge, to develop software development requirements, specifically, Use Case Diagrams and Use Case Narratives. Both requirements are extremely useful for software developers and also for the different stakeholders, in order to understand the different corners of the software or application under study.

Having correct requirements helps on having also correct effort estimations related to the software to be developed, which is of high importance for the project management organization and financial organization. As mentioned also before, to be efficient, we do not only need to save time on our work, but we also need our work to be well done. To this extent, we developed a detailed methodology on how can we take profit and explode ChatGPT's capabilities

in the best possible way, in order to have correct automated requirements, developed by the AI. To understand this, we need to know that we must think about this like a machine. We are going to have an input, which will be the carefully written prompt introduced to the model, and the we will have a process and an output, related to the LLM. These points are, in a simplistic way, the focal points that we explain throughout the methodology section of the study.

Afterwards, since the objective of the work is to analyze the effectiveness of ChatGPT on aiding humans with the automation of the diagrams and narratives, all the results need to be analyzed. For this, a selection of exercises related to both diagrams and narratives are selected and are the object of analysis. The original solution of the exercise can be compared to the generated solution of the AI, and the different components of the requirements can be further analyzed in detail. All these analyses can be found in the results section.

Overall, this study is able to demonstrate that ChatGPT, if used throughout the described methodology, is able to showcase very good outcomes, resulting in a useful tool both for the extent of saving time and of having good software development requirements.

Ringraziamenti

The completion of this work, even if realized by myself, would have been impossible without the support of many people, who, in one way or another, contributed in diverse manners in having the best possible results.

First of all, my tutor and co-tutor, Riccardo Coppola and Giacomo Garaccione, which, with their deep knowledge and passion related to this topic, supported and advised me in an excellent way. A special thank to both of them, who were a main pillar.

Moreover, I am very thankful for the fact of being able to have access to education, since, not all the people in the world have the same opportunities. To this point, I want to thank all the professors that I had during my career, both from my home university in Argentina (Universidad de Belgrano), and from the Politecnico di Torino. Finally. last but not least, my family and my wife, who supported me unconditionally throughout this amazing journey.

I hope, every reader of this investigation, finds it helpful for their requirements engineering tasks, and that it can help people having a more efficient allocation both from a time perspective and from an economic perspective.

Indice

Elenco delle tabelle 8

Elenco delle figure 9

1 Introduction 11

2 Background 15

2.1 Use case modeling . . . 16

2.1.1 Use Cases in Software Development: A Comprehensive Overview . . . 16

2.1.2 Use Cases as Components of Requirements Documents 16

2.1.3 What Is a Use Case, and What Is Its Purpose? . . . 17

2.1.4 The Multifaceted World of Actors . . . 18

2.1.5 Goals, Interactions, and Use Case Diagrams . . . 19

2.1.6 The Interplay Between Functional Requirements and Use Cases . . . 20

2.1.7 Representation of Use Cases with UML Use Case Diagrams . . . 21

2.1.8 Summary Level, User-Goal Level, and Sub-Function . . 22

2.2 Use Case Narratives . . . 22

2.3 Use Case Points Modeling for Effort Estimation . . . 26

2.3.1 Application Size Determination . . . 26

2.3.2 Contextual Factors . . . 27

2.3.3 Calculation of the Points . . . 29

2.4 Large Language Models (LLMs) . . . 30

2.4.1 What's the Deal with Large Language Models (LLMs)? 30

2.4.2 How Do They Work? . . . 32

2.4.3 LLMs in Action: Applications Across Industries . . . 32

2.5 The use of LLMs for Information Systems . . . 33

2.5.1 Applications of LLMs in Information Systems 33
2.5.2 LLMs in Software Development 34
2.5.3 Challenges and Considerations 35

3 Methodology 37
3.1 Introduction to the Methodology 37
3.2 Selection of Exercises 38
3.3 ChatGPT Prompting Strategy 40
3.4 Use of PlantUML 42
3.5 Automated UCD Generation Process and Feedback Loop . . . 44
3.6 Integrations of Use Case Narratives 46
3.7 Exercise Portfolio 51
3.7.1 UCD Exercises 51
3.7.2 Use Case Narrative Exercises 92

4 Results 113
4.1 Analysis of Use Case Diagram Exercises 113
4.1.1 Actors Identification 113
4.1.2 Use Cases Identification 114
4.1.3 Associations Identification 115
4.1.4 Number of Necessary Prompts Evaluation 117
4.2 Analysis of Use Case Narrative Exercises 118
4.2.1 Estimation of Exercise Difficulty 120
4.3 Correlation Analysis 122
4.3.1 Correlation with FK Ease Score 123
4.3.2 Correlation with Exercise Difficulty 125
4.3.3 Correlations Related to Number of Prompts and ChatGPT's Accuracy 127
4.3.4 Correlations Related to Number of Prompts and Number of Necessary Actors/UC/Associations 129
4.4 Discussion and Interpretation 131
4.4.1 ChatGPT's Proficiency in Generating Use Case Diagrams 132
4.4.2 Quality Evaluation of Use Case Narratives 132
4.4.3 Implications and Recommendations 133
4.4.4 Future Directions 133
4.5 Limitations and Future Work regarding UCD Exercises 134
4.5.1 Heterogeneity of Exercise Sources and Difficulties . . . 134
4.5.2 Limited Association Coverage 135

4.5.3 Dependency on ChatGPT 135
4.5.4 FK Ease Score Precision 135
4.5.5 Subjectivity in Exercise Difficulty Assessment 135
4.5.6 Evaluation of Other Diagram Types 136
4.6 Limitations and Future Work regarding UCN Exercises 136
4.6.1 Consistency and Completeness 136
4.6.2 Level of Detail 136
4.6.3 Structural Clarity 137
4.6.4 Integration with Use Case Diagrams 137
4.6.5 Recommendations for Improvement 137

5 Conclusion and Future Work 139

Bibliografia 141

Elenco delle tabelle

2.1 Weighting factors for actor types[29] 27
2.2 Use Cases weights[10] 27
2.3 Technical Complexity Factors[10] 28
2.4 Environmental Factors[10] 28
3.1 Tools comparison 43

Elenco delle figure

2.1 Use case example.[1] 20
2.2 Use Case Narrative example, extracted from *"A PM's Guide to Use Cases Part 3: Use Case Narratives from Elizabeth Larson, 2021* 23
2.3 Effort Estimation Process, extracted from *Simplifying effort estimation based on Use Case Points* from M.Ochodek &J. Nawrocki &K. Kwarciak, Y. 2010. 31
3.1 Feedback Process 46
3.2 Use Case Narratives Exercises Methodology 50
3.3 UCD generated with the PlantUML code provided by ChatGPT. 56
3.4 Original UCD provided by the exercise. 57
3.5 UCD generated with the PlantUML code provided by ChatGPT. 63
3.6 Original UCD provided by the exercise. 64
3.7 UCD generated with the PlantUML code provided by ChatGPT. 70
3.8 Original UCD provided by the exercise. 71
3.9 UCD generated with the PlantUML code provided by ChatGPT. 77
3.10 Original UCD provided by the exercise (First Version). 78
3.11 Original UCD provided by the exercise (Second Version). . . . 78
3.12 UCD generated with the PlantUML code provided by ChatGPT. 83
3.13 Original UCD provided by the exercise. 84
3.14 Original Narrative provided by the exercise. 94
3.15 Original Narrative provided by the exercise. 97
3.16 Original Narrative provided by the exercise. 101
3.17 Original Narrative provided by the exercise. 109
4.1 Actors Identification. 114
4.2 Use Case Identification. 115
4.3 Associations Identification. 116
4.4 Prompts Evaluation. 117
4.5 22 Question Analysis. 119

4.6 Pass/Fail Test. 119
4.7 Rseults Histogram. 120
4.8 Estimation of Exercise Difficulty. 121
4.9 Estimation of Exercise Difficulty. 122
4.10 Correlation of FK Score and % of Correctly Identified Actors. 123
4.11 Correlation of FK Score and % of Correctly Use Cases. 124
4.12 Correlation of FK Score and Number of Prompts. 124
4.13 Correlation with % of Correctly Identified Actors. 125
4.14 Correlation with % of Correctly Identified Use Cases. 126
4.15 Correlation with Number of Prompts. 127
4.16 Correlation between Number of Prompts and % of Correctly Identified Actors. 128
4.17 Correlation between Number of Prompts and % of Correctly Identified Use Cases. 128
4.18 Correlation between Number of Prompts and Number of Necessary Actors. 130
4.19 Correlation between Number of Prompts and Number of Necessary Use Cases. 130
4.20 Correlation between Number of Prompts and Number of Necessary Associations. 131

Capitolo 1

Introduction

The need of improved communication between the stakeholders of a project has always been one of the most important issues of a project effectivness. This is very related to the fact that we need better productivity for human tasks, and simpler processes which to some extent, allows us to do our job in a faster way, with the same quality. In the software development field, requirements engineering and use case modelling are extremly important pillars which allow the software development process to run smoothly, but at the same time, it's also a time consuming task. Given the high dynamism in software development and technological improvements in general, we may ask ourselves, how can these tasks become more efficient, allowing humans to allocate their time better.

In order to respond to this question, this thesis depps into the possible strategies and methodlogies in which the artificial intelligence can enhance the software deveopment process, helping humans saving time and money. To this point, it suggests the use of Large Language Models (LLM) as a possible tool to automate the creation of use case narratives and comprehensive specifications. We hope to transform requirements definition and improve stakeholder communication by utilizing the enormous cognitive powers of sophisticated language models.

The pillars upon which software development projects are conceived, created, and implemented are requirements engineering and use case modeling. These procedures not only are the baseline for a software system, but they also contain the key point in order to understand the complex web of features, interactions, and objectives that must be met. They are essentially the blueprint that leads a software project from conception to completion. Effective requirements engineering ensures that project teams understand what needs to be built in a clear and unambiguous manner. It serves as the foundation for decision-making, system design, and, ultimately, software functioning. On the other hand, use case modelling gives the information n how the final users would interface with the system, which makes of it a highly important tool for checking and confirming the requirements. In this setting, both efforts become critical paths for turning business requirements into technological specifications. However, as the software development field is continuously growing and becoming more complex and diverse, the same has happened to the requirements and use case modeling issues. Ambiguities, omissions, and misconceptions in requirements can result in costly rework, delays, and even project failure in some situations. Effective communication is critical among all stakeholders, from developers and designers to customers and end users. It is in this context, that Large Language Models appear as a promising option in this scenario.

The incredible improvements in technology during recent years have been marked by the apparition of the so caled Large Language Models, which showcase an ability never seen befre by technology, wich is the natural language understanding and generation. This models, without any doubt, are a milestone for the new era of artificial intelligence innovation. The fact that this models are able to write similarly to a human, and are able to understand human written prompts, has opened a new window of opportunities, which, until now, we are continuously exporing, since they have the ability to change the way we approach software development difficulties. To this point, or investigation focuses in one important point of software development, which are the requirements, and we will explore the way in which this models, with emphasis on ChatGPT, are able to help us on the automation of the development of software requirements, like use cases diagrams and narratives. The objective that we have on mind, is the one of identifying the opportunity of reducing time in manual development, which also represent cost savings, but also to understand if it has the ability to increase the quality and clarity of requirements documentation. As a result, this thesis encloses both technology and practice, positioned to investigate how advanced language models might optimize efficiency in requirements creation and improve the efficacy of communication between project stakeholders. Throughout empirical investigations, we will assess the practical usability of this methodology, by comparing the models generated by LLM to those made by human users. During this path of exploration and innovation we target the ultimate goal of making a contribution to the rapidly developing field of software development. We invite the reader to join us as we explore the possibilities of Large

Language Models to transform software development in the future as we go deeper into the fields of requirements engineering and use case modeling. In the following sections, we will provide an in-depth exploration of the background, methodology, empirical studies, and conclusions that constitute this transformative journey.

Capitolo 2

Background

The software development world is very wide in terms of the importance of the factors that may have an impact on it, and the *use case modeling*, which is the main topic related to this investigation, helps describing how users interact with the system and connects business goals with technical specifications, is the cornerstone of requirements capture. The other main topic that deals with our investigation is related to the *narratives* , which are also very important since they give textual explanations of system behavior and present a a viewpoint in a story way, on how software satisfies user wants and business procedures. There is also anther useful topic known as *Use Case Points Modeling*, which is related to the world of narratives, which models the functional needs of a software system using quantitative measures. It offers a methodical way to assess the size and complexity of use cases. The *development of Large Language Models (LLMs)* represents a major breakthrough in AI.[30] Natural language generation and comprehension skills demonstrated by LLMs position them to transform text-based jobs, such as the formulation and sharing of software requirements. The field of large language models for information systems is growing and has the potential to automate use case narratives and requirements. Thinking on the real goal or objective of the use of AI for the automation of this requirements, we may say that is is mainly aimed at improving stakeholder communication and expedite software development processes.[30]

These components work together to create the framework for this thesis, which explores how requirements definition efficiency may be increased and communication within the software development domain can be improved through the use of LLMs. Sections that follow go into greater detail on each subject, providing a deeper comprehension of both the importance of each topic on its own and how they work together to advance software engineering methods.

2.1 Use case modeling

2.1.1 Use Cases in Software Development: A Comprehensive Overview

In the intricate realm of software development, the process of defining and managing requirements is an essential aspect. Among the many tools and methodologies available, use cases stand out as a fundamental and versatile approach. They play a pivotal role in the requirements engineering process, acting as indispensable vehicles for communicating the desired behavior of a software system to all stakeholders. In this section, we embark on an exploration of use cases, delving into their definition, purpose, and how they are represented using UML use case diagrams. According to G. Goos, J. Hartmanis and J. van Leeuwen, "in the Unified Modelling Language, use cases are provided as a way of summarising the requirements on a system. They are defined informally as specifications of sets of sequences of actions, and several relationships between use cases are informally defined. Dependable, tool-supported software development necessitates precise definitions of all these concepts" [1]

2.1.2 Use Cases as Components of Requirements Documents

Use cases serve as integral components of requirements formal documents in the software development process. These documents are comprehensive records that outline what a software system is expected to do and how it should behave. Within this context, use cases form the narrative backbone, describing the system's functionality from the perspective of its users. They are, in essence, detailed stories of how users interact with the system, which is crucial for a shared understanding among project stakeholders. Inside the overall requirements file, the use cases have a very important role, and even though there is not a standard outline for this document, it is important to

[1] G. Goos, J. Hartmanis and J. van Leeuwen 2001. Fundamental approaches to software engineering. Heinrich Haussmann. p. 12.

highlight that they will only occupy the behavioral part, or the functional part. A possible outline for the requirements document, could be:

- *Chapter 1: Purpose, Chapter 2: Use cases, Chapter 3: The terms used/glossary, Chapter 4: The technology to be used, Chapter 5: The various requirements and Chapter 6: Human backup, legal, political, organizational issues.*[1]

2.1.3 What Is a Use Case, and What Is Its Purpose?

Use cases may be seen as a descriptive account of how a software system can responds to specific interactions from its users. It may be seen as "a description of the possible sequences of interactions between the system under discussion and its external actors, related to a particular goal." [2] It delineates the various scenarios in which the system is expected to operate, providing a clear, human-readable depiction of how the software functions. As there is somehow a gap between the business objectives, which smetimes are very abstarct, and the technical specifications, use cases are somehow the tool used to bridge this gap, and help to guide the development process. They offer an unambiguous and detailed perspective on the system's behavior, making them instrumental in the software development journey. When we have a determined system, with the use case tool we can try to explain how it interacts with other actors, for example, when you go to the automatic checkout in a supermarket, as a person (actor), you will need to interact with the automatic checkout system. This process can be diagramed through use cases. Later on, we will distinguish the different type of actors and their goals. In a use case we can show all the different ways or paths in which several actors interact with the system as the main actor works toward that objective.

A use case typically encompasses the following components:

1. **Actors:** These are entities that interact with the system. Actors can represent end-users, external systems, or even automated processes. Each actor is associated with specific goals they aim to achieve through their interaction with the system. These components of the system are the one having a behavior, to explain it in other words, making an analogy with programming languages, they are the ones being able to execute an

[2] (Cockburn 2000), p. 15

"IF" statement. They might be a human being, a business, a program, between other, which we will mention later on.

2. **Goals:** Use cases are goal-driven, meaning each of them serves the purpose of achieving a specific objective. For example, in an e-commerce application, a use case might be "Place Order," with the goal of allowing a customer to make a purchase.

3. **Stakeholders:**Stakeholders in the software development process include end-users, project managers, designers, and developers. Use cases clarify how the software will fulfill the needs of these stakeholders.

4. **Actor goals in use case diagrams:**Use case diagrams are graphical representations of use cases and their interactions with actors. These diagrams visually capture how the system functions from a user's perspective. They help translate abstract ideas into tangible, visual representations.

2.1.4 The Multifaceted World of Actors

Actors are a cornerstone of use cases. They represent various entities or roles interacting with the system. Deeping a little more in what we said before, it could be said that actually, the word "Role" might be a more appropriate word for a seek of understanding what actors actually do in the system. "Role" refers to the specific function that an individual does when utilizing the system, such as "cashier," for example. Most people appear to associate the term "actor" with a specific individual, but in reality, it refers to anyone who is actively involved with a system's operation. Actors can take on different forms, including:

1. **Primary actors:**These are the principal users or entities who actively interact with the system to accomplish their goals. In other words, It is the object whose goal the use case is attempting to fulfill, and the actor who starts the use case activity with their action. Even though, this might seem awkward because it is not always the primary actor who triggers the use case, we can say that they are actually the motive of why the use case was triggered, but of course that a use case may be initiated by another actor by means of technological convenience[31]. A good example can be seen in hospitals, when determined actions are first taken by clerks and then continued by more specific roles, because it is not the goal of the secretary of a hospital to have the use case run, but

actually they are convenient from a technological point of view for higher managers of the hospital. For this cases, it's important to understand and determine for who the use case is truly beneficial to. [1]

2. **Secondary actors:** The role of this type of actors in the use case is not the main, but yes to provide support or services to the system but are not the primary focus of the use case. They often influence the system's behavior indirectly. Something important to understand here is that the word "secondary" is not strictly related to the weight of the importance of the actor for the system, and the same reasoning applies for primary actors. Sometimes, it may happen that external actors may be seen as secondary in some use cases and as primary in other ones. For example, for the payment gateway, in an e-commerce system, the primary actor is the customer, while the payment gateway (secondary actor) securely processes transactions, verifies payment details, communicates with the customer's bank, and ensures successful use case.

3. **Offstage actors:**Offstage actors are not explicitly shown in the use case diagram but are involved in the system's operation. They may represent external systems, events, or environmental factors.

4. **System actors:**These are automated processes or external systems that interact with the software.

2.1.5 Goals, Interactions, and Use Case Diagrams

When we think about Use Case Diagrams, we need to think about the relationship between actors and use cases , which is represented in these diagrams. They turn to be very helpful diagrams for the user, since they provide a visual map of how actors interact with the system and achieve their goals. A use case diagram for an e-commerce system or example could show how the "Customer" actor interacts with the "Browse Products," "Add to Cart," and "Place Order" use cases. In order to understand this better, we need to see the use case from the *actors and goals model* perspective, where a use case describes the primary actor's purpose in relation to the system's responsibilities, including alternative scenarios between the system and external players, illustrating how the goal might be achieved or failed. In this sense, we must understand that a scenario is a sequence of events aimed at achieving a specific goal, initiated by a triggering action, which continues until the goal is achieved or abandoned, and the system fulfills its obligations for

Use Case: Get paid for car accident
Design Scope: The insurance company ("MyInsCo")
Primary Actor: The claimant
Main success scenario
1. Claimant submits claim with substantiating data.
2. Insurance company verifies claimant owns a valid policy
3. Insurance company assigns agent to examine case
4. Agent verifies all details are within policy guidelines
5. Insurance company pays claimant

Extensions:
1a. Submitted data is incomplete:
 1a1. Insurance company requests missing information
 1a2. Claimant supplies missing information
2a. Claimant does not own a valid policy:
 2a1. Insurance company declines claim, notifies claimant, records all this, terminates proceedings.
3a. No agents are available at this time
 3a1. (What does the insurance company do here?)
4a. Accident violates basic policy guidelines:
 4a1. Insurance company declines claim, notifies claimant, records all this, terminates proceedings.
4b. Accident violates some minor policy guidelines:
 4b1. Insurance company begins negotiation with claimant as to degree of payment to be made.

Figura 2.1. Use case example.[1]

the interaction. Fig 2.1, extracted from *Cockburn, 2000*[1] provides a high-level overview of communication between an insurance policyholder and an insurance firm, MyInsCo, with the main actor's objective being to receive payment for harm. It presents the primary success scenario and appends brief scenario fragments explaining various scenarios.

2.1.6 The Interplay Between Functional Requirements and Use Cases

"Functional requirements are those actions that a system must be able to perform, without taking physical constraints into consideration. The functional requirements specify the input and output behavior of a system" [3].These requirements are a crucial part of the software development process. To explain it in a simpler way, it's important to know and understand what the software is expected to do, and that is why requirements show detailed descriptions of what the software shoud actually do. Use cases are directly related to

[3](Bittner and Spence 2003. pg. 8)

functional requirements, as they provide a narrative that translates into these requirements. For example, the "Login" use case may result in functional requirements such as "The system must validate the user's credentials against the database."

2.1.7 Representation of Use Cases with UML Use Case Diagrams

In order to represent the actors and use cases in ne system, we may use the Use Case Diagrams, which use stick figures for actors and ellipses for use cases, and also show the relationships between actors and use cases with arrows, with arrowheads identifying the initiator. The diagram's purpose is to summarize the system's functionality, but many people mistake it for a complete model. An overview can be found in the UCD, but the majority can be found in the textual documents[3]. Unified Modeling Language (UML) diagrams provide a standardized way to represent use cases. UML use case diagrams are a visual tool for conveying how actors and use cases interact. These diagrams consist of several elements:

1. **Actors:** Represented as stick figures, they are positioned outside the system boundary.

2. **Use cases:** Shown as ovals, they are placed inside the system boundary. Each use case is labeled with its name.

3. **Association lines:** These lines connect actors to their associated use cases, visually illustrating the interactions.

4. **Generalization and inheritance:** Sometimes, actors or use cases may share common attributes or behavior. In such cases, a generalization relationship, represented by an arrow, signifies inheritance.

5. **Include and extend relationships:** Use cases can relate to each other in different ways. "Include" relationships represent scenarios where one use case includes the behavior of another. "Extend" relationships signify optional or conditional behavior that can be added to a use case.

It is important to understand that use-case descriptions provide a comprehensive account of a scenario, describing how actors and systems collaborate to achieve the goal. So when we refer to a use case, "we mean the totality of the use case, including its iconic representation, its relationships, and, most importantly, its detailed description”[4], ensuring a comprehensive understanding of the entire use case. Later on we will explain about the so called use case narratives.

2.1.8 Summary Level, User-Goal Level, and Sub-Function

According to Cockburn [1], use cases come in different levels of detail to cater to various stages of development and the needs of different stakeholders:

1. **Summary Level UC:** These provide an overview of the system’s functionality, often focusing on major user interactions.

2. **User-Goal Level Use Cases:** This type of level goes much deeper into specific user interactions, breaking down the steps and conditions required to fulfill a user’s goal.

3. **Sub-Function Use Cases:** These are the most detailed, breaking a use case into smaller, specific sub-functions, often showing alternative and exceptional scenarios.

2.2 Use Case Narratives

As mentiones lso before, the use case diagram is very helpful for showing a visual image f the actors and their interaction with the different use cases, and in order to complement this, the use case narratives describe the phases involved in each use case. However, without clearer explanations, users may not understand the process names. This is why narratives are crucial. Developers would have to guess a lot about user interaction when designing software, for instance, if we gave a developer just the name "request product" and instructed them to design a software, they would lack of a lot of valuable information and would lose a lot of time asking questions to the client about the requirements. Narratives save time for developers and are appreciated for their ability to explain complex concepts.

The UC is identified by a number and name, and is distributed according to the organization’s requirements processes, templates, and standards. The parameters of each use case are outlined in pre- and post-conditions. The start and finish points are provided. The narrative also includes a matrix

[4](Bittner and Spence 2003). pg. 5

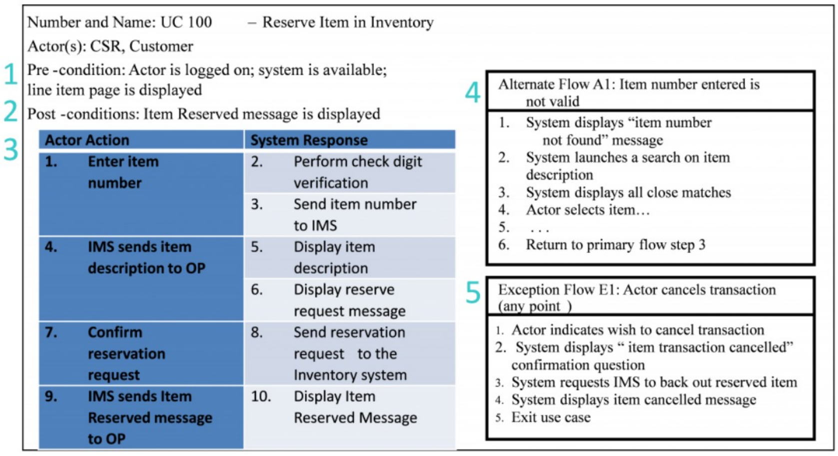


Figura 2.2. Use Case Narrative example, extracted from *"A PM's Guide to Use Cases Part 3: Use Case Narratives from Elizabeth Larson*, 2021

illustrating the actions of the actor and the system's responses to each request. We describe the primary, alternate, and exception flows [4]. Fig 2.2 shows a short example of a narrative with 5 important point which we will explain.

1. **Initiating a use case:** Before a use case can start, several factors need to be considered, such as whether the customer must be logged in, the system's availability, and the user interface in use. These conditions are addressed as pre-conditions and are vital to avoid conflicts during development.[4]

2. **Determining use case completion:** It is not very easy to realize for a user, when a use case is actually completed, and in this case, the use case narratives are also a useful tool since they provide answers through post-conditions. Post-conditions, along with pre-conditions, set the use case's boundaries. For example, reserving items does not encompass shipping them, which is the scope of a different use case.[4]

3. **The primary path:** The most common way to progress from the start to the end of a use case is the primary path, often referred to as the "happy path." It always commences with the actor's action and concludes with the system's response, detailing the process steps. These steps continue until the post-condition is achieved.[4]

4. **Alternative path:** Alternate paths offer different routes from the beginning to the end. They may or may not return to the main path. In some cases, as illustrated in the previous example, the primary path is revisited.[4]

It is important to highlight that use cases are more than just a diagram, and that they are described in the narratives, which shows the different scenarios from the start to the completion. As we mentioned before, use cases can be described on 3 different levels, and so, also we can have narratives at different levels.

Even though in Fig. 2.2, we showed an example of a very brief narrative, we did it for a seek of understanding, but use case narratives could be quite more detailed and comply with determined standards which will describe the functional requirements in a better way. In order to follow a schematic route, we will follow Cockburn's "fully dressed" use case template, which has the following characteristics:

[1]

- One column (no table)
- Sequenced, meaning that the steps form the mains success scenario and from extensions are numbered following the Dewey decimal system, and extensions should have a letter beside the number in order to show a difference with respect to the original step from the main success scenario. For example, the second step from the main scenario can be "1", and one possible extension could be "1.b".
- **Use case name:** A verb or brief phrase should describe the objective of the use case.
- **Scope:** The extent of the "system" under consideration
- **Level:** State from which of the 3 levels the narrative corresponds
- **Intention in context:** A declaration of the main actor's goal and the situation in which it is being done.
- **Primary actor**
- **Stakeholders interest:** The list of parties involved and the primary interests they have for the use case.

- **Precondition:** What we can infer about the environment and system as it stands right now.
- **Minimum guarantees:** How stakeholders' interests are safeguarded under all conditions.
- **Success guarantees:** The system's and the environment's condition if the main actor's objective is accomplished.
- **Main success scenario:** The scenario's numbered steps, from trigger to goal delivery and any cleanup in between. In the part on extensions, conditions and options are displayed.
- **Extensions:** <step_altered> <condition> ":" <action_description> or <sub-use_case>

The last point that I consider crucial in use case narratives is the concept of "transactions". Interactions or information transfers between the system under description and external entities, other systems, or external services are referred to as transactions. Typically, a use case narrative describes precise situations or examples of how a system will be applied to accomplish a given objective. According to Jacobson [6], "*a use case is a sequence of transactions performed by a system, which yields an observable result of value for a particular actor*", and he also states that a transaction is a series of operations carried out by a system, initiated by a timed trigger within the system or a stimulus from an actor to the system.

In a use case narrative, a transaction may involve more than one activity, such as the intake, processing, and output of data. They show how data moves through the system and what has to be done to complete the tasks outlined in the use case.

For instance, a "Make a Purchase" use case in an e-commerce system may include the following transaction:

User Sign-in:

[Transaction: To log in, the user gives their credentials.

System Reaction: The system authorizes access and verifies credentials.]

So, as we can see, the transaction is composed of the action of the user plus the system reaction.

In conclusion, use cases are crucial in software development because they show how a system works from the user's perspective. They help connect business goals with technical details, making sure everyone involved understands the system. The diagrams make these use cases clearer. In the next sections, we'll look at more related ideas and see how Large Language Models can help make defining requirements and communication among team members easier and better.

2.3 Use Case Points Modeling for Effort Estimation

The demand for specialized software to assist companies is expanding in tandem with technological advancements, leading to a rise in the number of software development professionals and consequently increasing competition in the field. The fact of estimating the effort during software development is highly important and valuable since it directly affects how resources are allocated and project timelines. Knowing the size of a project helps determine the amount of effort needed. The utilization of Use Case Points (UCP) modeling offers a systematic approach to estimating effort, taking into account various factors inherent to the software system that is being developed.[7][8]

2.3.1 Application Size Determination

Number of Actors

As stated in the previous section, actors are entities that engage with the system in the context of software systems. The total system size and the effort needed to construct it are strongly influenced by the complexity and quantity of actors. It is crucial to classify actors into simple, average, and complex types. For instance, "a simple actor is an actor defined in a program interface, an average actor is an actor involving a system that communicates via a protocol for example, TPC/IP, and a complex actor is defined as an actor that communicates through a Graphical user Interface (GUI) or a web page"[5], therefore, influencing the overall application size. The relation between the actor type of complexity and its respective associated value are illustrated in table 2.1.

[5]A. Effendi & R. Setiawan & Z. Erlisa Rasjid, Y. 2019. pg. 692 [9]

Actor type	Weighting factor
Simple	1
Average	2
Complex	3

Tabella 2.1. Weighting factors for actor types[29]

Number of Use Cases

Use cases define the tasks a system needs to perform. The number and complexity of use cases affect the application's size. In order to make a good estimation of the effort required, it is important to distinguish between simple and complex use cases . The complexity of a use case depends on the number of transactions it involves, which impacts the application's size[10]. Table 2.2 shows this relationship.

Use Case Type	Number of Transactions	Weighting factor
Simple	≤ 3	5
Average	4 to 7	10
Complex	≥ 8	15

Tabella 2.2. Use Cases weights[10]

2.3.2 Contextual Factors

There are several contextual factors that may have an impact on the effort calculation apart to the actors and use cases, which include for example industry standards, regulatory requirements, and project-specific constraints. These factors ensure the application meets specific requirements and compliance standards. The UCP approach considers both the technical complexity of the system (13 elements) and the development environment (8 factors). Tables 2.3 and 2.4 [10] list these adjustment factors.

Technical Complexity Factors

Technical complexity factors include data processing needs, performance limitations, and architectural complexities. When considering the system's

Factor	Description	Weight
T1	Distributed system	2
T2	Performance	1
T3	End-user efficiency	1
T4	Complex processing	1
T5	Reusable code	1
T6	Easy to install	0,5
T7	Portable	0,5
T8	Easy to change	2
T9	Concurrent	1
T11	Security features	1
T12	Access for 3rd parties	1
T13	Special training required	1

Tabella 2.3. Technical Complexity Factors[10]

Factor	Description	Weight
T1	Familiarity with the standard process	1,5
T2	Application experience	0,5
T3	Object-oriented experience	1
T4	Lead analyst capability	0,5
T5	Motivation	1
T6	Stable requirements	2
T7	Part-time workers	-1
T8	Difficult programming language	-1

Tabella 2.4. Environmental Factors[10]

total complexity, these factors have a direct impact. For example, a system with extensive data processing or strict performance requirements has higher technical complexity, affecting the estimation process. Technical complexity is rated on a scale from 0 to 5, with higher values indicating greater complexity. The value multiplied by the factor's weight and summed produces the following equation:

$$TCF = 0.6 + (0.01 \cdot \sum_{i=1}^{13} TFweight_i \cdot value_i)$$

The anticipated level of impact of each technical complexity factor on the project is represented by a value called $value_i$ This value ranges from 0 to 5. [10]

Environmental Complexity Factors

Environmental factors, such as the team's familiarity with the process and their motivation, can add complexities. These factors directly impact the effort needed to integrate the software. If a team lacks motivation or skills, more development effort is required. The equation below illustrates how the influence of environmental factors (EF) is determined in a manner similar to that of technical complexity factors. [10]

$$EF = 1.4 + (-0.03 \cdot \sum_{i=1}^{8} EFweight_i \cdot value_i)$$

2.3.3 Calculation of the Points

Unadjusted Use Case Points (UUCP)

UUCP is the basic measure for estimation, representing the total weighted actors and use cases. To compute UUCP, multiply the weights of the actors and add the weights of the use cases.[10]

$$UUCP = (TotalActorWight) + (TotalUseCaseWeight)$$

Example: If a system has three simple actors (weight 1 each), one average actor (weight 2), and four use cases with weights 3, 4, 2, and 5, the UUCP calculation would be:
$(3 \times 1) + (1 \times 2) + (3 + 4 + 2 + 5) = 20$ UUCP.

Adjusted Use Case Pints (AUCP)

After calculating UUCP, adjustments are made for technical and environmental complexities. These adjustments, usually percentages, are applied to the UUCP to determine the Adjusted Use Case Points (AUCP). The formula for determining Adjusted Use Case Points (AUCP) is as follows [10]:

$$AUCP = UUCP \cdot (TCF \cdot EF)$$

Productivity Factor and Effort Estimation

The approach involves estimating effort in terms of man-hours by multiplying the Unadjusted Use Case Points (UCP) with the productivity factor (PF). Initially, Karner suggested setting a default value of 20 hours per UCP for PF.[10]

Schneider and Winters proposed a technique for determining the starting point of PF. They recommended counting the number of environmental variables F1-F6 that are expected to have an influence of less than three, as well as factors F7-F8 that are predicted to have an influence greater than three, based on their experience. If the total count is two or less, the default value of 20 hours per UCP should be used. If the count falls between 3 and 4, a PF of 28 hours per UCP should be utilized. For counts exceeding 4, a PF of 36 hours per UCP should be applied. However, it is important to note that in cases where the project is deemed extremely risky, these values may not be suitable.[10]

Fig. 2.3 shows a diagram with the overall process with sequenced steps of how effort estimation is calculated, in the same way we have explained in this section.

In overall, the important thing for the reader to understand about use case point modelling is the fact that it is a tool that allows the possibility to estimate the effort needed for software development projects, by analyzing the important factors regarding use case modelling, like actors, use cases, and various complexities. This helps people having reliable project estimates for the software development, ensuring that projects run efficiently.

2.4 Large Language Models (LLMs)

2.4.1 What's the Deal with Large Language Models (LLMs)?

We can say that LLMs are the latest big thing when we want to talk about artificial intelligence, most of all when it comes to understanding and processing language, since they are able to process very big amounts of text and understand it, making them great for things for a wide variety of tasks and that allows them to be a tool that can be used in many industries.

Essentially, and in order to explain it in a simple way, these models are just very advanced machine learning models that get trained on very big datasets to learn how language worksn and thanks to this, they are able

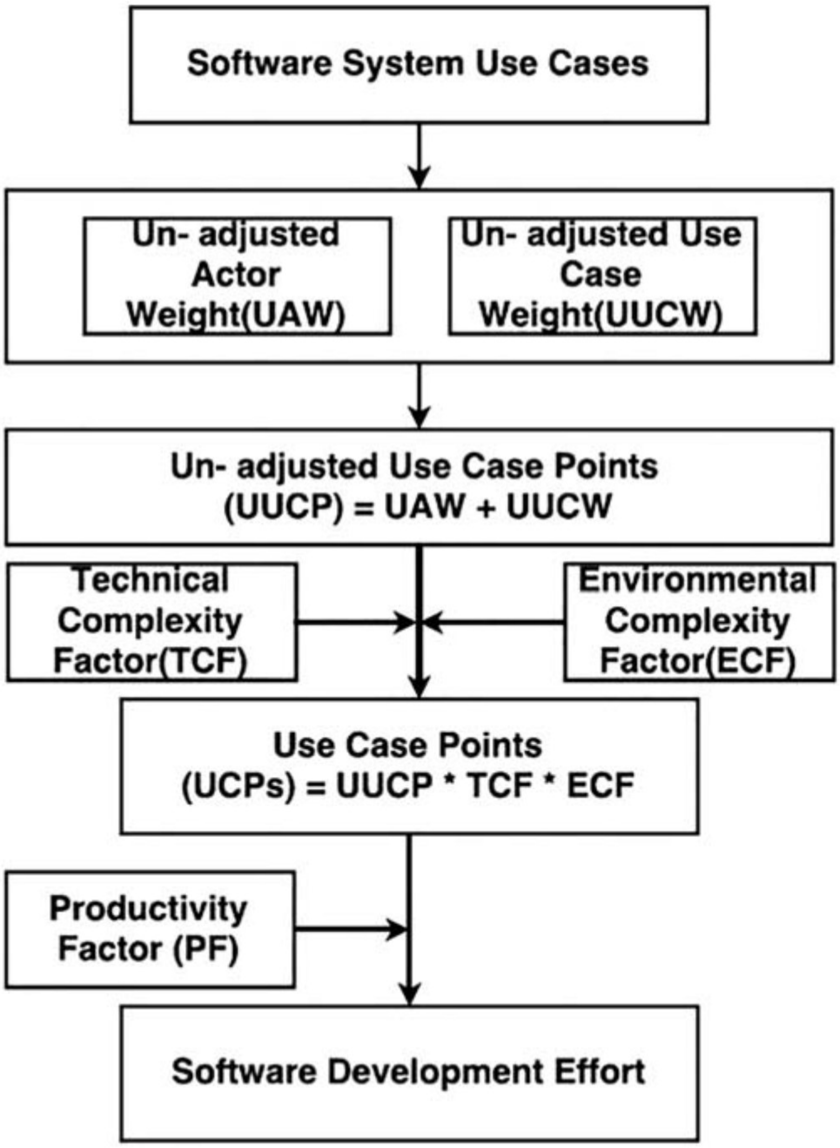


Figura 2.3. Effort Estimation Process, extracted from *Simplifying effort estimation based on Use Case Points* from M.Ochodek & J. Nawrocki & K. Kwarciak, Y. 2010.

to generate texts that are coherent and that are also correct according to the context. Now a days we also have many models like BERT, GPT-3, and Codex that are helpful as virtual assistants, content creation, sentiment analysis, between many others. (He, H., 2023).[11]

2.4.2 How Do They Work?

As mentioned before, the model first learns a lot from reading many text to understand how language works. The importance of this is that in this way, they don't need a special programming to know what you mean. After their big training, they can learn specific things like law or medicine or any other field for example. (Rana, S., 2023).[12]. [13].

2.4.3 LLMs in Action: Applications Across Industries

The versatility and capabilities of LLMs extend across various industries, enabling them to process and understand human language for a wide range of applications. The capability of LLMs at extracting precise information from extensive data, as a result of their training on diverse sources positions them to be an ideal tool in numerous fields.[13, 14]

Healthcare Industry

In this field, the model has become really a very important tool since they can look at clinical notes, lab results, and health records for example, so it means that if it is correctly used, it may be helpful for doctors to diagnose and treat patients. Specifically, the are models like Med-PaLM which are even better for this field. Of course, there are some concerns about privacy and the risk of spreading false information when using patient data. [13, 14, 16, 17]

Finance Industry

In finance, LLMs are able to analyze market trends, predict stock prices, and offer insights into investment strategies. Specifically, we have models like BloombergGPT which are better at financial tasks than the normal models, even though, here there's also a risk of generating false or harmful content, so it is important to have some regulation in its use.[13, 14]

Educational Field

We can ask ourselves on how can LLM's be helpful in schools or universities, and the truth is that when talking about education, these models can be extremely helpful if well used, and can make a big difference. They can help with reading and writing in classrooms for example, as well as answer questions from almst any kind of topic that a student or teacher may want to ask, helping with educational tasks with facility and precision. But it is a tool that need to be used with precaution, since it may also give wrong information, so it is not good to just rely on the artificial inteligence to solve educational problems, but just to be used as a tool to help when a specific question arises, and if possible, it's always better to double check the information provided by the model. So, while LLMs have lots of potential, we need to handle these issues carefully.[13, 14, 15]

Law Industry

In the legal field, LLMs have the ability to help with writing documents, predicting court decisions, and analyzing legal texts. Studies show that LLMs can understand and reason about law very well. For example, GPT-4 scored in the top 10% on a simulated bar exam. However, there are still concerns about copyright, privacy, and biases. Despite these challenges, LLMs can be very useful in law.[14]

2.5 The use of LLMs for Information Systems

2.5.1 Applications of LLMs in Information Systems

Retrieval and Processing of Data

Information Systems in general are forsure highly benefited by the advancements in artificial intelligence and specially large language models, since the models are great at understanding and processing data, which helps information systems work better. They can summarize documents and find important information, making it easier to make decisions.[18, 19]

Natural Language Interfaces

With their natural language processing capabilities, LLMs make it possible to talk to computers in plain language. This makes systems easier to use and more accessible.[18, 20]

Information Retrieval and Search

LLMs help make search engines better by understanding what people are really looking for and giving more accurate results.[18, 19]

Content Generation and Augmentation

LLMs have demonstrated remarkable versatility in generating and improving content for information systems, as noted by Huang[6]. These models are adaptable to making high-quality content across diverse platforms such as social media posts, product descriptions, and news articles. Beyond textual applications, large language models play a role in advancing the design and commercialization of multimedia communication systems, a point emphasized by Akyildiz and Guo[7][18, 19]

2.5.2 LLMs in Software Development

The use of LLM's for the software development is higly related to the goal of our investigation, since the goal of this research exactly is to see how well LLMs can help with software development by comparing their work with respect to the one of human users, and in this way be able to understand their strengths and weaknesses better. This will help us use LLMs more effectively for software projects, specially to write requirements and create use case scenarios, making the whole process more accurate and efficient. [22]

Existing studies, though preliminary, demonstrate the potential of LLMs in assisting with efficient requirements engineering. For instance, prompt engineering with BERT has been employed for automated requirement classification. Addressing requirements completeness, D.Luitel[8] utilized BERT to predict satisfying requirements for masked slots.[22]

[6]Huang et al. in 2022

[7]Akyldiz and Guo, Y. 2022.

[8]Improving requirements completeness:Automated assistance through large language models, 2023

Traceability and Beyond:

While many LLM applications focus on activities like code creation, testing, and maintenance, their innate language processing abilities offer significant potential to enhance requirements engineering tasks. Notably, traceability in software engineering, a widespread and multidisciplinary challenge, can benefit from LLMs. Given that requirements are frequently expressed in everyday language, LLMs become powerful allies in establishing traceability connections between requirements and other technical artifacts like code and tests.[22]

2.5.3 Challenges and Considerations

Data security and privacy are major problems when integrating LLMs into information systems. To preserve user trust and system integrity, organizations need to have strong privacy policies in place and protect sensitive data. It is difficult for LLMs to be interpreted, and increasing transparency is necessary to win over users and stakeholders. To guarantee accountability, it is essential to investigate interpretability frameworks and techniques. Biases, fairness, and possible effects on users or stakeholders are among the ethical factors to be taken into account. Achieving equilibrium between efficiency enhancements and moral considerations guarantees that LLMs make valuable contributions to information systems without sustaining prejudices or causing harm.[23] *Methodical Inquiry: Evaluating LLMs in Software Development*: To accomplish the research objective, a methodical approach will be employed. The empirical study design will involve comparing the quality of requirements and use case models generated by LLMs against those done by human users. In order to have more reliability, some key parameters, such as clarity, specificity, and completeness, will be used for comparison, aiming to provide a better understanding of how LLMs can optimize efficiency in requirements definition and communication within software development projects.

Capitolo 3

Methodology

3.1 Introduction to the Methodology

During this methodology section, the main objective is to explain how was the approach to evaluate the ability of ChatGPT n the creation and automation of the desired diagrams that are under study, in order to understand if it is a useful tool that could potentially help humans for automation of software requirements. As we als mentioned before, the goal of this methodology is to find the right way to use ChatGPT in order o help humans in reducing the amount of time spent on the difficult process of manually creating UCDs and narratives. As indicated in [24], surveys carried out in recent times have shown that 54.5% of those surveyed found it difficult to deal with the time-consuming process of manually creating use case diagrams using programs like Visual UML, Draw.io, Rational Rose, and Smart Draw.[24] Specifically, regarding UML diagrams, luke use case diagrams, 40.2% of system engineers chose manual creation, 23.9% used semi-automated methods, and only 13% employed automatic methods. Drawing use case diagrams manually demands system analysts to retain accurate notation, which emphasizes the need for automation. **Key Components:**

- Purpose: The main goal is to evaluate ChatGPT's effectiveness in supporting the development of UCDs and narratives, with a focus on correctness and efficiency.

- Selection of Exercises: A carefully selected collection of exercises, sourced from a variety of online resources, including books, papers, websites, and search engines, provides the basis for assessing ChatGPT's effectiveness. Specific requirements were implemented to guarantee that the exercises were suitable for evaluating the model's capabilities.

- ChatGPT Prompting Strategy: The methodology emphasizes clarity and simplicity by using a carefully defined vocabulary to prompt ChatGPT. Different prompts were designed to account for differences in the difficulty of the exercises.

- Evaluation Criteria: The research establishes particular standards by which ChatGPT's solutions will be examined. These standards include accuracy in classifying actors and use cases, as well as associations of different types, and the completeness of the Plant UML code that is produced.

- Plant UML Integration: The tool chosen to automate the production of UCDs is Plant UML. This decision's motivation is provided later on, and any changes to the Plant UML code produced by ChatGPT are also discussed in following subsections.

- Feedback Mechanism: Examples of how ChatGPT results were improved through the use of iterative feedback and corrections are discussed. The resulting Plant UML code was refined by the use of multiple prompts when necessary.

- Integration of Use Case Narratives: In order to ensure that there is no confusion with the UCD assessments, a method for integrating use case narratives into the evaluation process is outlined.

  This technique aims to assess ChatGPT's technological skills as well as providing insights into how it might speed up and enhance the software development requirements process. The next parts provide an in-depth review of the research process by going into particular aspects of the method used.

3.2 Selection of Exercises

The exercises chosen for this study were carefully chosen in order to assess ChatGPT's ability to extract relevant information from textual descriptions in order to automate the creation of use case diagrams (UCD). In order to accomplish this goal, a portfolio including thirty distinct types of exercises were chosen and carefully examined. A total of ten exercises related to use case narratives and twenty exercises dealing to use case diagrams were chosen and examined. Exercises that included both UCD and narratives from the same textual description were included when it was feasible; as a result, some narrative exercises correspond to the same UCD exercise that was examined, while other narrative exercises are independent. In order to select the best possible exercises for the investigation, a criteria was undertaken.

- Criteria: Inclusion of Textual Descriptions: The most important thing was to prioritize exercises that included textual descriptions in any kind of textual description, from stories, narratives or sequential interactions. The main objective of this was to have the requisite of extracting information from the text in order to generate the diagram. In this way, we are able to evaluate throughout many factors, ChatGPT's capacity of automating this diagrams.

- Exclusion of Solution-Only Exercises: Exercises that only provided solution use case diagrams without any textual support about actors or interactions were not included in the analysis. Assessing ChatGPT's capacity to extract data from narrative-based descriptions was the main goal. Of course that the rationalle behind this, is the emphasis on information extraction from the model, that our investigation is evaluating. By excluding exercises with predetermined solutions without textual context allowed the study to stay focused on ChatGPT's ability to extract information from various types of text.

- Consideration of Complexity: Something important to highlight is the big variety of the textual complexity of exercise. There were some exercises which presented very long and complex descriptions but the solution was actually quite simplistic, meaning that the trick of the exercise was precisely to know the exact information that must be extracted to create the diagram. But on the other hand there were also exercises were you could find simple descriptions or short ones, that actually required more sophisticated diagrams. This is something that could test ChatGPT's abiities under different kind of exercise complexities, in order to have better analysis of the results obtained. In other words, this consideration of different exercise complexities also has a rationale, which aimed to put ChatGPT under certain situations which required different types of cognitive processing capacities.

- Insight into ChatGPT's Evaluation: The wide range of exercises allows an in-depth investigation of ChatGPT's advantages and disadvantages when interpreting textual data. The results section of the thesis will include a detailed study and presentation of this analysis.

  This meticulous exercise selection procedure guarantees that the methodology is in line with the main goal of evaluating ChatGPT's ability to interpret and integrate textual data for the automated creation of UCDs. We will go into more detail about how these exercises were implemented to assess ChatGPT's performance in the next sections.

3.3 ChatGPT Prompting Strategy

A carefully designed prompting technique was used for guiding the ChatGPT interaction in order to obtain the best possible responses for the automated use case diagram (UCD) production. The main prompt that was used in every exercise had a similar structure:

"The following should be textually analyzed, and a use case diagram created containing several use cases. Identify the actors, use cases, and associations. Please give me the PlantUML code for the use case diagram corresponding to the following text: (TEXT)."[28]

This structure was partially extracted from the following website: https://www.scribd.com to-Use-Use-Cases

- Refinement of Prompts:

User Goal Level Approach:

In certain instances, the prompting strategy was refined to request a use case diagram with a specific focus, using a prompt such as: *PROMPT 1:* **"The following should be textually analyzed, and a use case diagram created containing several use cases. Identify the actors, use cases, and associations. Please, use the user goal level approach. Please give me the PlantUML code for the use case diagram corresponding to the following text: (TEXT)."**

Consideration of Relationships:

In order to address potential variations in the exercises, prompts were also adjusted to instruct ChatGPT to consider specific relations or associations: *PROMPT 2:* **"The following should be textually analyzed, and a use case diagram created containing several use cases. Identify**

the actors, use cases, and associations. Please, use the user goal level approach. Also, please consider any possible generalization relationship between use cases or between actors, and any possible ’include’ or ’extend’ relationship between use cases. Please give me the PlantUML code for the use case diagram corresponding to the following text: (TEXT)."

- Spelling and Grammar:

It happened many times when during the investigation, that without even taking it into account, I introduced certain prompts to ChatGPT that included grammar or spelling mistakes, and this are actualy some thing that according to many studies, can affect ChatGPT’s capacity performance in creating the response that we expect.It is something very common to obtain responses from the AI which are not the wanted ones, sometimes with ceratin unwanted errors, or sometimes with to any missing items, or maybe with syntax errors, and that is why having good grammar and spelling could help very much in avoiding this errors in ChatGPT’s responses.

- Insight from Relevant Literature:

This method uses info from research that shows how following language rules helps make accurate use case diagrams (UCDs). Papers like "Automated use case diagram generator using NLP and ML"[24] and "Domain-specific language for automatic generation of UML models"[25] talk about how using specific languages can improve communication and reduce mistakes in software design.

- Specialized Methods:

The method was influenced by special approaches like the Language of Use Case for Model Automation (LUCAM). Knowing about domain-specific languages for UCD automation helped shape the prompts we used.

- Balancing Detail and Simplicity:

To make ChatGPT work better, we tried to keep the prompts detailed yet simple, considering that language models might struggle with very complex prompts.

3.4 Use of PlantUML

Plant UML was picked as the tool for making use case diagrams (UCDs) for a few reasons:

Open-Source and Accessibility: Plant UML is free and open to everyone, making it easy for lots of people to use. This fits with the research goals of being inclusive and practical.[26]

UML Standard Compliance: Plant UML sticks to the Unified Modeling Language (UML) standards, ensuring the diagrams are reliable and can be easily used in software development.

Text-Based Syntax: Plant UML's text-based syntax works well with ChatGPT's text-focused nature, making it easier to turn text descriptions into diagrams.

Versatility in Diagram Types: Plant UML can make different kinds of diagrams, not just use case diagrams, which is useful for future research.

Graphical Representation Quality: Plant UML makes high-quality, clear diagrams, which are important for effective communication in software development.

Community Support and Documentation: Plant UML has lots of help available, with good documentation and community support, making it easier for researchers to use.

Integration with Existing Workflows: Plant UML integrates well with current software development tools and systems, so it's easy to add the automated diagrams into existing processes.

Consistency in Evaluation: The fact of using just one tool for the creation of the use case diagrams, somehw helped the study to be more reliable and consistent throughout the overall evaluation of all the exercises. To this end,using Plant UML for all exercises ensured that this investigation is more consistent and of course, also makes it easier to assess how well ChatGPT performs without worrying about differences in diagram tools.

Consideration of Alternatives: While taking the decision on which tool to use for the diagram creation, some investigation was also made in this field in order to fully understand the differences with some of the main tols available, and even though Plant UML was chosen, there were some ther tools that could have also been used for the investigation. Mainly, the decision was based on factors like ease of use, community support, and UML standards compliance. Plant UML stood out as the best fit for these needs. A thorough review compared Plant UML with other tools like yUML, Chart Mage, ZEN UML, Umple, UML Graph, and Dot UML, considering their features and strengths. This comparison analysis took into account two major aspects, which are presented in table 3.1.[24]

Tool Name	Use correct UML notation	Support Use Case Diagrams
PlantUML	YES	YES
yUML	NO	YES
Chart Mage	YES	NO
Zen UML	YES	NO
Umple	YES	NO
UML Graph	YES	NO
Dot UML	YES	NO

Tabella 3.1. Tools comparison

Factor 1: Use Correct UML Notation:

The first factor taken into account was the correct UML notation from the tool, since it is highly important for our study that the tool that created the diagram respected and followed the correct UML notation in order for the evaluated exercises to be reliable and compliant with the rules of the UML notation. To this end, from all of all the tools that were assessed, Plant UML was the only one that really showed a complete conformance to UML notation requirements, and this aspect is essential in order to have exercises with high quality standards that at the same time could meet the requirement that we are studying, which at the end is to see if the created diagrams could be actually hepful for human users to use for the software development processes.

Factor 2: Support Use Case Diagrams

A tool that not only followed UML standards but also explicitly supported the creation of use case diagrams was required due to the research's particular focus on use case diagrams. Plant UML turned out to be the best option, meeting this requirement properly. Its unique use case diagram creation functionality sets it apart from competitors and offers a focused and efficient solution that is adapted to the objectives of the research.

The comparative analysis's findings showed that, of the solutions identified, Plant UML is the only one that performs well in both evaluated domains, which placed Plant UML as the most suitable tool for the current study. The careful evaluation of these variables is consistent with the dedication to accuracy, uniformity, and relevance in the field of software modeling.

3.5 Automated UCD Generation Process and Feedback Loop

A methodical approach was taken in the automated use case diagram (UCD) generating process, which adapted to the several and detailed exercises that ChatGPT employed to assess. The workflow was designed with feedback loops for improvement and coherence, ensuring an evaluative and accurate process.

ChatGPT's preliminary analysis:

ChatGPT started each exercise by analyzing the text to identify actors, use cases, and possible associations. After analyzing the exercise description, ChatGPT created a PlantUML code that contained its interpretation.

Input into PlantUML Platform: The PlantUML code provided by ChatGPT was then input into the PlantUML platform for the automated generation of the corresponding UCD. This step aimed to visualize the UCD based on ChatGPT's textual analysis.

Comparative Analysis: After the creation of UCDs, a comparative analysis was made. The generated UCD was placed under evaluation by comparing it against the original solution (if available) and also considering the existing knowledge on UCDs. This comparative procedure served as an initial assessment of ChatGPT's accuracy.

Feedback Loop Initiation: Feedback was provided to ChatGPT according to the situation in order to improve the UCD's correctness. The quality of the initial UCD correlated with the feedback's extension and depth. Different situations influenced the feedback procedure:

- Scenario 1: Minor Corrections:

Feedback was brief if the generated UCD had small errors but was otherwise accurate. For example, a small correction was sufficient if a few actors were missed or misidentified. In the same way, if associations were missing, the gaps were fixed with specific feedback.

- Scenario 2: Missing Generalization or Relations:

When actors and use cases were correctly identified by the UCD but no specific associations or generalization relations were found, the feedback was intended to guide ChatGPT to explicitly address these needs. This strategy ensured a more complete and accurate UCD.

- Scenario 3: Complex Feedback for Messy Diagrams:

Occasionally, exercises with long, complex descriptions generated confused diagrams. In these situations, feedback needed more specific details to address the structural and semantic problems. Sometimes, the complex nature of these exercises required several rounds of feedback in order to well enough improve the UCD.

- Scenario 4: Self-Generated Information:

An alternate method involves manually reviewing the exercise description and explicitly giving ChatGPT the necessary actors, use cases, and relations for exercises that contained a vast quantity of irrelevant data, with respect to the needed information for the UCD creation. By taking the initiative, ChatGPT's processing was simplified, producing UCDs that were more understandable and accurate.

- Scenario 5: Syntax Errors and Tool Limitations:

When ChatGPT presented PlantUML code that contained syntax errors, the feedback identified the line of code that needed to be fixed. Sometimes, problems continued in spite of repeated feedback, indicating a constraint in the tool's capacity to reliably generate error-free code for some exercises.

Iterative Feedback and Refinement: Upon each round of feedback, ChatGPT incorporated the modifications suggested into account and produced an updated version of the PlantUML code for assessment. This method was iterative in order to gradually improve the UCD, work around constraints, and raise the overall quality of the diagrams that were produced.

Tool Limitations and Future Considerations: The cases in which feedback failed to correct syntax errors demonstrate a weakness in the tool's effectiveness. This limitation makes the tool less useful in certain circumstances and presents difficulties for users who are not familiar with UCD. Future considerations might go further into methods of minimizing these constraints, assuring a more intuitive and effective user experience.

Fig. 3.1 shows a diagram with the overall process with sequenced steps of how the feedback process was.

Overall, the varying complexity of each exercise and the clear limitations of ChatGPT made it something extremely important and valuable the fact of introducing this feedback loop as described. This approach made the methodology very dynamic and helped to improve the code created by the

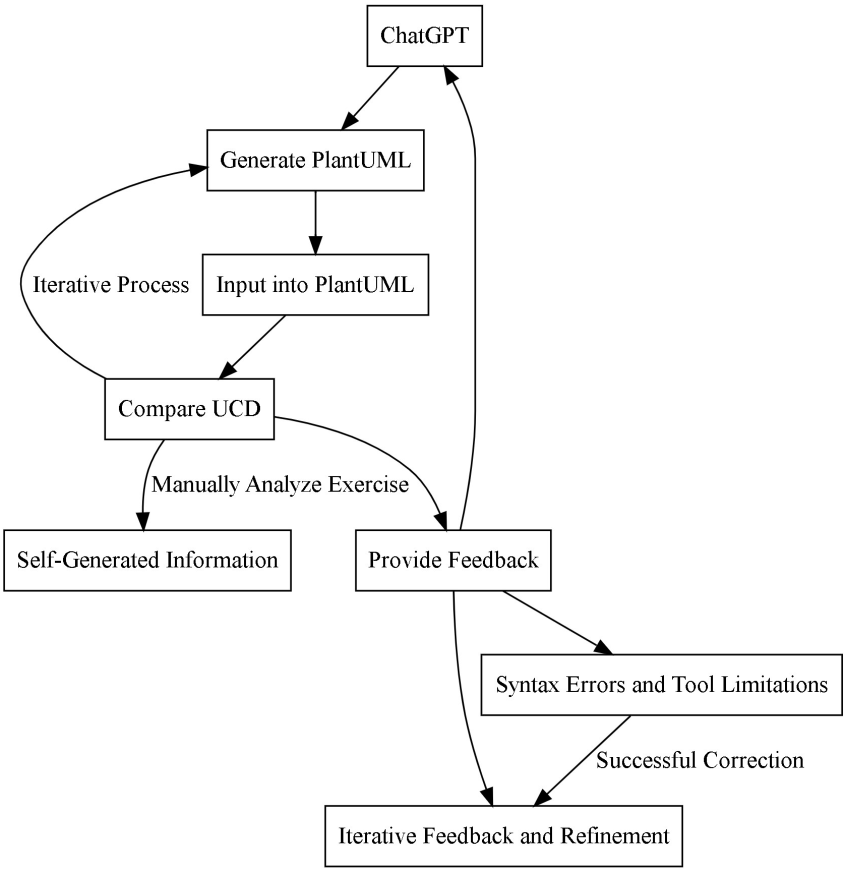


Figura 3.1. Feedback Process

model, which finally had a direct impact on the diagram that the user can perceive. So it can be stated that this methodology requires n adaptive approach in order to fully show a reliable solution to be further evaluated.

3.6 Integrations of Use Case Narratives

Unlike the integration of use case diagrams (UCDs), the use case narratives presented a unique set of complex issues when incorporated into the research approach. The main goal now is to get more into detail about this aspect and be able to show the analysis of the rules observed and soe of the most important difficulties faced, as well as the approaches used.

Refined Search Methodology:

Similar to UCDs, the search for use case narrative exercises placed a strong emphasis on reliability and consistency. However, attention shifted to exercises that shared descriptions with earlier identified UCD exercises. This attempt intended to establish a complementary relationship between UCDs and use case narratives for a more cohesive and comprehensive analysis.

Exercise Selection Criteria: Use case narrative exercises were chosen with special consideration, with a focus on relevancy and complexity. The selection was based on how well they fitted into a narrative format that could be comparable to the standards established by industry expert Alistair Cockburn. The selecting criteria made sure that the exercises included specific system behaviors, actor interactions, and detailed context descriptions.[1]

Narrative Subjectivity:

Use case narratives are characterized by nature more subjective than UCDs. This subjectivity is demonstrated in Cockburn's work through the use of several, though equally correct, narrative frameworks. The different writing styles, like whole prose sentences or a play-style framework, highlight the flexibility in creating narratives and make complex the analysis of them further.[1]

Assessment Challenges:

Because use case narratives are subjective, evaluating them added a layer of complexity. Cockburn[1] also provides examples that demonstrate how many narrative styles (not only structures) can coexist together. Due to this subjectivity, a more detailed review methodology was adopted, putting less emphasis on standard correctness ratings and more on the narrative's clarity and depth, which should be as simple as possible always.

Quality Criteria and Evaluation Approach:

Evaluating the use case narrative exercises was also a challenge, and as the aim of this investigation is also to show the highest standards and reliability, some important quality criteria were analyzed for the assessment of this type of exercises. The readability, clarity, and conciseness were the most important thing taken into account, and moreover, "pass/fail" tests were used to assess the quality and effectiveness of the narratives. These tests will be discussed more in detail in the sections that follow, when we present the results. The ChatGPT prompt designed, included some specific instructions for guiding ChatGPT to take into account important components, like the use case name and stakeholders, looking forward to obtain a complete and deliberate answer from the AI. In other words, we intended to guide the model to develop narratives that met the established quality standards. The prompt employed was the following:

"*The following description contains many use cases: "(DESCRIPTION OF THE EXERCISE)" One of the use cases identified in the text is "(NAME OF THE USE CASE)". I need you to analyze the text and to write an appropriate Use Case description of the use case "(NAME OF THE USE CASE)". Please take into account the following information when writing the use case description:*

- *“Use Case Name” is any name that is the simplest possible but descriptive too.*
- *"Scenario": Each use case contains the internal activities set which are unique and it is represented as a scenario .*
- *“Triggering event” is basically an event that is initialized by the primary actor to start use case execution.*
- *“Brief description”: is to describe the whole use case description in a short form.*
- *“Actor” can be a human user, a hardware device, or a software system that interacts with the system for goal accomplishment.*
- *“Related use case” represents the associated use cases.*
- *“Stakeholders” are the persons who are all involved in the successful execution of the system.*
- *“Precondition” is a condition that should be met before the use case starts.*
- *“Postcondition” is a condition that should be met after the use case completed successfully.*
- *“Flow of activities” is the tenth compartment of the use case description.*
- *It consists of two columns about the steps performed by the actor and the response of the system.*
- *“Exception conditions” describes the details of the other activities and the exception conditions.*"

This prompt was adopted by taking a deep look at the solutions of the use case narrative exercises from the book "Systems Analysis and Design" [27], where it clearly explained in a very brief and simplistic manner the main components of the solution narratives.

Differently with respect to the iterative process which was described for use case diagrams (UCDs), here, the choice was to employ a singular prompt for each use case narrative exercise.

Strategic Rationale Behind Singular Prompts:

This strategic decision was made because of the subjective qualities of the use case narrative exercises selected for this study. As the exercises featured detailed textual descriptions, they provided a rich amount of information in a single iteration. This strategy minimized the need for repetitive interactions but still accurately expressing the essence of each exercise.

The exercises that I selected were carefully chosen to mirror real world scenarios, showing a detailed textual environment. This context helped ChatGPT gather a lot of information in just one interaction. The model's ability to create detailed narratives from information-packed descriptions showed how effective the single prompt strategy was.

Qualitative Comparative Analysis:

After its generation, each of the use case narratives developed by ChatGPT was subject to a qualitative comparative analysis, highlighting main distinctions. The principal objective was not only identifying differences but also to offer a complete evaluation, knowing the contextual complexities relative to the narrative construction, by identifying both advantages and disadvantages in each solution. This procedure was made in order for the reader to have a deeper understanding of narrative efficacy. We looked at each solution based on how detailed the narrative was, how clear it was, and how well it matched the exercise's context.

Fig. 3.2 shows a diagram with the overall methodology with sequenced steps of how the the incorporation of use case narrative exercises was.

In overall, and similarly to the exercises of use case diagrams, the inclusion of narratives exercises also had a deliberate and refined approach, in which I tried to follow and stick to Cockburn's guidelines, emphasizing contextual relevance, and addressing the inherent subjectivity of narratives, which set the stage for a complex evaluation process. The challenges found provided valuable insights into the difficulties of narrative construction and evaluation, giving the possibility of a deeper exploration in the following sections.

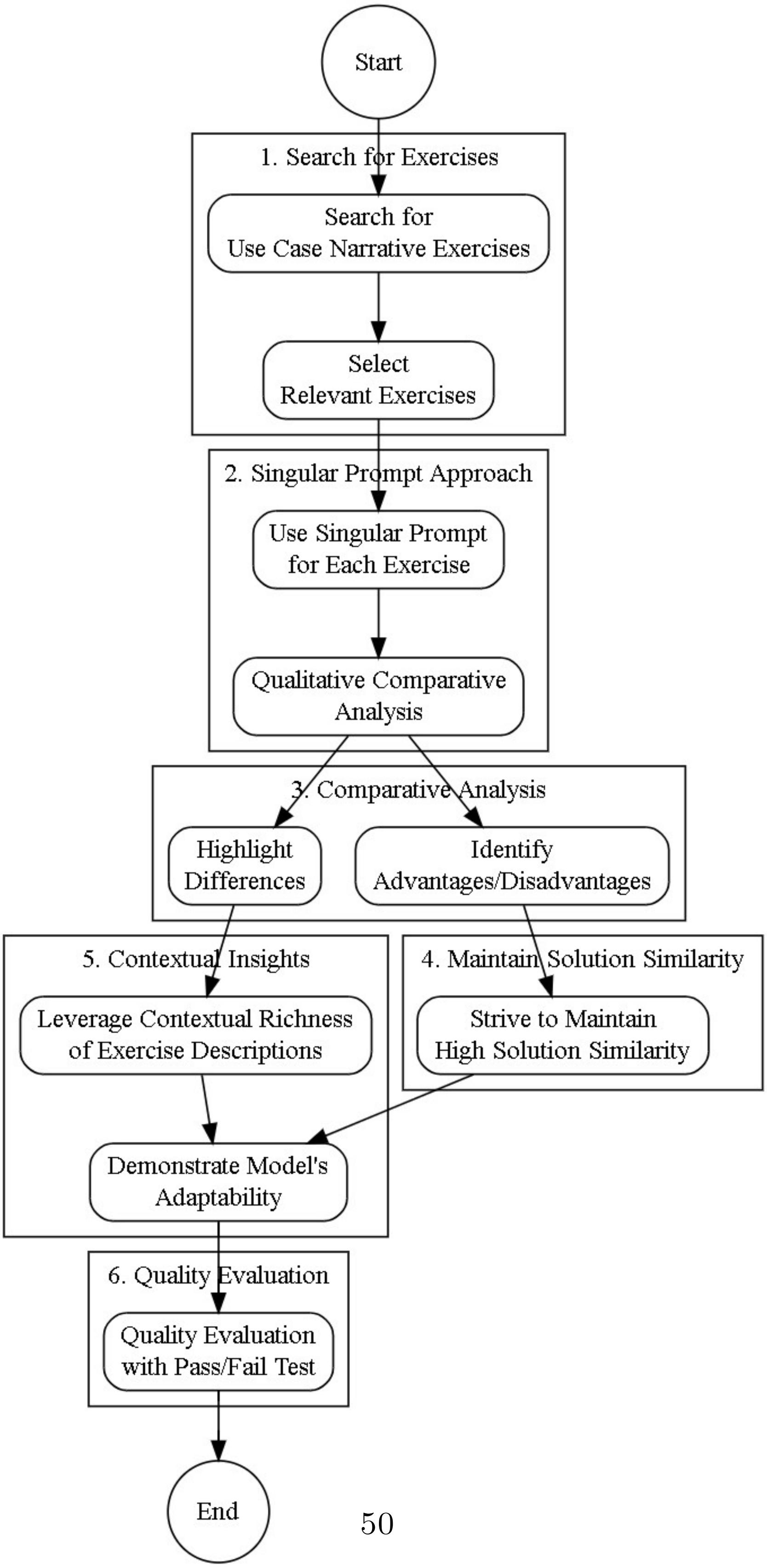


Figura 3.2. Use Case Narratives Exercises Methodology

3.7 Exercise Portfolio

In this section, all the exercises selected for the investigation will be shown, both for the UCD analysis and UC narrative analysis. Firstly, the exercises which provide the possibility to analyze UCD and narrative from the same problem description will be prioritized in order for the reader to have a more comprehensive understanding of the investigation. Afterwards, a subset of exercises will be exposed with the complete prompt and the representation of the Use Case Diagrams or Use Case Narratives and their respective comparative analysis. Since some prompts are too extensive, the complete set of exercises with their full prompt will be included inside a "Google Drive" open link, for any consultation. What will be shown here completely, is the qualitative comparative analysis that was made on each exercise. In this way, the reader has the possibility to see the number of each exercise, have a look at the analysis made and then consult on the exercise description and both the original solution and the generated solution by ChatGPT.

3.7.1 UCD Exercises

Something to take into consideration is that the description of each exercise will be explicitly written inside each prompt, after the phrase **"The following should be textually analyzed and a use case diagram created..."**. This is done in order to avoid repetition. Also, the link or the book/article were the exercises can be completely seen is provided in the "Google Drive" link mentioned before.

Exercise 1.

https://irm.com.au/use-use-cases/

PROMPT: The following should be textually analyzed and a use case diagram created containing several use cases. Identify the actors, use cases and associations. Please give me the PlantUML code for the use case diagram corresponding to the following text: *At the start of each semester a student can request a prospectus containing a course list. Information about a course is provided, such as the tutor, department and pre-requisites. The new system will allow students to create a schedule, then select four courses. Each student chooses two others in case their first choices become full or are cancelled. No course can have more than 10 students. No course can have less than 3 students or it will be cancelled. This will be the same functionality as* *available to other internal users of the system. When registration is complete, the registration system sends a message to the billing system to send out a bill to the student. Tutors use the system to find which classes they are teaching and who the students are. The registrar will administer the system. For a period at the beginning of the semester the student can change their schedule. Students must be allowed to access the system during this time to add or delete courses.*

ChatGPT

Sure, let's analyze the text and identify the actors, use cases, and associations.

Actors:

1. Student
2. Tutors
3. Registrar
4. Billing System

Use Cases:

1. Request Prospectus
2. View Course Information
3. Create Schedule
4. Select Courses
5. Modify Schedule
6. Send Registration to Billing System
7. View Teaching Schedule (for Tutors)
8. Administer System (for Registrar)

Associations:

1. Student requests a prospectus.
2. Student views course information.

3. Student creates a schedule.
4. Student selects courses.
5. Student modifies schedule during the specified period.
6. Registration system sends registration to the billing system.
7. Billing system sends out a bill to the student.
8. Tutors view their teaching schedule and associated students.
9. Registrar administers the system.

Now, let's create the PlantUML code for the use case diagram:

```
@startuml
left to right direction
actor Student
actor Tutors
actor Registrar
actor BillingSystem
rectangle "System"
usecase "Request Prospectus" as UC1
usecase "View Course Information" as UC2
usecase "Create Schedule" as UC3
usecase "Select Courses" as UC4
usecase "Modify Schedule" as UC5
usecase "Send Registration to Billing System" as UC6
rectangle "Internal Users"
usecase "View Teaching Schedule" as UC7
usecase "Administer System" as UC8
Student –> UC1
Student –> UC2
Student –> UC3
Student –> UC4
Student –> UC5
Student –> UC6
UC3 –> UC4
UC3 –> UC5
UC6 –> BillingSystem
Tutors –> UC7
```

```
Registrar –> UC8
@enduml
```

PROMPT: Shouldn't the registrar maintain course information, maintain tutors information, maintain curriculum and maintain student information for example? I mean, shouldn't these things be considered as use cases of the actor "registrar"?

ChatGPT

Certainly! If the responsibilities of the Registrar involve maintaining various information, we can indeed represent those as separate use cases for the Registrar actor. Here's an updated PlantUML code reflecting these additional use cases:

```
@startuml
left to right direction
actor Student
actor Tutors
actor Registrar
actor BillingSystem
rectangle "System"  usecase "Request Prospectus" as UC1
usecase "View Course Information" as UC2
usecase "Create Schedule" as UC3
usecase "Select Courses" as UC4
usecase "Modify Schedule" as UC5
usecase "Send Registration to Billing System" as UC6
rectangle "Internal Users"
usecase "View Teaching Schedule" as UC7
usecase "Administer System" as UC8
usecase "Maintain Course Information" as UC9
usecase "Maintain Tutors Information" as UC10
usecase "Maintain Curriculum" as UC11
usecase "Maintain Student Information" as UC12
Student –> UC1
Student –> UC2
Student –> UC3
Student –> UC4
Student –> UC5
Student –> UC6
UC3 –> UC4
UC3 –> UC5
UC6 –> BillingSystem
```

Tutors –> UC7

Registrar –> UC8

Registrar –> UC9

Registrar –> UC10

Registrar –> UC11

Registrar –> UC12

@enduml

Fig.3.3 and Fig.3.4 show both the original solution and the one generated with the UMLPlant code provided by ChatGPT.

From this exercise, we observe that the PlantUML code provided by ChatGPT generated a use case diagram that differed from the one in the original answer, but in terms of correctness, it is a good diagram. In completeness, both answers are substantially complete, but the original answer contained only one use case for the actor "student," seen as a summary use case of all the actions the text describes that the student can do. In contrast, ChatGPT's output gives a more detailed use case diagram for the actions associated with the student.

Analyzing the text, we realize that the use cases associated with the student, generated by the PlantUML code, align perfectly with the explicit description given. The original answer included some use cases related to the actor "registrar," not explicitly stated in the text but inferred from a commonsense perspective, essentially thought by human intelligence. For instance, the use case "maintain tutor's information" is not explicitly stated, but it is reasonable to think that the registrar must do this for the student to access information and register for courses.

Moreover, the code generated by ChatGPT provides two different systems: one called "internal users," where tutors and registrars can interact with the system, and the other called "system," where students can interact, along with the secondary actor, the billing system. Although this separation of use case diagrams is not necessary, it adds a higher level of detail, which is a positive aspect of ChatGPT's output.

After evaluating the output, I decided to prompt something new to ChatGPT, as shown in the prompt sequence. In this second case, it generated a very high-level diagram, encompassing complete use cases related to both the student and the registrar. It can be said that this final use case diagram is the best one provided in terms of quality indicators.

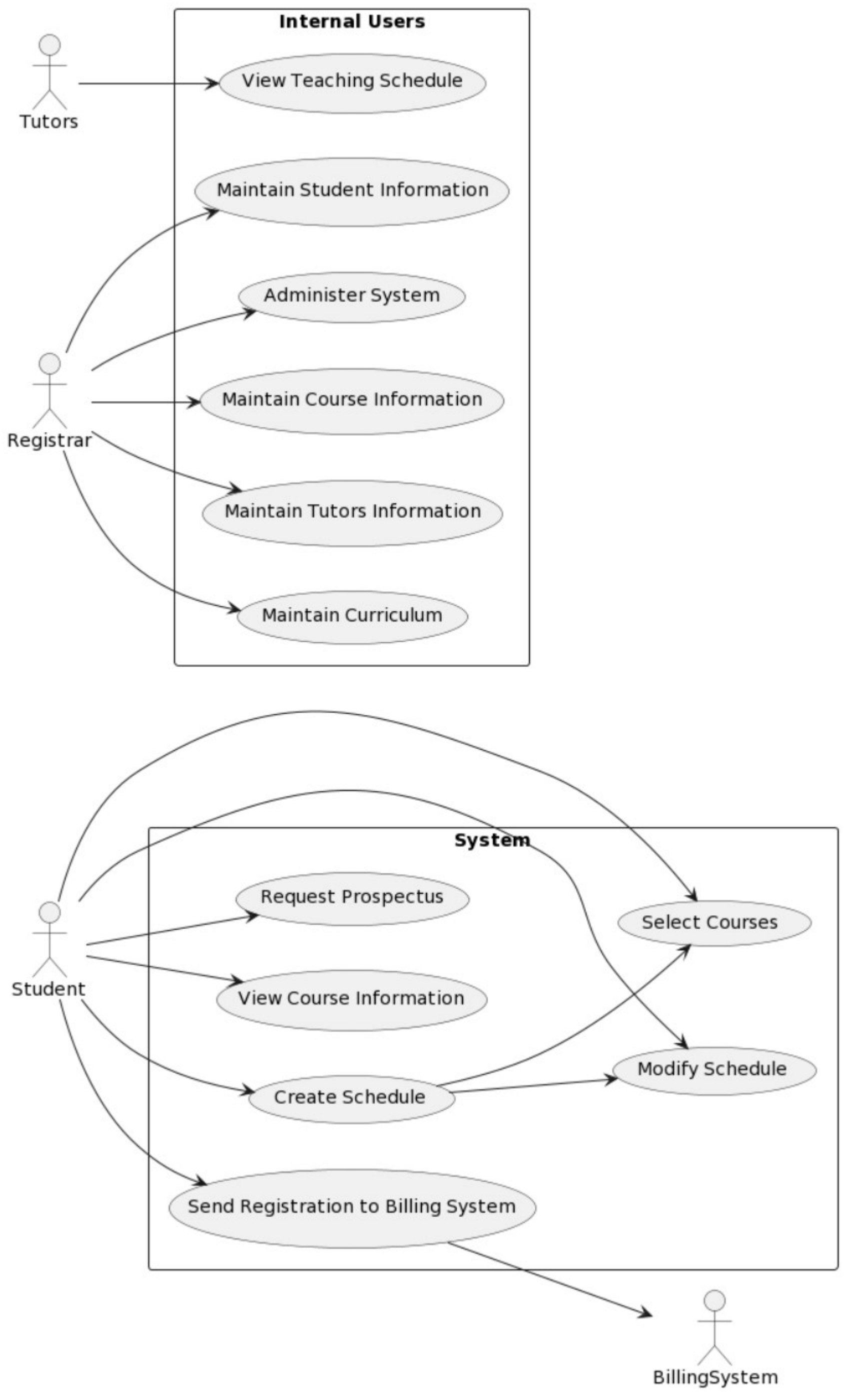


Figura 3.3. UCD generated with the PlantUML code provided by ChatGPT.

Exercise 2.

https://mygust.com/uploads/BOOK-Systems_analysis_and_design_in_a_

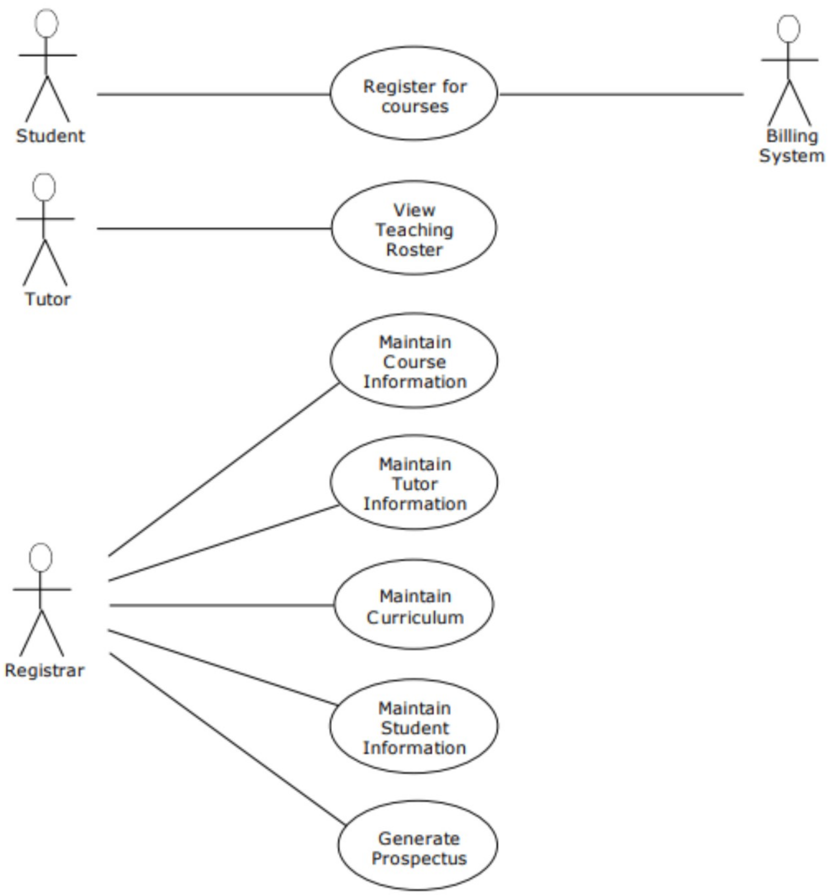


Figura 3.4. Original UCD provided by the exercise.

changin.pdf,https://mygust.com/wp-content/uploads/2014/12/Solutions-Chap03.pdf

PROMPT: The following should be textually analyzed and a use case diagram created containing several use cases. Identify the actors, use cases and associations. Please give me the PlantUML code for the use case diagram corresponding to the following text: *One of the functions of the Board of Realtors is to provide a Multiple Listing Service (MLS) system that supplies information that local real estate agents use to help them sell houses to their customers. During the month, agents list houses for sale (listings) by contracting with homeowners. The agent works for a real estate office, which sends information on the listing to the MLS. Therefore, any agent in the community can get information on the listing. Information on a listing includes the address, year built, square feet, number of bedrooms, number of* *bathrooms, owner name, owner phone number, asking price, and status code. At any time during the month, an agent might directly request information on listings that match customer requirements, so the agent contacts the MLS with the request. Information is provided on the house, on the agent who listed the house, and on the real estate office for which the agent works. For example, an agent might want to call the listing agent to ask additional questions or call the homeowner directly to make an appointment to show the house. Once each week, the MLS produces a listing book that contains information on all listings. These books are sent to some real estate agents. Some real estate agents want the books (which are easier to flip through), so they are provided even though the information is often out of date. Sometimes, agents and owners decide to change information about a listing, such as reducing the price, correcting previous information on the house, or indicating that the house is sold. The real estate office sends in these change requests to the MLS when the agent asks the office to do so.*

ChatGPT

Actors:

1. Real Estate Agent
2. Homeowner
3. Multiple Listing Service (MLS)
4. Real Estate Office

Use Cases:

1. List House for Sale (by Real Estate Agent)
2. Request Listing Information (by Real Estate Agent)
3. Produce Weekly Listing Book (by MLS)
4. Change Listing Information (by Real Estate Agent)

Associations:

1. Real Estate Agent lists houses for sale by contracting with homeowners.
2. Real Estate Agent sends information on the listing to the MLS.
3. Any agent can get information on a listing from the MLS.

4. Real Estate Agent requests information on listings that match customer requirements from the MLS.

5. MLS provides information on the house, the listing agent, and the real estate office.

6. MLS produces a weekly listing book containing information on all listings.

7. Some real estate agents receive the weekly listing book.

8. Real Estate Agent can request changes to listing information from the Real Estate Office.

9. Real Estate Office sends change requests to the MLS.

Now, let's create the PlantUML code for the use case diagram:

```
@startuml left to right direction
actor "Real Estate Agent" as Agent
actor Homeowner
actor "Multiple Listing Service (MLS)" as MLS
actor "Real Estate Office" as Office
rectangle "System"
usecase "List House for Sale" as UC1
usecase "Request Listing Information" as UC2
usecase "Produce Weekly Listing Book" as UC3
usecase "Change Listing Information" as UC4
Agent –> UC1
Agent –> UC2
MLS –> UC2
MLS –> UC3
Agent –> UC4
Office –> UC4
@enduml
```

PROMPT: I think that you are missing the "management" actor and I also think that you are missing too much use cases related to the actor "real estate officer". The PlantUML code you provided me is too basic. I need really much more detailed code. Analyze the whole text again and provide me a better code please.

ChatGPT

I apologize for the oversight. Let's analyze the text again and provide a more detailed use case diagram with the necessary actors and use cases.

Actors:

1. Real Estate Agent
2. Homeowner
3. Multiple Listing Service (MLS)
4. Real Estate Office
5. Management

Use Cases:

1. List House for Sale (by Real Estate Agent)
2. Request Listing Information (by Real Estate Agent)
3. Produce Weekly Listing Book (by MLS)
4. Receive Weekly Listing Book (by Real Estate Agent)
5. Change Listing Information (by Real Estate Agent)
6. Request Change to Listing Information (by Real Estate Office)
7. Approve Change Request (by Management)
8. Update Listing Information (by Real Estate Office)

Associations:

1. Real Estate Agent lists houses for sale by contracting with homeowners.
2. Real Estate Agent sends information on the listing to the MLS.
3. Any agent can get information on a listing from the MLS.
4. Real Estate Agent requests information on listings that match customer requirements from the MLS.
5. MLS provides information on the house, the listing agent, and the real estate office.

6. MLS produces a weekly listing book containing information on all listings.
7. Some real estate agents receive the weekly listing book.
8. Real Estate Agent can request changes to listing information from the Real Estate Office.
9. Real Estate Office sends change requests to the MLS.
10. Real Estate Office requests changes to listing information.
11. Management approves change requests.
12. Real Estate Office updates listing information.

Now, let’s create the PlantUML code for the improved use case diagram:

```
@startuml
left to right direction
actor "Real Estate Agent" as Agent
actor Homeowner
actor "Multiple Listing Service (MLS)" as MLS
actor "Real Estate Office" as Office
actor Management
rectangle "System"
usecase "List House for Sale" as UC1
usecase "Request Listing Information" as UC2
usecase "Produce Weekly Listing Book" as UC3
usecase "Receive Weekly Listing Book" as UC4
usecase "Change Listing Information" as UC5
usecase "Request Change to Listing Information" as UC6
usecase "Approve Change Request" as UC7
usecase "Update Listing Information" as UC8
Agent –> UC1
Agent –> UC2
MLS –> UC2
MLS –> UC3
Agent –> UC4
Agent –> UC5
Office –> UC5
Office –> UC6
```

Management –> UC7

Office –> UC8

@enduml

PROMPT: What about if you consider just 3 actors: real estate office, agent, and management? Give me a new code.

ChatGPT

Certainly, if we simplify the actors to just "Real Estate Office," "Agent," and "Management," and focus on the core interactions, the PlantUML code would look like this:

@startuml left to right direction

actor "Real Estate Office" as Office

actor "Real Estate Agent" as Agent

actor Management

rectangle "System"

usecase "List House for Sale" as UC1

usecase "Request Listing Information" as UC2

usecase "Produce Weekly Listing Book" as UC3

usecase "Receive Weekly Listing Book" as UC4

usecase "Change Listing Information" as UC5

usecase "Request Change to Listing Information" as UC6

usecase "Approve Change Request" as UC7

usecase "Update Listing Information" as UC8

Office –> UC1

Agent –> UC2

Office –> UC2

Office –> UC3

Agent –> UC4

Agent –> UC5

Office –> UC5

Office –> UC6

Management –> UC7

Office –> UC8

@enduml

Fig.3.5 and Fig.3.6 show both the original solution and the one generated with the UMLPlant code provided by ChatGPT.

This exercise offers an abundance of context and explanations regarding interactions with the system. Unfortunately, ChatGPT did not manage to provide a solution as close to the original one this time. Initially, the generated output code was overly summarized and significantly different from

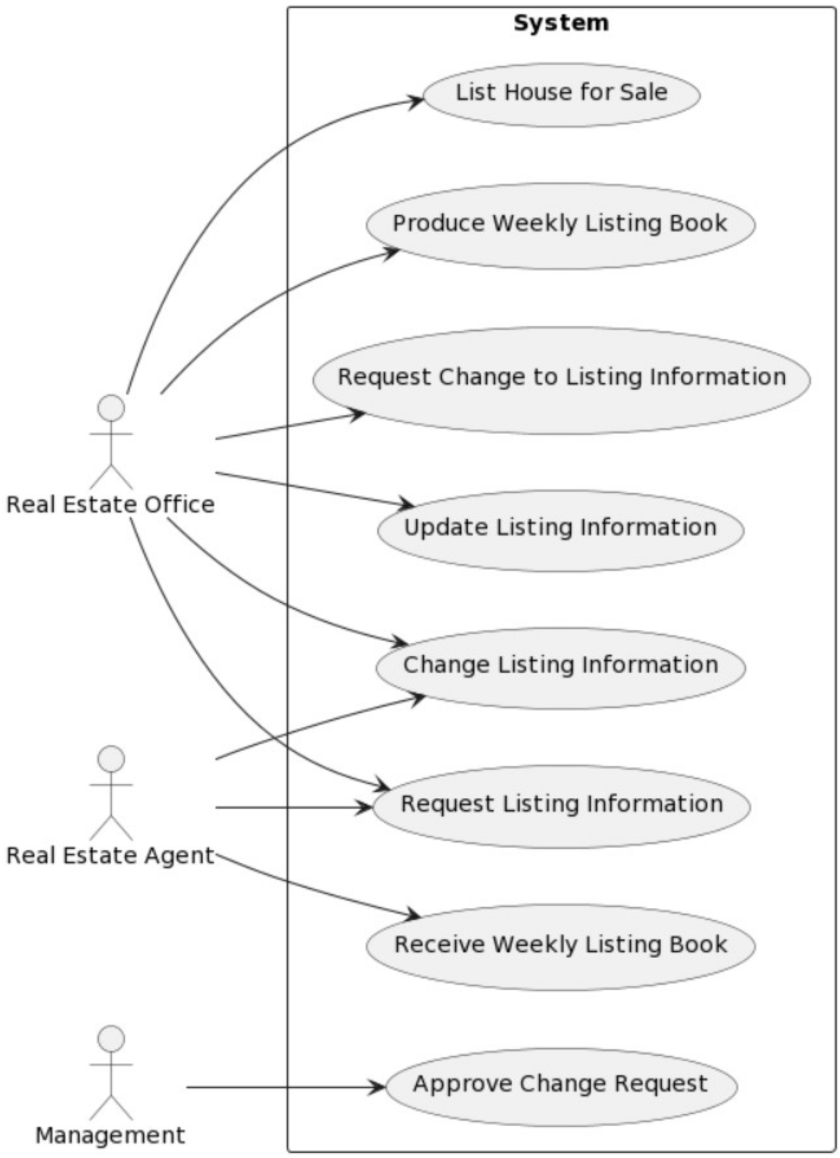


Figura 3.5. UCD generated with the PlantUML code provided by ChatGPT.

the actual answer. Something important to say, is that there were two new actors : the Multiple Listing Service (MLS) and the Homeowner. The MLS is a necessary actor, possibly omitted in the original answer, as the text clearly shows real interactions between the MLS and the system. However, the Homeowneris not really necessary for this diagram.

Moreover, the second code provided was somehow more detailed and specific but still was not comparable to the original solution. Maybe some of

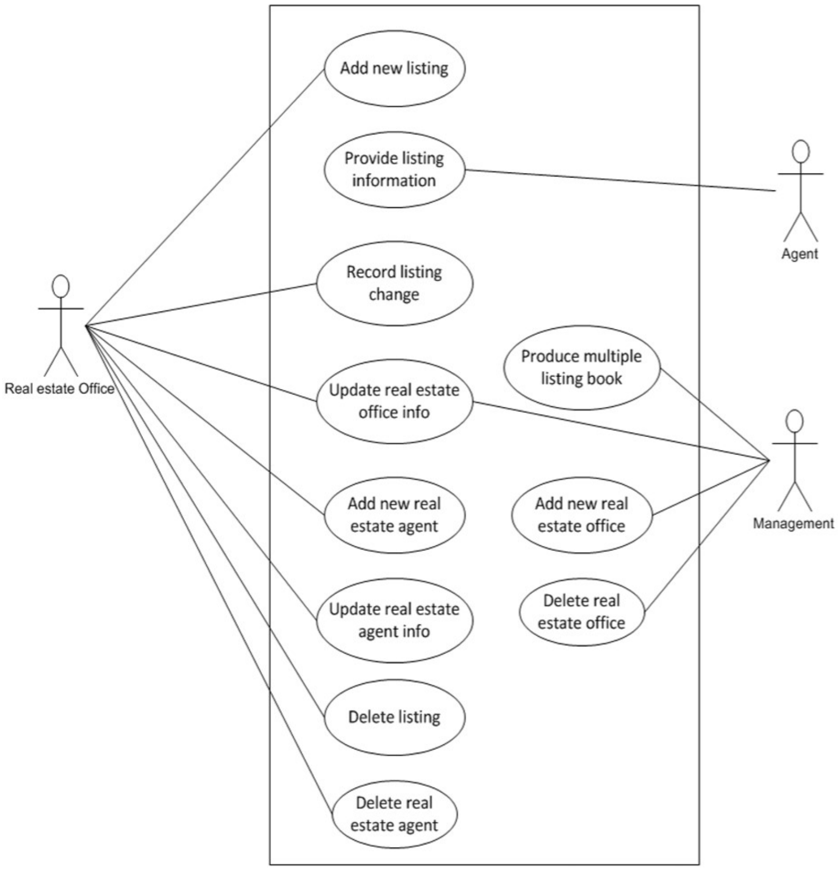


Figura 3.6. Original UCD provided by the exercise.

this differences were due to the inclusion of use cases associated with new actors that were not present in the original solution. Although the second code generated a Use Case Diagram (UCD) that was more or less correct in comparison to the text description, I wanted to explore what ChatGPT could produce if we instructed it to consider the same actors as in the original answer. Surprisingly, the third solution was the most close to the one given in the exercise, with a big mistake, which was identifying "receive weekly listing book" as an agent's goal, which it is not. This shows an error in the model's understanding and highlights a limitation when processing extensive descriptive contexts.

Something very important comparison between the original solution and ChatGPT's output is that the actor "management" went unidentified initially. Despite my request to include this actor, the model couldn't generate the associated use cases. While this stands as a limitation, it is a minor one, as even as a reader, it wasn't easy for me to identify this actor in the text. At this juncture, it becomes evident that when the textual description lacks explicitness, ChatGPT struggles to identify and incorporate these elements into the code.

Exercise 3.

https://mygust.com/uploads/BOOK-Systems_analysis_and_design_in_a_changin.pdf,https://mygust.com/wp-content/uploads/2014/12/Solutions-Chap03.pdf

PROMPT: The following should be textually analysed and a use case diagram created containing several use cases. Identify the actors, use cases and associations. Please give me the PlantUML code for the use case diagram corresponding to the following text: *Recall the On the Spot courier service introduced in Chapter 2. The details of the package pickup and delivery process are described here. When Bill got an order, at first, only on his phone, he recorded when he received the call and when the shipment would be ready for pickup. Sometimes, customers wanted immediate pickup; sometimes, they were calling to schedule a later time in the day for pickup. Once he arrived at the pickup location, Bill collected the packages. It was not uncommon for the customer to have several packages for delivery. In addition to the name and address of the delivery location, he also recorded the time of pickup. He noted the desired delivery time, the location of the delivery, and the weight of the package to determine the courier cost. When he picked up the package, he printed out a label with his portable printer that he kept in the delivery van. At first, Bill required customers to pay at the time of pickup, but he soon discovered that there were some regular customers who preferred to receive a monthly bill for all their shipments. He wanted to be able to accommodate those customers. Bills were due and payable upon receipt. To help keep track of all the packages, Bill decided that he needed to scan each package as it was sorted in the warehouse. This would enable him to keep good control of his packages and avoid loss or delays. The delivery of a package was fairly simple. Upon delivery, he would record information about when the delivery was made and who received it. Because some of the packages were valuable, it was necessary in those instances to have someone sign for the package.*

ChatGPT

Actors:

1. Courier (e.g., Bill)

2. Customer
3. Warehouse Personnel
4. Signee (for valuable packages)

Use Cases:

1. Receive Order Call (by Courier)
2. Schedule Pickup Time (by Customer)
3. Perform Immediate Pickup (by Courier)
4. Perform Scheduled Pickup (by Courier)
5. Collect Packages (by Courier)
6. Record Pickup Details (by Courier)
7. Determine Courier Cost (by Courier)
8. Print Label (by Courier)
9. Receive Payment at Pickup (by Courier)
10. Send Monthly Bill (by Courier)
11. Receive Package at Warehouse (by Warehouse Personnel)
12. Scan Package for Sorting (by Warehouse Personnel)
13. Deliver Package (by Courier)
14. Record Delivery Details (by Courier)
15. Sign for Valuable Package (by Signee)

Associations:

1. Courier receives order calls and records pickup details.
2. Customers schedule pickup times (immediate or scheduled later).
3. Courier collects packages from customers.

4. Courier records pickup details including time, delivery location, and package weight.
5. Courier determines the courier cost based on package details.
6. Courier prints a label for the package using a portable printer.
7. Courier receives payment at the time of pickup or sends monthly bills.
8. Warehouse personnel receive packages, scan for sorting, and keep track.
9. Courier delivers packages and records delivery details.
10. For valuable packages, someone (Signee) signs upon delivery.

Now, let's create the PlantUML code for the use case diagram:

```
@startuml left to right direction
actor "Courier" as Courier
actor "Customer" as Customer
actor "Warehouse Personnel" as WarehousePersonnel
actor "Signee" as Signee
rectangle "On the Spot Courier Service"
usecase "Receive Order Call" as UC1
usecase "Schedule Pickup Time" as UC2
usecase "Perform Immediate Pickup" as UC3
usecase "Perform Scheduled Pickup" as UC4
usecase "Collect Packages" as UC5
usecase "Record Pickup Details" as UC6
usecase "Determine Courier Cost" as UC7
usecase "Print Label" as UC8
usecase "Receive Payment at Pickup" as UC9
usecase "Send Monthly Bill" as UC10
usecase "Receive Package at Warehouse" as UC11
usecase "Scan Package for Sorting" as UC12
usecase "Deliver Package" as UC13
usecase "Record Delivery Details" as UC14
usecase "Sign for Valuable Package" as UC15
Courier –> UC1
Courier –> UC2
Courier –> UC3
Courier –> UC4
```

Courier –> UC5
Courier –> UC6
Courier –> UC7
Courier –> UC8
Courier –> UC9
Courier –> UC10
WarehousePersonnel –> UC11
WarehousePersonnel –> UC12 Courier –> UC13
Courier –> UC14
Signee –> UC15
Customer –> UC2
Customer –> UC9
@enduml

PROMPT: I think that "recieve order call" is not a goal of the courier and should not be included as a use case, and I think that "recieve package at warehouse" is also not a goal for the warehouse personnel so it should not be included as a use case also.

ChatGPT

You are correct, and I appreciate your clarification. Let's refine the use case diagram by removing "Receive Order Call" and "Receive Package at Warehouse" as separate use cases. Instead, we can integrate these actions into existing use cases. Here is the updated PlantUML code:

@startuml
left to right direction
actor "Courier" as Courier
actor "Customer" as Customer
actor "Warehouse Personnel" as WarehousePersonnel
actor "Signee" as Signee
rectangle "On the Spot Courier Service"
usecase "Schedule Pickup Time" as UC2
usecase "Perform Immediate Pickup" as UC3
usecase "Perform Scheduled Pickup" as UC4
usecase "Collect Packages" as UC5
usecase "Record Pickup Details" as UC6
usecase "Determine Courier Cost" as UC7
usecase "Print Label" as UC8
usecase "Receive Payment at Pickup" as UC9
usecase "Send Monthly Bill" as UC10
usecase "Scan Package for Sorting" as UC12

```
usecase "Deliver Package" as UC13
usecase "Record Delivery Details" as UC14
usecase "Sign for Valuable Package" as UC15
Courier –> UC2
Courier –> UC3
Courier –> UC4
Courier –> UC5
Courier –> UC6
Courier –> UC7
Courier –> UC8
Courier –> UC9
Courier –> UC10
WarehousePersonnel –> UC12
Courier –> UC13
Courier –> UC14
Signee –> UC15
Customer –> UC2
Customer –> UC9
@enduml
```

Fig.3.7 and Fig.3.8 show both the original solution and the one generated with the UMLPlant code provided by ChatGPT.

For this exercise, ChatGPT did not produce the outcome that we could have expected. The long and extense textual description was a challenge for the AI, and this generated it to have significant semantic errors that impacted the correctness of the Use Case Diagram (UCD). It was evidenced that the AI struggled to accurately identify actors, associations, and included a huge number of unnecessary use cases for the actor "courier."

Even though that we inputted additional information in the second prompt to address the unnecessary use cases for the courier (which were not actual goals), ChatGPT only took into account the specific use cases I had signaled. Unfortunately, it failed to identify the broader distinction between real goals of the actors and actions that were not genuine goals.

Furthermore, an additional error arose when ChatGPT incorrectly associated some use cases related to the delivery person with the character "Bill," who was not the actual delivery person. Also, we may see that a new actor named the "signee" was identified, and it could have been recognized as the customer, giving the reader a misunderstanding about who actually is signing. This tells us that if the person prompting ChatGPT doesn't have a big knowledge of use cases, it could lead to a confusion.

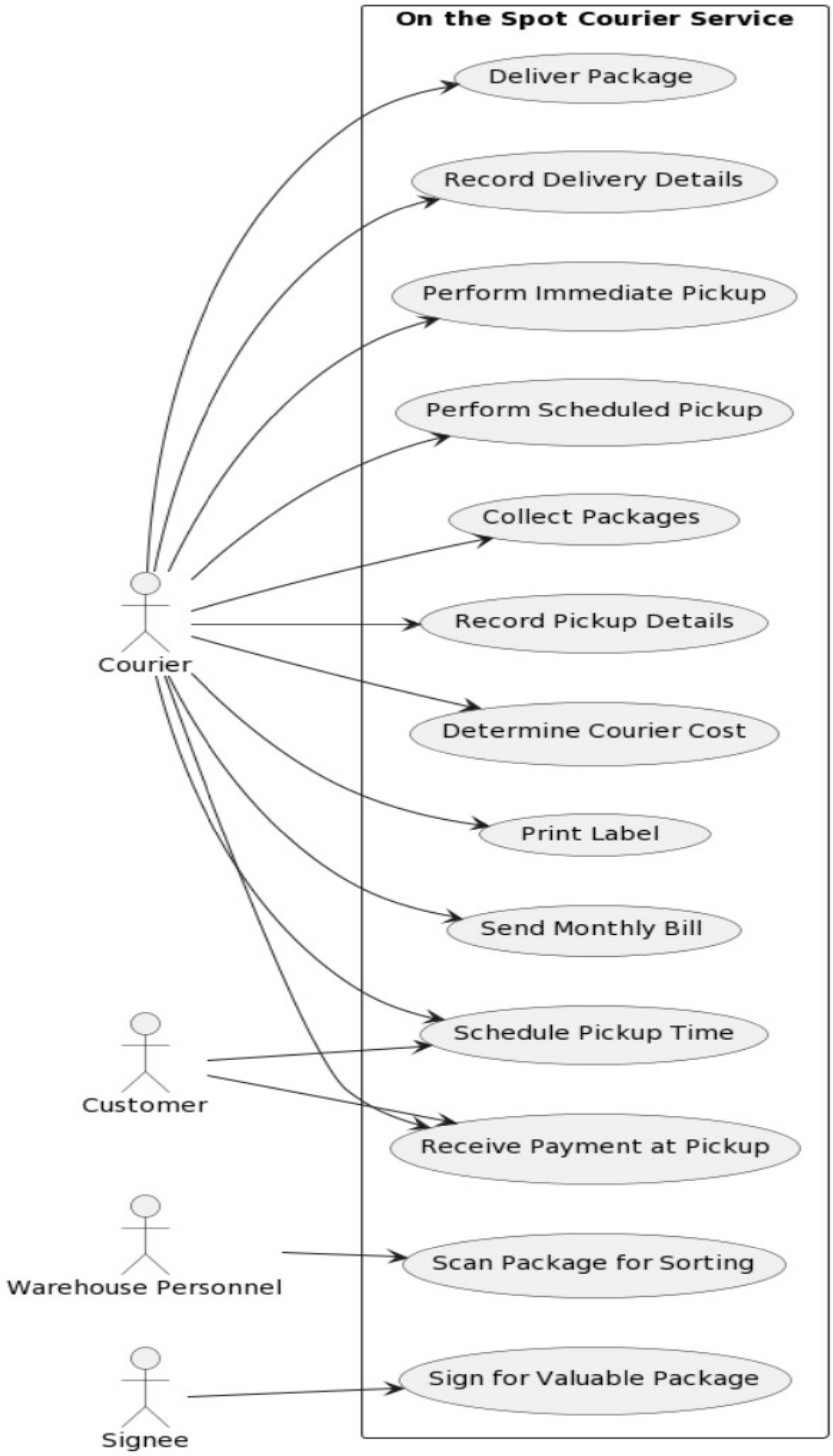


Figura 3.7. UCD generated with the PlantUML code provided by ChatGPT.

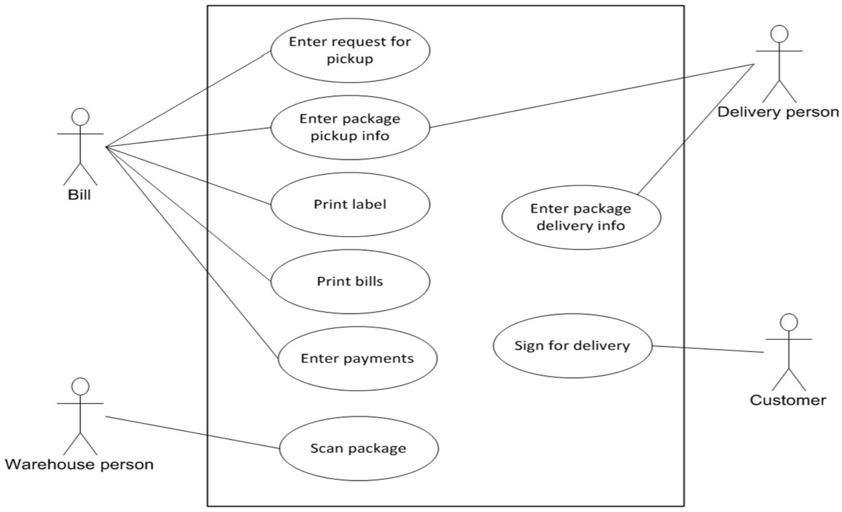


Figura 3.8. Original UCD provided by the exercise.

In the other hand, if the person providing the prompts has some basic knowledge of use cases, they may be able to re-inquire, correcting certain aspects and providing additional feedback to obtain a more refined code, although not entirely correct.

Exercise 4.

https://cmps-people.ok.ubc.ca/bowenhui/310/8-UML.pdf

PROMPT: *The following should be textually analysed and a use case diagram created containing several use cases. Identify the actors, use cases and associations. Please, use the user goal level approach. Also, please consider any possible generalization relationship between use cases or between actors, and any possible "include" or "extend" relationship between use cases. Please give me the PlantUML code for the use case diagram corresponding to the following text: This case study concerns a simplified system of the automatic teller machine (ATM). The ATM offers the following services: 1. Distribution of money to every holder of a smart-card via a card reader and a cash dispenser. 2. Consultation of account balance, cash and cheque deposit facilities for bank customers who hold a smart-card from their bank. Do not forget either that: 3. All transactions are made secure. 4. It is sometimes necessary to refill the dispenser, etc. From these four sentences, we will work through the following activities: • Identify the actors, • Identify the use cases, • Construct a use case diagram, • Write a textual description of the use* *cases, • Complete the descriptions with dynamic diagrams, • Organise and structure the use cases. What are the external entities that interact directly with the ATM? Let's look at each of the sentences of the exposition in turn. Sentence 1 allows us to identify an obvious initial actor straight away: every "holder of a smart-card". He or she will be able to use the ATM to withdraw money using his or her smart-card. However, be careful: the card reader and cash dispenser constitute part of the ATM. They can therefore not be considered as actors! You can note down that the identification of actors requires the boundary between the system being studied and its environment to be set out exactly. If we restrict the study to the control/ command system of physical elements of the ATM, the card reader and cash dispenser then become actors. Another trap: is the smart-card itself an actor? The card is certainly external to the ATM, and it interacts with it... Yet, we do not recommend that you list it as an actor, as we are putting into practice the following principle: eliminate "physical" actors as much as possible to the advantage of "logical" actors. The actor is the who or what that benefits from using the system. It is the card holder who withdraws money to spend it, not the card itself! Sentence 2 identifies additional services that are only offered to bank customers who hold a smart-card from this bank. This is therefore a different profile from the previous one, which we will realise by a second actor called Bank customer. Sentence 3 encourages us to take into account the fact that all transactions are made secure. But who makes them secure? There are therefore other external entities, which play the role of authorisation system and with which the ATM communicates directly. An interview with the domain expert4 is necessary to allow us to identify two different actors: • the Visa authorisation system (VISA AS) for withdrawal transactions carried out using a Visa smart-card (we restrict the ATM to Visa smart-cards for reasons of simplification); • the information system of the bank (Bank IS) to authorise all transactions carried out by a customer using his or her bank smart-card, but also to access the account balance. Finally, sentence 4 reminds us that an ATM also requires maintenance work, such as refilling the dispenser with bank notes, retrieving cards that have been swallowed, etc. These maintenance tasks are carried out by a new actor, which – to simplify matters – we will call Maintenance operator.*

ChatGPT

Based on the information provided, here's the textual analysis and the corresponding PlantUML code for the use case diagram:

Actors:

1. Holder of a Smartcard
2. Bank Customer
3. Visa Authorization System (VISA AS)
4. Bank Information System (Bank IS)
5. Maintenance Operator

Use Cases:

1. Withdraw Money
2. Consult Account Balance
3. Deposit Cash
4. Deposit Cheque
5. Secure Transactions
6. Refill Dispenser
7. Retrieve Swallowed Card

Associations:

- Holder of a Smart-card (Actor) interacts with:
- Withdraw Money (Use Case)
- Consult Account Balance (Use Case)
- Deposit Cash (Use Case)
- Deposit Cheque (Use Case)
- Secure Transactions (Use Case)
- Bank Customer (Actor) interacts with:
- Consult Account Balance (Use Case)
- Visa Authorization System (VISA AS) (Actor) interacts with:
- Withdraw Money (Use Case)

- Bank Information System (Bank IS) (Actor) interacts with:
- Consult Account Balance (Use Case)
- Secure Transactions (Use Case)
- Maintenance Operator (Actor) interacts with:
- Refill Dispenser (Use Case)
- Retrieve Swallowed Card (Use Case)

Generalization Relationships:

- Withdraw Money (Use Case) and Consult Account Balance (Use Case) can be generalized under a common Use Case named "Basic Transaction."

Here's the PlantUML code:

```
@startuml
left to right direction
actor "Holder of a Smartcard" as Holder
actor "Bank Customer" as BankCustomer
actor "Visa Authorization System (VISA AS)" as VisaAS
actor "Bank Information System (Bank IS)" as BankIS
actor "Maintenance Operator" as Operator
rectangle "ATM"
usecase "Withdraw Money" as Withdraw
usecase "Consult Account Balance" as ConsultBalance
usecase "Deposit Cash" as DepositCash
usecase "Deposit Cheque" as DepositCheque
usecase "Secure Transactions" as SecureTransactions
usecase "Refill Dispenser" as RefillDispenser
usecase "Retrieve Swallowed Card" as RetrieveCard
Holder –> Withdraw
Holder –> ConsultBalance
Holder –> DepositCash
Holder –> DepositCheque
Holder –> SecureTransactions
BankCustomer –> ConsultBalance
VisaAS –> Withdraw
```

BankIS –> ConsultBalance
BankIS –> SecureTransactions
Operator –> RefillDispenser
Operator –> RetrieveCard
Withdraw <|.. ConsultBalance : «generalization»
@enduml

PROMPT: Thank you. but I think that Visa AS and Bank IS should be clearly identified as secondary actors, and you should consider that a cardholder can only withdraw money, cannot deposit and cannot consult balance, on the other hand, the bank customer is able to do everything (deposit, consult balance and withdraw cash). Also, the use case "secure transactions" should not exist, since you are confusing it with the fact that they are secured by the visa AS and bank IS.

ChatGPT

Certainly, I appreciate your feedback. Here's the revised PlantUML code considering the provided clarifications:

@startuml
left to right direction
actor "Cardholder" as Cardholder
actor "Bank Customer" as BankCustomer
actor "Maintenance Operator" as Operator
rectangle "ATM" usecase "Withdraw Money" as Withdraw
usecase "Deposit Cash" as DepositCash
usecase "Deposit Cheque" as DepositCheque
rectangle "Authorization System"
usecase "Authorize Visa Transaction" as AuthorizeVisa
usecase "Authorize Bank Transaction" as AuthorizeBank
rectangle "Maintenance"
usecase "Refill Dispenser" as RefillDispenser
usecase "Retrieve Swallowed Card" as RetrieveCard
Cardholder –> Withdraw
Cardholder –> DepositCash
Cardholder –> DepositCheque
BankCustomer –> Withdraw
BankCustomer –> DepositCash
BankCustomer –> DepositCheque
AuthorizeVisa –> Withdraw
AuthorizeBank –> Withdraw
Operator –> RefillDispenser

```
Operator –> RetrieveCard
@enduml
```

PROMPT: You are still considering that a cardholder can deposit, but I told you that the cardholder is only able to withdraw cash, while the bank customer is the one who is able to do everything. Apart from this, the bank IS is a secondary actor that should support for deposits and consulting balance, while the visa AS is a secondary actor that only supports for withdrawing money.

ChatGPT

I appreciate the clarification. Here's the revised PlantUML code based on your feedback:

```
@startuml
left to right direction
actor "Cardholder" as Cardholder
actor "Bank Customer" as BankCustomer
actor "Maintenance Operator" as Operator
actor "Bank Information System" as BankIS
actor "Visa Authorization System" as VisaAS
rectangle "ATM"
usecase "Withdraw Cash" as Withdraw
rectangle "Bank Services"
usecase "Deposit Cash" as DepositCash
usecase "Deposit Cheque" as DepositCheque
usecase "Consult Balance" as ConsultBalance
rectangle "Authorization Systems"
usecase "Authorize Visa Transaction" as AuthorizeVisa
usecase "Authorize Bank Transaction" as AuthorizeBank
rectangle "Maintenance"
usecase "Refill Dispenser" as RefillDispenser
usecase "Retrieve Swallowed Card" as RetrieveCard
Cardholder –> Withdraw
BankCustomer –> Withdraw
BankCustomer –> DepositCash
BankCustomer –> DepositCheque
BankCustomer –> ConsultBalance
BankIS –> DepositCash
BankIS –> ConsultBalance
VisaAS –> Withdraw
Operator –> RefillDispenser
```

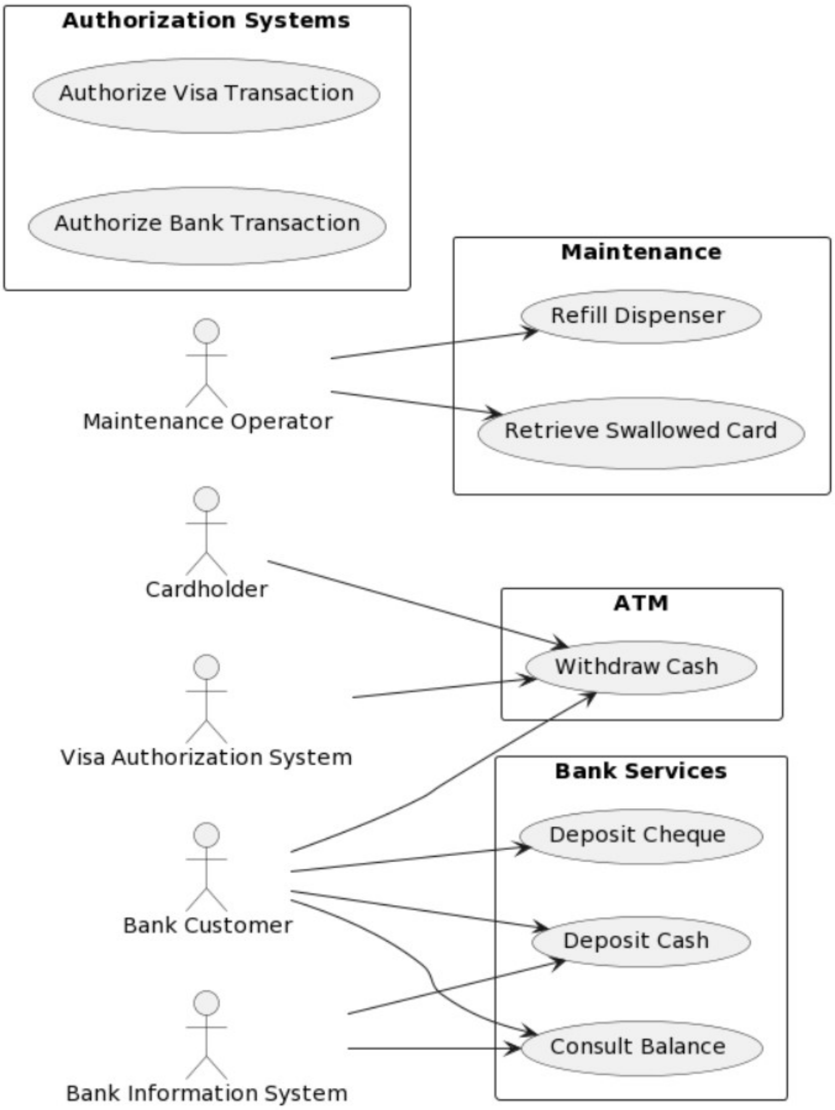


Figura 3.9. UCD generated with the PlantUML code provided by ChatGPT.

Operator –> RetrieveCard
@enduml

Fig.3.9, Fig.3.10 and Fig.3.11 show both the original solution and the one generated with the UMLPlant code provided by ChatGPT.

This exercise, which included an extensive textual explanation, needed three prompts to get a better outcome. The Use Case Diagram (UCD) was acceptable but the first code that was generated had some undesirable errors. One significant issue was that ChatGPT misinterpreted the use cases related

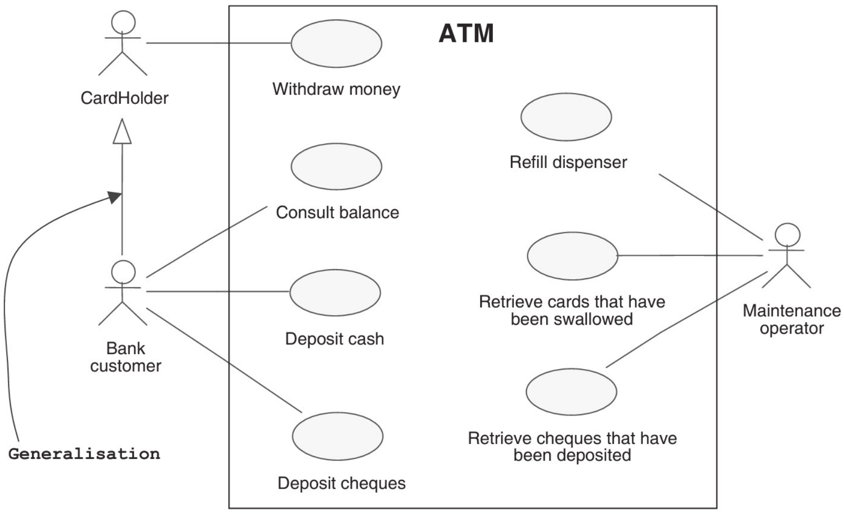


Figura 3.10. Original UCD provided by the exercise (First Version).

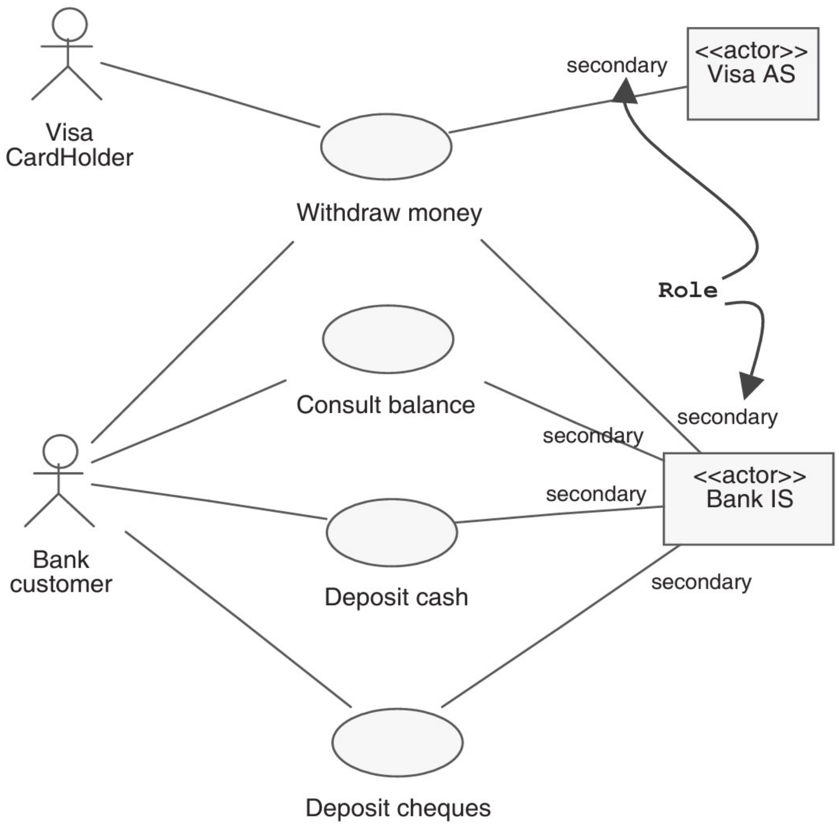


Figura 3.11. Original UCD provided by the exercise (Second Version).

to both bank and card clients, incorrectly giving to cardholders capabilities like balance inquiries and deposits that were actually aimed for bank customers. Even though this mistake is not very serious, it might lead to confusion contemplating how long the exercise is. Thankfully, the final code corrected this error.

Another inconvenience arose when ChatGPT included a use case for "secure transactions" associated with the bank customer, which was unnecessary and inconsistent with the text's description. In the text it clearly described that transactions were made through the VISA Authorization System and Bank Information System. In the final code, this issue was solved.

The last notable point is that both the VISA Authorization System and the Bank Information System are secondary actors. This was not made evident in the first code that was generated. However, taking into account the long description, we can conclude that, if the user had some understanding of Use Case Diagram (UCD) design, ChatGPT proved helpful in producing a preliminary code that could be further refined by the user.

Exercise 5.

https://web.uettaxila.edu.pk/CMS/AUT2010/seSREbs/notes%5CLecture_12%20SRE.pdf

PROMPT: *The following should be textually analyzed and a use case diagram created containing several use cases. Identify the actors, use cases and associations. Please, use the user goal level approach. Also, please consider any possible generalization relationship between use cases or between actors, and any possible "include" or "extend" relationship between use cases. Please give me the PlantUML code for the use case diagram corresponding to the following text: Problem Description: A bank has several automated teller machines (ATMs), which are geographically distributed and connected via a wide area network to a central server. Each ATM machine has a card reader, cash dispenser, a keyboard/display, and a receipt printer. By using the ATM machine, a customer can withdraw cash from either checking or savings account, query the balance of an account, or transfer funds from one account to another. A transaction is initiated when a customer inserts an ATM card into the card reader. Encoded on the magnetic strip on the back of the ATM card are the card number, the start date, and the expiration date. Assuming the card is recognized, the system validates the ATM card to determine: that the expiration date has not passed, that the user-entered PIN matches the PIN maintained by the system, that the card is not lost or* *stolen. The customer is allowed three attempts to enter the correct PIN; the card is confiscated if the third attempt fails. Cards that have been reported lost or stolen are also confiscated. If the PIN is validated satisfactorily, the customer is prompted for a withdrawal, query, or transfer transaction. Before a withdrawal transaction can be approved, the system determines that sufficient funds exist in the requested account, that the maximum daily limit will not be exceeded, and that there are sufficient funds available at the local cash dispenser. If the transaction is approved, the requested amount of cash is dispensed, a receipt is printed containing information about the transaction, and the card is ejected. Before a transfer transaction can be approved, the system determines: that the customer has at least two accounts and that there are sufficient funds in the account to be debited. For approved query and transfer requests, a receipt is printed, and the card ejected. A customer may cancel a transaction at any time; the transaction is terminated, and the card is ejected. Customer records, account records, and debit card records are all maintained at the server. An ATM operator may start up and close down the ATM to replenish the ATM cash dispenser and for routine maintenance. It is assumed that functionality to open and close accounts and to create, update, and delete customer and debit card records is provided by an existing system and is not part of this problem.*

ChatGPT

Based on the provided text, I've identified the actors, use cases, and associations for the ATM system. Here is the PlantUML code for the corresponding use case diagram:

```
@startuml left to right direction
actor "Customer" as Customer
actor "ATM Operator" as Operator
database "Central Server"
folder "Customer Records"
[Maintain Customer Records]
[Maintain Debit Card Records] [Maintain Account Records]
rectangle "ATM Machine"
usecase "Withdraw Cash" as Withdraw
usecase "Query Balance" as QueryBalance
usecase "Transfer Funds" as TransferFunds
usecase "Cancel Transaction" as CancelTransaction
Operator –> [Maintain Customer Records]
Operator –> [Maintain Debit Card Records]
Operator –> [Maintain Account Records]
```

Customer –> Withdraw
Customer –> QueryBalance
Customer –> TransferFunds
Customer –> CancelTransaction
Withdraw –> [Validate ATM Card]
QueryBalance –> [Validate ATM Card]
TransferFunds –> [Validate ATM Card]
CancelTransaction –> [Terminate Transaction]
[Validate ATM Card] –> [Check Expiration Date]
[Validate ATM Card] –> [Check PIN]
[Validate ATM Card] –> [Check Card Status]
[Check PIN] –> [Card Confiscated]
[Check PIN] –> Withdraw
[Check PIN] –> QueryBalance
[Check PIN] –> TransferFunds
[Withdraw] –> [Check Account Balance]
[Withdraw] –> [Check Daily Limit] [Withdraw] –> [Check Cash Availability]
[Withdraw] –> [Dispense Cash]
[Withdraw] –> [Print Receipt]
[Withdraw] –> [Eject Card]
[TransferFunds] –> [Check Multiple Accounts]
[TransferFunds] –> [Check Debit Account Balance]
[TransferFunds] –> [Print Receipt]
[TransferFunds] –> [Eject Card]
[CancelTransaction] –> [Terminate Transaction]
[CancelTransaction] –> [Eject Card]
@enduml

PROMPT: This is too messy, Please consider the following summarized information of the problem: The customer is able to: Withdraws funds from the checking or savings account, Query the balance of an account, Transfer funds from one account to another. Consider that the action of PIN validation is common to all these 3 actions. The ATM customer interacts with the system via the ATM card reader, keyboard/display, cash dispenser, and receipt printer. Regarding the operator interactions: Shuts down the ATM, Replenishes the ATM cash dispenser, and Starts the ATM.

ChatGPT

Certainly! Here is a more concise and simplified PlantUML code based on the summarized information:

```
@startuml left to right direction
actor "Customer" as Customer
actor "ATM Operator" as Operator
rectangle "ATM System"
usecase "Withdraw Funds" as Withdraw
usecase "Query Balance" as QueryBalance
usecase "Transfer Funds" as TransferFunds
usecase "Shutdown ATM" as ShutdownATM
usecase "Replenish Cash" as ReplenishCash
usecase "Start ATM" as StartATM
Customer –> Withdraw
Customer –> QueryBalance
Customer –> TransferFunds
Withdraw –> [Validate PIN]
QueryBalance –> [Validate PIN]
TransferFunds –> [Validate PIN]
[Validate PIN] –> [Common Validation]
Operator –> ShutdownATM
Operator –> ReplenishCash
Operator –> StartATM
@enduml
```

PROMPT: It is good, just that I think that the relationship between the 3 use cases of the customer and the Validate PIN use case should be with an "include" dashed arrow, since the PIN validation use case is included within the other 3 use cases, meaning that in order to achieve those actions, I first need to validate pin.

ChatGPT

Absolutely, I appreciate the clarification. Here is the updated PlantUML code with "include" relationships between the customer actions and the "Validate PIN" use case:

```
@startuml left to right direction
actor "Customer" as Customer
actor "ATM Operator" as Operator
rectangle "ATM System"
usecase "Withdraw Funds" as Withdraw
usecase "Query Balance" as QueryBalance
usecase "Transfer Funds" as TransferFunds
usecase "Validate PIN" as ValidatePIN
usecase "Shutdown ATM" as ShutdownATM
```

```
usecase "Replenish Cash" as ReplenishCash
usecase "Start ATM" as StartATM
Customer –> Withdraw
Customer –> QueryBalance
Customer –> TransferFunds
Withdraw –> (ValidatePIN)
QueryBalance –> (ValidatePIN)
TransferFunds –> (ValidatePIN)
(ValidatePIN) –> [Common Validation] : «include»
Operator –> ShutdownATM
Operator –> ReplenishCash
Operator –> StartATM
@enduml
```

Fig.3.12 and Fig.3.13 show both the original solution and the one generated with the PlantUML code provided by ChatGPT.

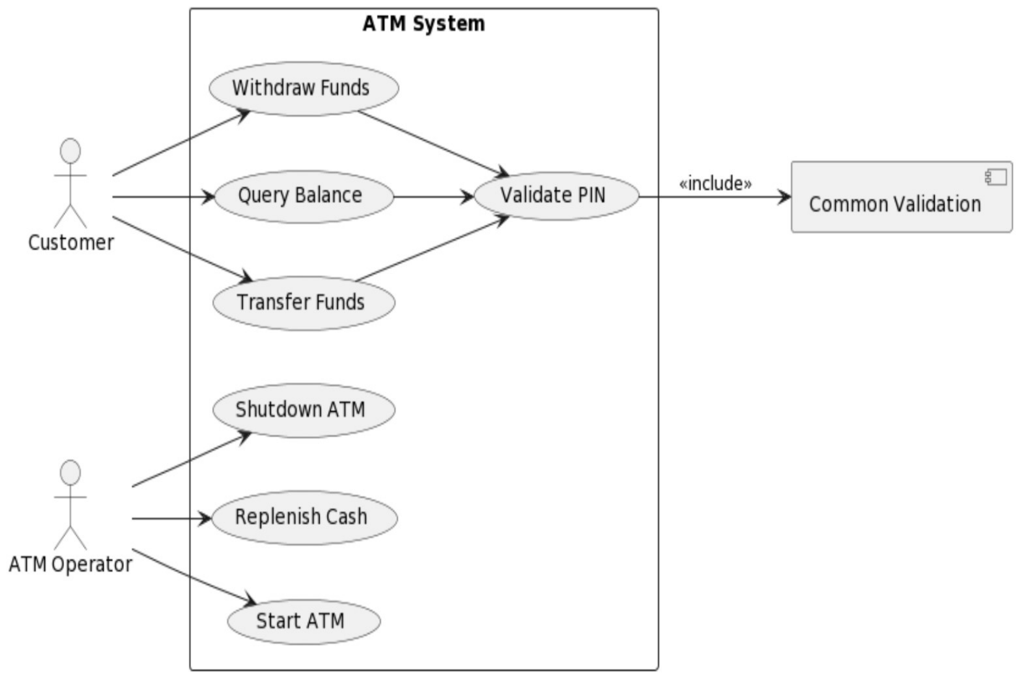


Figura 3.12. UCD generated with the PlantUML code provided by ChatGPT.

This exercise is very similar to EXERCISE 4, but with some slight differences in the actors, use cases, and problem description. The approach that I thought better was to verify ChatGPT's performance using two different methods. First, the problem description was given in the full form,

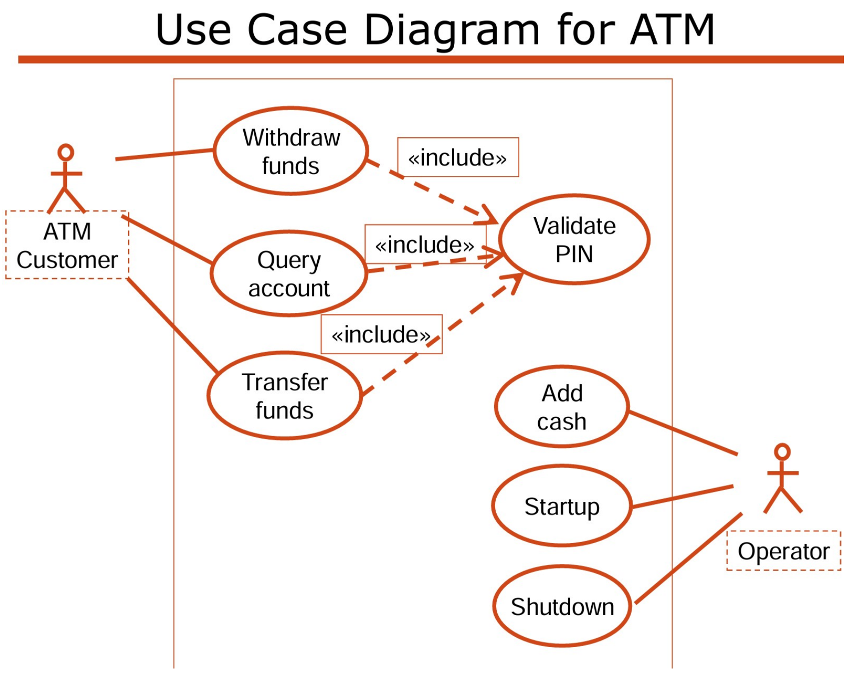


Figura 3.13. Original UCD provided by the exercise.

and ChatGPT was given the task of providing the necessary code. In this first instance, the resulting code was extremely disorganized, making the Use Case Diagram (UCD) incomprehensible. It included a huge amount of absolutely unneeded use cases and pointless associations. Having this result, I prompted ChatGPT once more, adopting the second strategy, which involved examining the actors and use cases that had been given based on the description.

For the second case, readability and correctness of the resulting code were significantly better than in the previous one. But there was still a small mistake, particularly with the "inclusion" related arrow, which wasn't marked with a dash. However, the user could understand it because the word "inclusion" was written above the arrow. With the improved code closely matching the analyzed description of actors and use cases, this solution was quite similar to the original UCD solution offered by the exercise.

For exercises 6 to 20, which are all regarding Use Case Diagrams, their full prompts, providing the Plant UML code for the generation of the UCD, and the description of the exercise with its respective link (for further consultation) can be found in the

following "Google Drive" link:
https://drive.google.com/drive/folders/1W3cI5jiLkLsVCyaVDQklLouh-wglntnE?usp=drive_link

---

Exercise 6.

https://www.cerritos.edu/dwhitney/SitePages/CIS201/Lectures/IM-7ed-Chapter03.pdf

The first thing to notice about ChatGPT's output in this exercise is the addition of a new actor—the "head of family"—that was mentioned implicitly in the text. The only slight difference between it and the original solution is that the PlantUML code includes one extra use case, "enter treatment information." Other than that, both use cases are actually identical and similar. It seems that the original answer's "record visit information" use case includes this additional use case.

Another noteworthy distinction is absent in the original diagram – an "include" association between the use cases "record visit info" and "enter prescription info." The text's ambiguity leaves room for interpretation regarding the necessity of this association. Consequently, the code and diagrams generated by the Language Model (LLM) in this case are once again substantially correct, complete, and even more detailed than the original, reflecting a nuanced understanding of the textual content.

Exercise 7.

https://csis.pace.edu/~marchese/CS389/L9/Use%20Case%20Diagrams.pdf

Even if this exercise is simple, it is important to examine various types of exercises with different levels of difficulty in order to properly evaluate and comprehend the results produced by Language Models (LLMs) in supporting requirements. The Use Case Diagram (UCD) produced by ChatGPT in this particular case matched the original. The "sales representative" actor wasn' included in the original diagram, but this was actually a decision made by the author rather than a constraint on how the interaction could be shown. Since the salesperson interacts with the ordering system as well, the text makes it apparent that they may be included in the use case diagram.

ChatGPT, demonstrating precision in its outcomes, generated an exact diagram, considering every possible scenario regarding actors and use cases. It appears that for simple exercises, the outcomes provided by ChatGPT are remarkably accurate and aligned with the intended representation.

Exercise 8.

https://online.visual-paradigm.com/diagrams/templates/use-case-diagram/grocery-cart-system-/

If you look at this activity quickly, it may seem more complicated than it is. It is an effective way to evaluate ChatGPT's ability to extract relevant data in order to create a Use Case Diagram (UCD). Moreover, it is clear that the created diagram and the one provided in the exercise are the same when comparing the provided PantUML code. This shows also ChatGPT's ability in generating similar results when you input an accurate prompt.

When we see that the associations between "proceed to checkout," "calculate total price," "process payment," and "send confirmation" are all "include" associations, the AI's diagram becomes very meaningful, since it is easy to see that in order to continue with the checkout process, a customer should compute the total cost, handle payment, and validate the transaction. A very obvious constraint here is the lack of a payment gateway for the "process payment" use case as a secondary or supporting actor. But even though of this, the diagram maintains its overall accuracy.

Upon introducing a new prompt inquiring about this limitation, ChatGPT addressed the issue in the above transcript. It emphasized that for a more realistic and detailed solution, the payment system support actor should be included. The subsequent PlantUML code provided by ChatGPT incorporated this refinement.

Exercise 9.

https://www.lucidchart.com/pages/UML-use-case-scenario-examples

Compared to the exercises we've previously examined, this one used a different methodology given that the information was provided as a narrative that described the full Use Case Diagram (UCD). I gave ChatGPT a revised text description of the main scenario, leaving out any information about the phases in order. The that was created was very similar to the exercise's original UCD, but with a big difference that is the inclusion of a new actor named "Consumer." As the bookshop needs the customer to finish the narrative delivery, it might be seen as a potential support actor for the use case "deliver the story to consumers," even if it is not technically necessary in the publication process.

Another big distinction, though not new one, is that ChatGPT once again struggled to differentiate the user goal level of an actor. In this case, the goal of the "agent" should be to sell the story, and the action of contacting publishing houses is a means to achieve this goal. If we think that the agent could even sell the story on its own, this step might not even be required. This could be fixed by adding a type association of "extend" between the use cases "selling the story" and "contacting publishing house."

The final code generated by ChatGPT aligns well with the original solution and is the most suitable one, presenting a close match. In this case, it can be affirmed that ChatGPT's outcome was almost entirely accurate and correct.

Exercise 10.

https://ceur-ws.org/Vol-2503/paper2_2.pdf

This exercise is very simple in terms of the quantity of actors and use cases, but it actually created a challenge for ChatGPT to achieve a good solution. A we may appreciate, the first code generated included unnecessary use cases for the actions of getting permission (for employee) and granting permission (for manager). We could think that these secondary actions are included when organizing a trip, which is actually a use case for this diagram. It's also critical to emphasize that ChatGPT integrated the booking of a hotel and a train into a single use case. Although not completely incorrect, a more comprehensive method would see them as two distinct use cases, particularly when considering the user goal level. Alternatively, they may have been combined into a single use case using a summary-level method.

Furthermore, ChatGPT demonstrated a limitation in assessing when there should be an "include" or "extend" association between use cases. When planning a trip, it is essential to have made hotel and transportation reservations in advance. Therefore, for a higher-quality Use Case Diagram (UCD), these use cases belong in the "plan a trip" use case. It took three prompts to reach this ideal UCD, indicating the need for iterative improvement and awareness of the limitations of the solution.

Exercise 11.

https://www.cpe.ku.ac.th/~plw/oop/e_book/ood_with_java_c++_and_uml/ch7.pdf

This exercise was initially solved correctly but in a simple manner by ChatGPT, encompassing the two principal use cases. However, it lacked the extended use cases for the "add asset" use case. While the exercise and the initial outcome were satisfactory, a more comprehensive result was obtained with the second code generated. This improvement happened when I requested the inclusion of extended use cases, which shows ChatGPT's capacity to improve its solutions based on user feedback.

Exercise 12.

https://www.cpe.ku.ac.th/~plw/oop/e_book/ood_with_java_c++_and_uml/ch7.pdf

In this exercise, it's important to note that for generating a report, the librarian should perform some preliminary actions such as accessing the account database and asset database. ChatGPT did not take these actions into consideration, even though they are part of the original solution. Eventhough this was somehow expected, as we know ChatGPT's limitations to deduce information that isn't stated clearly in the prompt. It did not, however, take into account the possibility of a more generalized use case for report generation.

This could account for the two different specialized cases of reports, each included as a specialization with the corresponding arrow.

Surprisingly, when I prompted ChatGPT to address this issue, it generated a code with a syntax error for the first time, failing to produce an updated Use Case Diagram (UCD). The initial UCD generated by ChatGPT is fair and correct, but it lacks some details, notably the inclusion of a generalized use case for generating a report.

Exercise 13.

https://www.uml-diagrams.org/airport-checkin-uml-use-case-diagram-example.html

In this problem, information about the actors and use cases in an Airport Check-in and security screening context was provided without detailing the sequence of actions and interactions for each actor, as in other exercises. ChatGPT was expected to comprehend and generate a reasonable solution, but the first code it produced had syntax errors, preventing the generation of the UCD. In order to obtain a code that worked correctly in the PlantUML website, I needed to prompt three times. Even though at least the code worked on the third attempt, the UCD lacked quality in terms of critical indicators such as correctness, completeness, and readability.

The lack of generalizations between actors and use cases, the inclusion of clear extend/include relationships that one would expect the AI to comprehend, and the presence of outstanding actors with no association to any use case made the diagram unhelpful. Even though it was asked ChatGPT to fix these problems and send an updated code, another code that had syntax errors was given once again.

From this, we can easily see that the AI has problems in generating good outcomes when generalizations/specializations and extend/include relationships are involved in the UCD, ChatGPT lacks the capacity to fully understand and provide a correct code.

Exercise 14.

https://www.uml-diagrams.org/ticket-vending-machine-use-case-diagram-example
html?context=uc-examples

Although a more extensive Use Case Diagram (UCD) may have been expected, this exercise just required a basic UCD solution. However, the initial code generated by ChatGPT was accurate and concise; it's only weakness was that it did not distinguish between the two actors as primary (the commuter) and secondary (the bank). In the second prompt I wrote the necessary information for this issue to be solved, and the final UCD showed the intended solution.

Exercise 15.

https://www.uml-diagrams.org/examples/hospital-management-use-case-diagram-e
html?context=uc-examples

In this exercise, unlike others we've analyzed, the relatively short text description lacked some explicit information, requiring readers to infer details for inclusion in the Use Case Diagram (UCD) solution. Some confusion was generated to the user by the fact of the incomprehensible diagram with too many associations and unnecessary actors in the first code that ChatGPT developed. The goal was to analyze the hospital reception system, with the receptionist serving as the only major actor.

Upon highlighting this issue in the second prompt, a new code was generated that resulted in a more refined UCD, comparable to the original solution, albeit with some pros and cons. On the positive side, it correctly incorporated the use cases "record payment," "provide receipt," and "file insurance claims" with "include" relationships regarding the "receive payments" use case. But something negative to show is that it wasn't abe to distinguish "admit patient" and "schedule appointment" as extended use cases of "collecting patient information." Furthermore, there was no distinction made between the patients who were inpatients and outpatients. However, the relevant "include" connection was correctly integrated with the "bed allotment" use case.

In conclusion, ChatGPT's solution satisfied the needs of the average user and might be enhanced further by a user with more UCD expertise.

Exercise 16.

https://www.uml-diagrams.org/examples/software-license-use-case-diagram-exam
html?context=uc-examples

In this exercise, a lengthy description of the problem provided numerous possibilities for drawing an incorrect Use Case Diagram (UCD). The initial output by ChatGPT had a semantic issue, potentially beneficial if one considered a code error that led to the drawing of some use cases with stickmen instead of ovals. Apart from this, the UCD showed slight but incorrect differences from the suggested solution.

For example, redundant use cases were identified for the product manager and development actors. The "Protection Key Update" process was not included as an extended use case of "Entitlement Management," and "Product Activation" was not shown as a common use case for both "Entitlement Manager" and "Customer Service." Even though that I marked this problems, the new code did not take into account "Product Activation" and the "Protection Key Update" correctly.

Therefore, taking into account the complexity of the problem description, we may conclude that the initial generated code was not at all terrible. For someone with little knowledge of UML and use cases, it identified every actor and the majority of the use cases, making it a helpful place to start. With further input and prompts, the first UCD was used as a guide to create a more accurate one.

Exercise 17.

https://online.visual-paradigm.com/diagrams/templates/use-case-diagram/
restaurant-ordering-use-case-diagram/

In this problem, ChatGPT's first output was largely accurate, with only minor errors. These were pointed out, and a second prompt was given to address them. Surprisingly, the second output included more elements than requested, introducing fundamentally incorrect relationships.

In particular, the second method includes relationships that intuitively are incorrect—namely, "include" links between a use case and an actor. The first solution highlighted the difficulties in instructing ChatGPT to deliver appropriate responses throughout iterative refining, as it was more simple and approximated the original.

Exercise 18.

https://online.visual-paradigm.com/diagrams/templates/use-case-diagram/online-ticket-booking-system-/

The last paragraph, which identifies the user as the actor with the ability to add or remove events, may have a redaction error. Nevertheless, an administrative actor that wasn't specified in the description was part of the solution. The first code from the AI only included the "user" actor and missed the administrator. This was fixed after I gave a corrected description in the second prompt, and then the AI included the administrator. This showed that when given the right information, ChatGPT can accurately identify all the actors and their use cases.

Exercise 19.

https://online.visual-paradigm.com/diagrams/templates/use-case-diagram/ride-sharing-service-/

This exercise was very complex, so the first code generated was good enough, even though with some incomplete information Two actors were identified: "user" and "administrator." The "user" actor was ambiguously used for both drivers and riders, though they only share the "cancel ride" use case. The second prompt asked ChatGPT to address this and complete the missing use cases for the driver while assigning the "manage driver complaints" use case to the administrator.

The final UCD was nearly perfect, with only a minor error in assigning the "view driver details" use case, which should have been "view rider details." These slight errors could be easily corrected by someone familiar with the problem description. Overall, ChatGPT's solution was useful and time-saving in UCD creation.

Exercise 20.

https://online.visual-paradigm.com/diagrams/templates/use-case-diagram/movie-ticket-reservation-system-/

The description of this exercise concentrated on the perspective of the customer and how they interacted with the movie ticket reservation system.

One thing I want to highlight is that the administrator actor wasn't explicitly mentioned in the text, and even though, it was included with its relevant use cases in the original solution.This time, ChatGPT could identify the customer actor and all their use cases, but it had trouble with the administrator. Also, even though ChatGPT included some "include" relationships, it didn't use the right dashed arrows and "«include»" notation to show them properly. Despite these small issues, the quality of ChatGPT's output was impressive and showed its ability to create detailed diagrams even with slight deviations from the given instructions.

After this small error, ChatGPT generated output of very good quality, proving its capacity to produce precise and complete diagrams even when there are small variations from the guidelines.

3.7.2 Use Case Narrative Exercises

The ten exercises regarding UC Narratives are present in this subsection. First, five exercises which are common to some of the UCD exercise descriptions exposed in the previous subsection will be deeply analyzed here, with the full prompts and solutions. For the prompts, the reader should recall at section 3.6 to see the generic prompt presented for all exercises of narratives. Then, the parts that are going to be shown here are the specialized description of each exercise, and the specialized name of each use case under analysis. For the remaining five exercises, the comparative analysis is shown, but the exercises complete prompts and solutions can be consulted on the same "Google Drive" link where the UCD exercises are: https://docs.google.com/document/d/1mWaGZ-f0ObCoTqmtXWYVkgUd3T-tIzyr/edit?usp=drive_link&ouid=105778861586687512114&rtpof=true&sd=true

Exercise 1. **(Related to *Exercise 2.* from the UCD subsection.)**

https://www.chegg.com/homework-help/systems-analysis-and-design-in-a-changin

https://mygust.com/uploads/BOOK-Systems_analysis_and_design_in_a_changin.pdf

PROMPT:

Description: "One of the functions of the Board of Realtors is to provide a Multiple Listing Service (MLS) system that supplies information that local real estate agents use to help them sell houses to their customers. During the month, agents list houses for sale (listings) by contracting with homeowners. The agent works for a real estate office, which sends information on the *listing to the MLS. Therefore, any agent in the community can get information on the listing. Information on a listing includes the address, year built, square feet, number of bedrooms, number of bathrooms, owner name, owner phone number, asking price, and status code. At any time during the month, an agent might directly request information on listings that match customer requirements, so the agent contacts the MLS with the request. Information is provided on the house, on the agent who listed the house, and on the real estate office for which the agent works. For example, an agent might want to call the listing agent to ask additional questions or call the homeowner directly to make an appointment to show the house. Once each week, the MLS produces a listing book that contains information on all listings. These books are sent to some real estate agents. Some real estate agents want the books (which are easier to flip through), so they are provided even though the information is often out of date. Sometimes, agents and owners decide to change information about a listing, such as reducing the price, correcting previous information on the house, or indicating that the house is sold. The real estate office sends in these change requests to the MLS when the agent asks the office to do so."*

Use Case: "Add agent to real estate office".

ChatGPT

Use Case Name: Add Agent to Real Estate Office

Triggering Event: Real estate office decides to add a new agent to their team.

Brief Description: This use case involves the process of adding a new agent to a real estate office within the context of the Board of Realtors system.

Actor: Real Estate Office Manager/Administrator

Related Use Case: None directly related.

Stakeholders:

1. Real Estate Office Manager/Administrator: Initiates the process to add a new agent.

2. Board of Realtors System: Manages and records information related to real estate agents and listings.

Precondition: The real estate office is registered with the Board of Realtors system, and the system is operational.

Postcondition: The new agent is successfully added to the real estate office, and their information is updated in the Board of Realtors system.

Use Case Name:	Add agent to real estate office.	
Scenario:	Adding an agent by the clerk.	
Triggering event:	Hiring a new agent.	
Brief description:	A detail about the agent is entered into the system by finding the exact office location of the agent.	
Actors:	Clerk	
Related use case:	Add new agent (through web)	
Stakeholders:	Office of the real estate Agent of the real estate	
Precondition:	Existence of real estate office	
Postcondition:	Creating a real estate agent and linked with the real estate office	
Flow of activities:	**Actor**	**System**
	1. Identifying real estate office 2. Agent details are entered by the clerk	1.1 System displays the real estate office details 2.1 System adds the agent details and linked the exact real estate office
Exception conditions:	1.1 If the real estate office is not found, display it is not found message	

Figura 3.14. Original Narrative provided by the exercise.

Flow of Activities:

Actor: Real estate office manager logs into the Board of Realtors system.

System Response: System validates the manager's credentials and grants access.

Actor: Manager selects the option to add a new agent to the office.

System Response: System prompts the manager to enter the new agent's details (name, contact information, etc.).

Actor: Manager enters the required information and confirms the addition.

System Response: System validates the data and updates the real estate office's records.

Actor: System notifies the manager of the successful addition of the new agent.

System Response: Manager receives a confirmation message and can view the updated list of agents.

Original Solution shown in Fig.3.14:

When comparing the two solutions for the "Add Agent to Real Estate Office" use case, both have common elements like the triggering event, actors, stakeholders, preconditions, post-conditions, and the core flow of activities. But there are some key differences in how they approach things.

The original solution, which is longer,can be observed to have a considerably deeper description that places a lot of attention on the clerk's part in entering agent details and determining the precise office location. Additionally, the modified version offers an alternate procedure for adding a new agent via the web, whereas the original makes no mention of a corresponding use case.

The original provides a deep perspective of the clerk's actions and the system's reaction by including more specific steps in the flow of activities. This is made simpler in the new version, which has fewer stages for data entry and identification.

It can be said that in general, the original solution is better at providing detailed steps and a nuanced description, while the revised one is more concise and adds the idea of a related use case. Combining the strengths of both could result in a comprehensive and concise use case description.

***Exercise 2.* (Related to *Exercise 2.* from the UCD subsection.)**

https://www.chegg.com/homework-help/systems-analysis-and-design-in-a-changin

https://mygust.com/uploads/BOOK-Systems_analysis_and_design_in_a_changin.pdf

PROMPT: The prompt for this exercise is the same one as the one of *Exercise 1.*, with the only difference that the Use case asked to be analyzed is now "Create New Listing", instead of "Add Agent to Real State Office".

ChatGPT

Use Case Name: Create New Listing

Scenario: Creating a new listing for a property.

Triggering Event: A real estate agent initiates the process of listing a property for sale.

Brief Description: This use case involves the process of adding a new property listing to the Multiple Listing Service (MLS) system, enabling real estate agents to access and share information about the property with potential buyers.

Actor: Real Estate Agent

Related Use Case: None directly related.

Stakeholders:

1. Real Estate Agent: Initiates the process of creating a new listing.
2. Homeowner: Provides information about the property to be listed.
3. Multiple Listing Service (MLS) System: Records and disseminates information about property listings.

Precondition: The real estate agent is registered with the MLS system, and the system is operational.

Postcondition: The new property listing is successfully added to the MLS system and is accessible to other real estate agents.

Flow of Activities:

Actor: Real estate agent logs into the MLS system.

System Response: System validates the agent's credentials and grants access.

Actor: Agent selects the option to create a new listing.

System Response: System prompts the agent to enter details about the property (e.g., address, year built, square footage, bedrooms, bathrooms, owner's contact information, asking price).

Actor: Agent enters the required information and submits the listing.

System Response: 1) System validates the data and adds the new listing to the MLS database. 2) System confirms successful creation of the listing.

Actor: Agent receives a confirmation message and can view the newly created listing in the MLS system.

Exception Conditions:

- If the agent's credentials are invalid, the system denies access and prompts the agent to provide correct credentials.
- If there are technical issues preventing the creation of the listing, the system displays an error message and instructs the agent to try again later.
- If the entered information for the new listing is incomplete or incorrect, the system prompts the agent to correct the details before submission.
- In case the MLS system is unavailable, the agent is informed of the temporary outage and advised to try again when the system is back online.

Original Solution shown in Fig.3.15

Use Case Name:	Create new listing	
Scenario:	Creating new list by the clerk	
Triggering event:	Getting a new property to sale	
Brief description:	Identify and verify the agent who created the listing. Then add the details of the property to the system	
Actors:	Clerk	
Related use case:	Agent/Office creates the listing (through web).	
Stakeholders:	Office of the real estate Agent of the real estate Owner of the property	
Preconditions:	Existence of real estate office. Existence of real estate agent	
Postconditions:	Creating a listing and linked with the real estate office and real estate agent	
Flow of activities:	**Actor**	**System**
	1. Identifying real estate agent 2. Listing details are entered by the clerk	1.1 System displays the real estate agent and corresponding office details 2.1 System creates listing and links the exact real estate agent and then displays the listing details
Exception conditions:	1.1 If the real estate agent details are not found, display the message “Not found”.	

Figura 3.15. Original Narrative provided by the exercise.

For this exercise, when we take a look at the revised approach ChatGPT came up with and the solution offered for the "Create New Listing" use case both focus on the clerk’s role in listing a new property. Basics like the trigger event of receiving a new property for sale, the clerk’s role as the principal player, and the requirement to identify and validate the agent in charge of the listing are all covered by both solutions. They also recognize the importance of difficult factors like the existence of the real estate office and agent, as well as stakeholders like the agent, property owner, and office.

Maybe the biggest difference between them is in detail and structure. The provided solution keeps things short and concise, just outlining how the clerk finds the agent and fills in the listing details. On the other hand, the generated code by the AI goes deeper, breaking down the clerk’s actions and the system’s responses. ChatGPT also added a related use case where agents can create listings online, giving them another option.

Exercise 3. **(Related to *Exercise 3.* from the UCD subsection)**

https://www.chegg.com/homework-help/systems-analysis-and-design-in-a-changin
https://mygust.com/uploads/BOOK-Systems_analysis_and_design_in_a_changin.pdf

PROMPT:

Description: "As On the Spot Courier Services continues to grow, Bill discovers that he can provide much better services to his customers if he utilizes some of the technology that is currently available. For example, it will allow him to maintain frequent communication with his delivery trucks, which could save transportation and labor costs by making the pickup and delivery operations more efficient. This would allow him to serve his customers better. Of course, a more sophisticated system will be needed, but Bill's development consultant has assured him that a straightforward and not too complex solution can be developed. Here is how Bill wants his business to operate. Each truck will have a morning and an afternoon delivery and pickup run. Each driver will have a portable digital device with a touch screen. The driver will be able to view his or her scheduled pickups and deliveries for that run. (Note: This process will require a new use case—something the Agile development methodology predicted would happen.) However, because the trucks will maintain frequent contact with the home office via telephony Internet access, the pickup/delivery schedule can be updated in real-time—even during a run. Rather than maintain constant contact, Bill decides that it will be sufficient if the digital device synchronizes with the home office whenever a pickup or delivery is made. At those points in time, the route schedule can be updated with appropriate information. Previously, customers were able to either call On the Spot and request a package pickup or visit the company's website to schedule a pickup. Once customers logged in, they could go to a web-page that allowed them to enter information about each package, including "deliver to" addresses, size and weight category information, and type of service requested. On the Spot provided "three-hour," "same day," and "overnight" services. To facilitate customer self-service, On the Spot didn't require exact weights and sizes, but there were predefined size and weight categories from which the customer could choose. Once the customer entered the information for all the packages, the system would calculate the cost and then print mailing labels and receipts. Depending on the type of service requested and the proximity of a delivery truck, the system would schedule an immediate pickup or one for later that day. It would display this information so the customer would immediately know when to expect the pickup. Picking up packages was *a fairly straightforward process. But there was some variation in what would happen depending on what information was in the system and whether the packages were already labeled. Upon arriving at the scheduled pickup location, the driver would have the system display any package information available for this customer. If the system already had information on the packages, the driver would simply verify that the correct information was already in the system for the packages. The driver could also make such changes as correcting the address, deleting packages, or adding new packages. If this were a cash customer, the driver would collect any money and enter that into the system. Using a portable printer from the van, the driver could print a receipt for the customer as necessary. If there were new packages that weren't in the system, the driver would enter the required information and also print mailing labels with his portable printer. One other service that customers required was to be able to track the delivery status of their packages. The system needed to track the status of a package from the first time it "knew" about the package until it was delivered. Such statuses as "ready for pickup," "picked up," "arrived at warehouse," "out for delivery," and "delivered" were important. Usually, a package would follow through all the statuses, but due to the sophistication of the scheduling and delivery algorithm, a package would sometimes be picked up and delivered on the same delivery run. Bill also decided to add a status of "cancelled" for those packages that were scheduled to be picked up but ended up not being sent."*

Use Case: "request a package pick-up".

ChatGPT

Use Case Name: Request Package Pick-Up

Scenario: Initiating a package pick-up request by a customer.

Triggering Event: A customer initiates a request for On the Spot Courier Services to pick up a package.

Brief Description: This use case involves customers requesting the pick-up of packages through On the Spot Courier Services. Customers can log in to the system via the company's website, provide package details, select service options, and schedule a pick-up.

Actor: Customer

Related Use Case: None directly related.

Stakeholders:

1. Customer: Initiates the request for package pick-up.

2. On the Spot Courier Services: Manages and fulfills pick-up requests.

Precondition: The customer is registered on the On the Spot Courier Services website, and the system is operational.

Post-condition: The system acknowledges the pick-up request, schedules it based on the selected service, and updates the customer on the expected pick-up time.

Flow of Activities:

Actor: Customer logs in to the company's website.

System Response: System validates customer credentials and grants access.

Actor: Customer navigates to the pick-up request page.

System Response: System displays a web-page allowing the customer to enter package details (size, weight, "deliver to" address) and select service options ("three-hour," "same day," or "overnight").

Actor: Customer submits package information and service preferences.

System Response: System calculates the cost, schedules the pick-up based on service type and proximity of delivery truck, and displays the scheduled pick-up time to the customer.

Actor: Customer reviews the scheduled pick-up details and confirms the request.

System Response: System confirms the pick-up request, generates and prints mailing labels, and updates the package status to "ready for pickup."

Exception Conditions:

- If the customer's login credentials are invalid, the system denies access and prompts the customer to provide correct credentials.
- If there are technical issues preventing the submission of package details, the system displays an error message and instructs the customer to try again later.
- If the entered information for the pick-up request is incomplete or incorrect, the system prompts the customer to correct the details before submission.
- In case the system cannot schedule an immediate pick-up due to operational constraints, it informs the customer of the next available slot and confirms the scheduled pick-up time.

Original Solution shown in Fig.3.16

Use Case Name:	Request a package pickup	
Scenario:	Requesting a package pickup through online.	
Triggering event:	Customer request for a package pickup through online	
Brief description:	Customer wants the courier service to pick up the package by requesting through online. The package details such as delivery address, service type, size and weight of the package. The system calculates the cost, expected time of pickup and prints the delivery label.	
Actors:	Customer	
Related use case:	Customer request for a package pickup through phone Customer request for a package pickup in-person	
Stakeholders:	Customer Employee Bill	
Preconditions:	Account for customer should exit	
Post conditions:	Creating a pickup request and link with the customer account	
Flow of activities:	**Actor**	**System**
	1. For every package 1.1 The delivery address is entered by the customer 1.2 The service type is entered by the customer 1.3 The size and weight of the package is entered by the customer 1.4 Customer requests the delivery label	1.3.1System calculates the cost 1.3.2 System shows the expected time of pickup 1.3.3 System creates the record for request and package 1.4.1System prints the delivery label
Exception conditions:	1.4. Failure of printing the label	

Figura 3.16. Original Narrative provided by the exercise.

The solutions are very similar regarding all the elements included in them and the structure . In order to request a pick-up online, clients must follow the instructions provided by both systems, which include inputting package details, calculating charges, displaying displayed pickup times, and generating delivery labels.

Probably the major difference that we may see from both solutions could be related to the level of detail . The original solution looks more on what the customer does and how the system responds, keeping it pretty straightforward. But the ChatGPT-generated solution breaks down the process into smaller parts, giving a more thorough explanation. What makes the provided solution even better is that it includes related use cases for requesting package pickups by phone or in-person. This adds extra value to the description. Conversely, the narrative produced by AI maintains clarity by excluding any references to related use cases.

Even with some variations mostly in the level of detail, but we can see that both solutions are able to show a quite good narrative.

Exercise 4. (Related to *Exercise 4.* from the UCD subsection)

https://cmps-people.ok.ubc.ca/bowenhui/310/8-UML.pdf

PROMPT:

Description: "This case study concerns a simplified system of the automatic teller machine (ATM). The ATM offers the following services: 1. Distribution of money to every holder of a smartcard via a card reader and a cash dispenser. 2. Consultation of account balance, cash and cheque deposit facilities for bank customers who hold a smartcard from their bank. Do not forget either that: 3. All transactions are made secure. 4. It is sometimes necessary to refill the dispenser, etc. Let's look at each of the sentences of the exposition in turn. Sentence 1 allows us to identify an obvious initial actor straight away: every "holder of a smartcard". He or she will be able to use the ATM to withdraw money using his or her smartcard. However, be careful: the card reader and cash dispenser constitute part of the ATM. They can therefore not be considered as actors! You can note down that the identification of actors requires the boundary between the system being studied and its environment to be set out exactly. If we restrict the study to the control/command system of physical elements of the ATM, the card reader and cash dispenser then become actors. Another trap: is the smartcard itself an actor? The card is certainly external to the ATM, and it interacts with it... Yet, we do not recommend that you list it as an actor, as we are putting into practice the following principle: eliminate "physical" actors as much as possible to the advantage of "logical" actors. The actor is the who or what that benefits from using the system. It is the card holder who withdraws money to spend it, not the card itself! Sentence 2 identifies additional services that are only offered to bank customers who hold a smartcard from this bank. This is therefore a different profile from the previous one, which we will realise by a second actor called Bank customer. Sentence 3 encourages us to take into account the fact that all transactions are made secure. But who makes them secure? There are therefore other external entities, which play the role of authorisation system and with which the ATM communicates directly. An interview with the *domain expert4 is necessary to allow us to identify two different actors: • the Visa authorisation system (VISA AS) for withdrawal transactions carried out using a Visa smartcard (we restrict the ATM to Visa smartcards for reasons of simplification); • the information system of the bank (Bank IS) to authorise all transactions carried out by a customer using his or her bank smartcard, but also to access the account balance. Bank customer and CardHolder are, furthermore, mutually exclusive. Another solution, which is a little more developed, consists in considering Bank customer as a specialisation of CardHolder. Let's take the cardholder and list the different ways in which they can use the ATM: CardHolder: one of this ways is that it can withdraw money. The Withdraw money use case has two possible primary actors (but they cannot be simultaneous). Another way to express this notion is to consider the Bank customer actor as a specialisation (in the sense of the inheritance relationship) of the more general CardHolder actor. A bank customer is actually a particular card holder who has all the privileges of the latter, as well as others that are specific to him or her as a customer. For all use cases appropriate for the bank customer, you must explicitly bring in Bank IS as a secondary actor. But a problem arises for the shared use case, Withdraw money. Indeed, if the primary actor is a Visa card holder, the Visa AS must be called on (which will then be responsible for contacting the IS of the holder's bank); whereas the ATM will contact the Bank IS directly if it concerns a bank customer. Another solution would be to distinguish two use cases for the withdrawal of money: Withdraw money using a Visa card and Withdraw money using a bank card. This more precise, yet more cumbersome, modelling is easier for the reader of the diagram to grasp. Furthermore, it clearly tells against the use of generalisation between actors, which was mentioned beforehand. Indeed, the distinction between the two use cases is contradictory with the attempt at inheritance of the unique Withdraw money case, which had been viewed more highly, while the secondary actors had not yet been added."*

Use Case: "Withdraw money using a Visa Card".

ChatGPT

Use Case Name: Withdraw Money Using a Visa Card

Scenario: A Visa cardholder initiates a request to withdraw money from the ATM.

Triggering Event: The Visa cardholder inserts their Visa smartcard into the card reader.

Brief Description: The use case involves the Visa cardholder interacting with the ATM to withdraw money securely using their Visa smartcard. The system verifies the cardholder's identity, authorizes the transaction through the Visa Authorization System (VISA AS), and dispenses the requested amount.

Actor:

- Primary Actor: Visa Cardholder
- Secondary Actor: VISA AS

Related Use Case: Withdraw Money Using a Bank Card
Stakeholders:

- Visa Cardholder: Initiates the withdrawal transaction.
- VISA AS: Authorizes the transaction and communicates with the bank's Information System (Bank IS) for approval.

Precondition:

- The Visa cardholder has a valid Visa smartcard.
- The ATM is operational.

Postcondition:

- The requested amount is dispensed to the Visa cardholder.
- Transaction details are recorded.

Flow of Activities:

1. Visa Cardholder inserts the Visa smartcard into the card reader. 1.1 System prompts the cardholder to enter their PIN. 1.2 Visa Cardholder enters the PIN. 1.3 System validates the PIN.
2. System contacts VISA AS for authorization. 2.1 VISA AS verifies the cardholder's authorization.
3. Visa Cardholder specifies the withdrawal amount. 3.1 System validates the requested amount.
4. System dispenses the requested amount. 4.1 Visa Cardholder collects the dispensed cash.

Exception Conditions:

- If the PIN is incorrect, display an error message and allow the cardholder to retry.
- If the VISA AS denies authorization, display a transaction declined message.
- If the requested amount exceeds limits, display an error message.
- If the smartcard is invalid or damaged, prompt the cardholder to contact their bank.

Original Solution available at: https://cmps-people.ok.ubc.ca/bowenhui/310/8-UML.pdf, pg. 15-19.

The original use case is very comprehensive and structured and It clearly covers the main success scenario, alternative sequences, and error sequences. It is simple to follow the processes of the interaction between the Visa cardholder, the ATM, and the VISA authorization system because of the orderly structure of the events. Something very interesting is the inclusion of time in the error steps, like E4 and E5, where the ATM does some actions just aftes some specific period of time. By addressing probable problems like the card not being returned or the cash not being collected, this gives the situation a more realistic feeling.

Moreover, the use of clear and concise language in both the main success scenario and error sequences contributes to the overall readability and understanding of the use case. The detailed postconditions give a clear representation of the expected situation following the use case's conclusion. Since these errors may involve actions or communication with the Visa authorization system, one consideration for potential improvements or revisions would be to explicitly mention that the Visa AS was involved in the error sequences (E1, E2, E3, E4, E5) where appropriate. When everything is considered, the first use case is prepared effectively, providing detailed coverage of a variety of scenarios and using realistic features to enhance understanding of the system's behavior in different settings.

Exercise 5. (Related to _Exercise 5._ from the UCD subsection)

https://web.uettaxila.edu.pk/CMS/AUT2010/seSREbs/notes%5CLecture_12%20SRE.pdf

PROMPT:

Description: " A bank has several automated teller machines (ATMs), which are geographically distributed and connected via a wide area network to a central server. Each ATM machine has a card reader, cash dispenser, a keyboard/display, and a receipt printer. By using the ATM machine, a customer can withdraw cash from either checking or savings account, query the balance of an account, or transfer funds from one account to another. A transaction is initiated when a customer inserts an ATM card into the card reader. Encoded on the magnetic strip on the back of the ATM card are the card number, the start date, and the expiration date. Assuming the card is recognized, the system validates the ATM card to determine: that the expiration date has not passed, that the user-entered PIN matches the PIN maintained by the system, that the card is not lost or stolen. The customer is allowed three attempts to enter the correct PIN; the card is confiscated if the third attempt fails. Cards that have been reported lost or stolen are also confiscated. If the PIN is validated satisfactorily, the customer is prompted for a withdrawal, query, or transfer transaction. Before a withdrawal transaction can be approved, the system determines that sufficient funds exist in the requested account, that the maximum daily limit will not be exceeded, and that there are sufficient funds available at the local cash dispenser. If the transaction is approved, the requested amount of cash is dispensed, a receipt is printed containing information about the transaction, and the card is ejected. Before a transfer transaction can be approved, the system determines: that the customer has at least two accounts and that there are sufficient funds in the account to be debited. For approved query and transfer requests, a receipt is printed, and the card ejected. A customer may cancel a transaction at any time; the transaction is terminated, and the card is ejected. Customer records, account records, and debit card records are all maintained at the server. An ATM operator may start up and close down the ATM to replenish the ATM cash dispenser and for routine maintenance. It is assumed that functionality to open and close accounts and to create, update, and delete customer and debit card records is provided by an existing system and is not part of this problem. The customer is able to: Withdraws funds from the checking or savings account, Query the balance of an account, Transfer funds from one account to another. Consider that the action of PIN validation is common to all these 3 actions. The ATM customer interacts with the system via the ATM card reader, keyboard/display, cash dispenser, and receipt printer. "

Use Case: "Validate PIN".

ChatGPT

Use Case Name: Validate PIN

Scenario: A customer initiates the PIN validation process by inserting an ATM card into the card reader.

Triggering Event: Customer inserts the ATM card into the card reader.

Brief Description: The use case involves validating the customer's Personal Identification Number (PIN) when initiating a transaction. The system checks the validity of the ATM card, ensuring that the expiration date has not passed, the entered PIN matches the system's records, and the card is not reported as lost or stolen. The customer is given three attempts to enter the correct PIN; failure on the third attempt results in the confiscation of the card.

Actor: ATM Customer

Related Use Cases:

- Withdraw Funds
- Query Balance
- Transfer Funds

Stakeholders:

- ATM Customer
- Bank (System)

Precondition:

- The ATM card is inserted into the card reader.
- The card is recognized by the system.

Postcondition:

- Successful validation allows the customer to proceed with the desired transaction.
- If unsuccessful after three attempts, the card is confiscated.

Flow of Activities:
Actor:
1. Insert ATM card
System Response:
2. Read card information
3. Validate expiration date

4. Validate PIN (3 attempts allowed)
5. If PIN validation successful, proceed
6. If PIN validation fails after 3 attempts, confiscate card
7. If card reported lost or stolen, confiscate card
Actor:
8. Enter PIN
System Response:
9. Verify entered PIN against system records
Actor:
10. Verify PIN
11. Proceed with transaction or confiscate card if unsuccessful
System Response:
12. Eject card (after successful transaction or confiscation)
Exception Conditions:

- If the entered PIN is incorrect, inform the customer and allow additional attempts.
- If the card is reported lost or stolen, confiscate the card.
- If the expiration date has passed, inform the customer and do not proceed with the transaction.
- If the card is not recognized, inform the customer.

Original Solution shown in Fig.3.17

The solution provided by the exercise is very concise but at the same time describes very well the PIN validation transaction by showinf the essential steps, including card recognition, PIN entry, and potential alternative scenarios like card expiration or being reported lost or stolen.

Regarding ChatGPT's solution, it offered a solution much more comprehensive by including certain aspects like a table that had the flow of all the actionsor even some details like a triggering event, related use cases, stakeholders, and a more explicit breakdown of the flow of activities.

In this exercise, even though both solution are fine, one can say that ChatGPT´s response was able to provide a bit more detail and structure, most of all through the use of a table and additional sections for related information. The differences are largely in the presentation and level of detail rather than a substantial deviation from the core content of the original solution.

Validate PIN Use Case - 1

- Name: Validate PIN
- Summary : System validates customer PIN
- Dependency: none
- Actors: ATM Customer
- Preconditions: ATM is idle, displaying a Welcome message.
- Flow of Events: Basic Path
  1. Customer inserts the ATM card into the Card Reader
  2. If the system recognizes the card, it reads the card number
  3. System prompt customer for PIN number
  4. Customer enters PIN
  5. System checks the expiration date and whether the card is lost or stolen
  6. If card is valid, the system then checks whether the user entered PIN matches the card PIN maintained by the system
  7. If PIN numbers match, the system checks what accounts are accessible with the ATM card
  8. System displays customer accounts and prompts customer for transaction type: Withdrawal, Query, or Transfer
- Alternatives:
  - ❑ If the system does not recognize the card, the card is ejected
  - ❑ If the system determines that the card date has expired, the card is confiscated
  - ❑ If the system determines that the card has been reported lost or stolen, the card is confiscated
  - ❑ If the customer-entered PIN does not match the PIN number for this card, the system re-prompts for PIN
  - ❑ If the customer enter the incorrect PIN three times, the system confiscates the card
  - ❑ If the customer enters Cancel, the system cancels the transaction and ejects the card
- Post condition: Customer PIN has been validated

Figura 3.17. Original Narrative provided by the exercise.

Exercise 6.**(Related to *****Exercise 12.***** from the UCD subsection**

https://web.uettaxila.edu.pk/CMS/AUT2010/seSREbs/notes%5CLecture_

12%20SRE.pdf

The original solution was able to showcase a very good narrative in terms of completeness. It may be stated that the "Validate PIN" use case shows an inclusion dependency, which highlights the significance of PIN validation prior to the withdrawal process, is one important change. Something interesting was the addition of alternatives, such as managing invalid account numbers or insufficient cash, strengthens the system and completely covers a range of eventualities. The postcondition clearly explains the intended outcome by briefly stating that the withdrawal was completed successfully. Even though both solutions are okay, in this case, the original one showcased a better representation of the "Withdraw Funds" use case due to its explicit inclusion of the "Validate PIN" requirement and its careful handling of alternate circumstances.

Exercise 7.

https://www.chegg.com/homework-help/systems-analysis-and-design-in-a-changin

https://mygust.com/uploads/BOOK-Systems_analysis_and_design_in_a_changin.pdf

The initial approach to this use case demonstrates a very well-structured, simple approach while also managing to incorporate all required process steps. The related use case "Add new driver," which guarantees a flexible and versatile method of managing new drivers inside the current policy, is one notable feature. We can see that the inclusion of some preconditions and also postconditions was able to ensure a good understanding of the state of the system both before and after the completion of the use case. To improve the activity flow, it could be helpful to show a simpler transition from the clerk's actions to the system's reactions. In this case and once again, ChatGPT was able to offer a better solution if we think in terms of a detailed flow of activities, since its solution showed directly the actor's steps and the corresponding system responses in separate columns, which generated a better way of viewing the interaction between the actor and the system, and at the same time made it simpler to understand. Although the two solutions show in a good way the use case, with the original solution offering a more streamlined and concise presentation, the solution generated by ChatGPT provides a more detailed breakdown of actor-system interactions.

Exercise 8.

https://www.chegg.com/homework-help/systems-analysis-and-design-in-a-changin https://mygust.com/uploads/BOOK-Systems_analysis_and_design_in_a_changin.pdf

Once again, the original narrative is simple but if we talk in terms of completeness and good quality, it has nothing missing at all. It is able to show all the necessary details such as resort information, facilities, and accommodations. Having said that, it is nevertheless crucial to note that things could become better. The actor referred to as "Clerk" is first identified; however, it would be helpful to clarify if this is a reference to a specific role in the resort management or SBRU system administration. We are also able to see that the flow of activities could be more detailed. Related to the related use cases and stakeholders, these were two aspect that were covered by the solution provided by the AI, offering a more comprehensive knowledge of the system's interactions. Additionally, the steps for validating the entered data were specified explicitly. The generated solution in this case then only differs from the original one in the sense that it offers more details.

Exercise 9.

https://www.usability.gov/how-to-and-tools/methods/use-cases.html#:~:text=A%20use%20case%20is%20a,when%20that%20goal%20is%20fulfilled.

There are some things that are alike between both solutions related to the main activities of doing laundry, like for example sorting, washing, drying, folding, and ironing. ChatGPT´s method, grouped related tasks under a generic flow of events and arranges alternate flows for particular cases, providing a more succinct and ordered presentation. ChatGPT´s version also addressed the importance of using a stakeholder named "Housekeeper" in order to better conform to accepted use case modeling practices. A complete understanding can be gained from the original solution's detailed analysis of alternate flows and exceptions, but a high-level use case description may not require such granularity. The new provided approach aimed for clarity while keeping essential information for clear understanding and communication.

Exercise 10.

https://www.chegg.com/homework-help/systems-analysis-and-design-in-a-changin

https://mygust.com/uploads/BOOK-Systems_analysis_and_design_in_a_changin.pdf

Between the structure of both narratives we may find just few differences even though that the solution from the AI showcases a better description of the process, by dividing in a better way the actions between actor and system. The updated solution also incorporates particular system interactions, such prompts, automatic updates, and validations, giving a more thorough understanding of the system's function in the use case. By taking into account probable unusual events, the provision of alternative scenarios, which specifically address credit limit concerns and missing item prices, improves the use case's robustness. Overall, the method is still explained in a clear and comprehensive manner in the updated solution, which makes it more detailed and informative.

Capitolo 4

Results

4.1 Analysis of Use Case Diagram Exercises

4.1.1 Actors Identification

Regarding the identification of actors, an Excel table was created to analyze the accuracy of actor identification in the diagrams generated by ChatGPT. The table includes the following columns:

- Exercise: Lists the number of each exercise (total of 20 exercises).
- Number of Necessary Actors: Indicates the number of necessary actors involved in each exercise.
- Number of Correctly Identified Actors: Shows the count of correctly identified actors in the diagrams generated by ChatGPT.
- % of Correctly Identified Actors: Presents the percentage of correctly identified actors out of the total necessary actors for each exercise.
- Number of Wrongly Identified Actors: Displays the count of wrongly identified actors in the diagrams.

It's important to note that the number of correctly identified actors and the number of wrongly identified actors are not directly correlated. For instance, an exercise may require 2 necessary actors, but ChatGPT may identify extra actors incorrectly.

The analysis reveals that, on average, each exercise contains approximately 2.85 necessary actors. ChatGPT identified an average of 2.55 correct actors per exercise, indicating a nearly 90% accuracy rate in actor identification across all exercises. On average, there were 0.3 wrongly identified actors per exercise, totaling to 20 exercises. Overall, these findings suggest that ChatGPT's capabilities in identifying actors are relatively satisfactory.

ACTORS				
Exercis	Number of necessary acto	Number of actors correctly identifi	% of correctly identified actors	Number of wrong identified actors (Also count extra actors wrongly identified
1	4	4	100%	0
2	3	2	67%	0
3	4	3	75%	1
4	5	5	100%	0
5	2	2	100%	0
6	3	3	100%	1
7	1	1	100%	0
8	2	1	50%	0
9	6	6	100%	1
10	3	3	100%	0
11	1	1	100%	0
12	1	1	100%	0
13	4	4	100%	0
14	2	2	100%	0
15	1	1	100%	3
16	5	5	100%	0
17	3	2	67%	0
18	2	2	100%	0
19	3	2	67%	0
20	2	1	50%	0
AVERAGES	2.85	2.55	0.8875	0.3

Figura 4.1. Actors Identification.

4.1.2 Use Cases Identification

Regarding the identification of Use Cases, an Excel table was constructed to evaluate the accuracy of use case identification in the diagrams generated by ChatGPT. The table includes the following columns:

- Exercise: Lists the number of each exercise (total of 20 exercises).
- Number of Necessary Use Cases: Indicates the number of necessary use cases involved in each exercise.
- Number of Correctly Identified Use Cases: Displays the count of correctly identified use cases in the diagrams generated by ChatGPT.
- % of Correctly Identified Use Cases: Presents the percentage of correctly identified use cases out of the total necessary use cases for each exercise.

- Number of Wrongly Identified Use Cases: Specifies the count of wrongly identified use cases in the diagrams.

Similar to the identification of actors, the number of correctly identified use cases and the number of wrongly identified use cases are not directly correlated due to potential inclusion of extra or missing use cases by ChatGPT.

The analysis indicates that, on average, each exercise contains approximately 6.7 necessary use cases. ChatGPT identified an average of 5.4 correct use cases per exercise, representing an approximate 85% accuracy rate in use case identification across all exercises. On average, there were 0.75 wrongly identified use cases per exercise, totaling to 20 exercises. Overall, while ChatGPT's capabilities in identifying use cases are relatively satisfactory, they were slightly less accurate compared to the identification of actors.

USE CASES				
Exercis	Number of necessary UC	Number of UC correctly identifie	% of correctly identified UC	Number of wrong identified UC (also counts extra UC wrongly identified)
1	7	7	100%	0
2	11	7	64%	0
3	8	6	75%	5
4	7	6	86%	1
5	7	7	100%	0
6	6	6	100%	1
7	5	5	100%	0
8	7	7	100%	0
9	7	6	86%	1
10	3	3	100%	2
11	7	7	100%	0
12	2	2	100%	0
13	7	5	71%	0
14	1	1	100%	0
15	9	5	56%	2
16	9	6	67%	2
17	4	3	75%	1
18	7	7	100%	0
19	8	6	75%	0
20	12	6	50%	0
AVERAGES	6.7	5.4	0.851857864	0.75

Figura 4.2. Use Case Identification.

4.1.3 Associations Identification

To assess the identification of associations in the generated use case diagrams, an Excel table was created with the following columns:

- Exercise: Lists the number of each exercise (total of 20 exercises).
- Number of Necessary "Extend" Associations: Indicates the number of necessary "extend" type associations involved in each exercise.
- Number of Necessary "Include" Associations: Specifies the number of necessary "include" type associations involved in each exercise.
- % of Missing "Extend" Associations: Presents the percentage of missing identified "extend" associations by ChatGPT.
- % of Missing "Include" Associations: Displays the percentage of missing identified "include" associations by ChatGPT.

ASSOCIATIONS				
Exercis	Number of necessary ext a	Number of necessary inc ass	% of missing "extend" associatio	% of missing "include" associations
1	0	0	0%	0%
2	0	0	0%	0%
3	0	0	0%	0%
4	0	0	0%	0%
5	0	1	0%	0%
6	1	1	100%	0%
7	0	0	0%	0%
8	0	0	0%	0%
9	0	0	0%	0%
10	0	2	0%	100%
11	0	0	0%	0%
12	0	0	0%	0%
13	3	1	100%	100%
14	0	0	0%	0%
15	2	2	100%	50%
16	1	0	100%	0%
17	0	1	0%	100%
18	0	0	0%	0%
19	0	0	0%	0%
20	0	0	0%	0%
AVERAGES	**0.35**	**0.4**	**100%**	**58%**

Figura 4.3. Associations Identification.

The analysis of associations reveals that, on average, each exercise contains 0.35 necessary "extend" associations and 0.4 necessary "include" associations. These averages are notably low, indicating that the investigation's capability to study the accuracy of ChatGPT in identifying "extend" associations is limited.

However, further examination shows that 100% of the "extend" associations were missing in the diagrams generated by ChatGPT. This suggests a significant deficiency in ChatGPT's capability to identify "extend" associations, as only 4 out of 20 exercises contained "extend" associations, and in all of these exercises, the "include" associations were missing. Notably, the average for missing "extend" associations was calculated based only on the 4 exercises containing such associations, not across all 20 exercises.

Regarding missing "include" associations, the analysis shows an average of 58% across the exercises where such associations were present. While this percentage is relatively better than that of missing "extend" associations, it is still concerning.

It can be stated that ChatGPT's capabilities in identifying associations, both "extend" and "include" types, appear to be weak. This suggests that users relying only on ChatGPT to generate use case diagrams should be cautious, especially regarding the completeness of necessary associations. Basic users, without much expertise in developing use case diagrams, should not solely rely on the diagrams generated by ChatGPT as they may lack crucial associations. Advanced users may be better equipped to detect and correct missing associations, thereby mitigating this issue to some extent.

4.1.4 Number of Necessary Prompts Evaluation

Exercis	NUMBER OF PROMPTS
1	2
2	3
3	2
4	3
5	3
6	1
7	1
8	2
9	2
10	3
11	2
12	2
13	5
14	2
15	2
16	2
17	2
18	2
19	2
20	1
AVERAGE	2.2

Figura 4.4. Prompts Evaluation.

To evaluate the effectiveness of ChatGPT in generating correct outcomes, the number of necessary prompts for each exercise was recorded in an Excel table. This table consisted of two columns: one listing the exercise numbers from 1 to 20, and the other indicating the corresponding number of prompts required.

The results revealed that the range of prompts required varied from a minimum of 1 prompt to a maximum of 5 prompts across the 20 exercises. The average number of prompts needed was calculated to be 2.2 prompts per exercise.

These findings suggest that while some exercises were effectively solved with minimal prompts, others required multiple iterations before a satisfactory outcome was achieved. In the subsequent sections, we will explore if there is any correlation between the number of necessary prompts and the accuracy of ChatGPT's outcomes in identifying actors, use cases, and associations.

4.2 Analysis of Use Case Narrative Exercises

To evaluate the quality of the use case narrative exercises, a "Pass/Fail" test inspired by Alistair Cockburn's methodology in "Writing Effective Use Cases" was conducted. This test consisted of 22 questions aimed at assessing the content of each use case narrative. These questions were selected based on their relevance and ease of evaluation.

A table was created with 11 columns, where the first column listed the question numbers and the subsequent columns represented each of the 10 use case narrative exercises (Exercise 1 to Exercise 10). Each cell in the table was filled with either "YES" or "NO" to indicate whether the corresponding exercise answered the question satisfactorily.

Following this evaluation, another table was generated, featuring four columns: "Exercise" (numbered 1 to 10), "Passed" (indicating whether the exercise passed the quality test), "%" (the percentage of "YES" responses out of the 22 questions), and "Score (max:22)" (the total score out of a maximum of 22 questions).

A threshold of 60% was set to determine whether an exercise passed the quality test. If at least 60% of the questions were answered with "YES", the exercise was considered to have passed.

Additionally, a histogram was constructed to visually represent the quantity of questions answered with "YES" for each exercise.

	EXERCISES									
QUESTIO	1	2	3	4	5	6	7	8	9	10
1	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
2	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
8	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
9	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
10	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
11	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
12	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
13	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
14	NO	YES	YES	YES	YES	YES	YES	YES	YES	YES
15	YES	NO	YES	YES	YES	YES	NO	YES	YES	YES
16	YES	YES	YES	YES	YES	YES	YES	YES	YES	NO
17	YES	YES	YES	YES	YES	YES	YES	YES	YES	NO
18	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
19	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
20	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
21	YES	YES	YES	YES	NO	YES	YES	YES	YES	NO
23	YES	NO	NO	YES	YES	YES	YES	YES	YES	YES
24	YES	YES	YES	YES	NO	YES	YES	YES	YES	NO
25	YES	YES	YES	YES	YES	YES	YES	YES	YES	NO
26	YES	YES	YES	YES	YES	YES	YES	YES	NO	NO
27	YES	YES	YES	YES	YES	YES	YES	YES	NO	NO
28	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES

Figura 4.5. 22 Question Analysis.

PASS/FAIL TEST			
Exercise	Passed_	%	Score (Max:22)
1	YES	95%	21
2	YES	91%	20
3	YES	95%	21
4	YES	100%	22
5	YES	91%	20
6	YES	100%	22
7	YES	95%	21
8	YES	100%	22
9	YES	91%	20
10	YES	68%	15
AVERAGES	100%	93%	20.4

Figura 4.6. Pass/Fail Test.

The results revealed that all ten exercises were able to pass the quality test, with an overall average percentage of 93% and a score of 20.4 out of

22. This indicates ChatGPT's high proficiency in generating high-quality use case narratives.

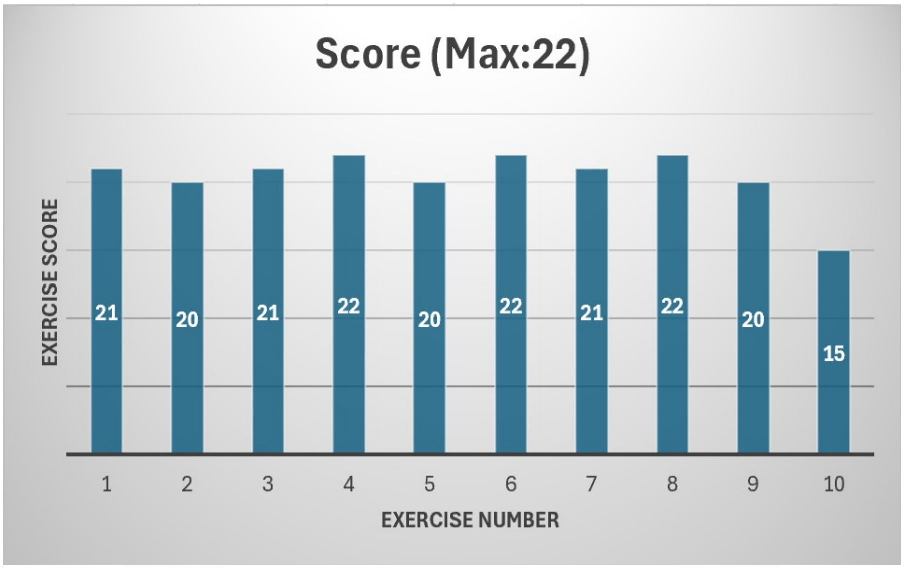


Figura 4.7. Rseults Histogram.

Having said all the above, we can say that given the subjective nature of use case narrative writing and the variability in templates and formats, leveraging AI tools like ChatGPT can significantly improve efficiency and ensure high-quality outcomes, as demonstrated by the positive results of this investigation.

4.2.1 Estimation of Exercise Difficulty

To provide a comprehensive assessment of the difficulty of each exercise, a table was constructed with six columns. The first column, labeled "Exercise," listed the numbers of the 20 exercises. The subsequent columns included "Number of Necessary Actors," "Number of Necessary Use Cases," "Number of Necessary Extend Associations," and "Number of Necessary Include Associations," providing quantitative data regarding the elements involved in each exercise. Finally, the "Mean Difficulty Score (Max: 5)" column presented the average difficulty score assigned to each exercise, ranging from 1 (very easy) to 5 (very difficult).

ESTIMATION OF EXERCISE DIFFICULTY					
Exercis	Number of necessary actors	Number of necessary UC	Number of necessary ext ass	Number of necessary inc ass	Mean Difficulty Score (Max=5)
1	4	0	0	0	2.33
2	3	0	0	0	3.67
3	4	0	0	0	3.67
4	5	0	0	0	3.67
5	2	0	1	1	3.00
6	3	1	1	1	2.33
7	1	0	0	0	1.33
8	2	0	0	0	3.00
9	6	0	0	0	2.67
10	3	0	2	2	2.00
11	1	0	0	0	2.33
12	1	0	0	0	1.33
13	4	3	1	1	4.67
14	2	0	0	0	1.00
15	1	2	2	2	5.00
16	5	1	0	0	4.67
17	3	0	1	1	3.67
18	2	0	0	0	3.33
19	3	0	0	0	4.00
20	2	0	0	0	3.33
AVERAGES	**2.85**	**0.35**	**0.4**	**0.4**	**3.05**

Figura 4.8. Estimation of Exercise Difficulty.

To generate the values for the "Mean Difficulty Score," an expert assessment involving three evaluators, including myself, was conducted. Each evaluator assessed the difficulty of the exercises based on the number of necessary actors, use cases, and associations. After each evaluator assigned scores to the exercises, the mean score was calculated, resulting in the final difficulty score for each exercise.

Furthermore, a table was created to display the scores assigned by each evaluator to every exercise, allowing for a detailed comparison of individual assessments. Additionally, this table included a column labeled "FK Ease Score (Max: 100)," which reported the difficulty of the textual descriptions of each exercise using the Flesch-Kincaid grade level tool. This tool evaluates the readability of texts based on factors such as sentence length and syllable count, providing an ease score ranging from 1 to 100. A higher ease score indicates easier readability, whereas a lower score suggests greater difficulty.

Upon analysis, the results indicated that the mean difficulty score for the 20 exercises was 3.05 out of 5, reflecting an average level of difficulty. The average scores given by each of the three evaluators were notably similar, demonstrating a high level of agreement among the experts. Specifically, the mean scores assigned by the evaluators were 3.1, 3.05, and 3, respectively.

	Score from evaluators			Select Evaluator	
Exercis	SCORE VEGA	SCORE COPPOLA	SCORE GARACCIONE	COPPOLA	FK EASE SCORE(MAX:100)
1	3	2	2	2	65.6
2	4	4	3	4	49.3
3	3	4	4	4	54.4
4	3	5	3	5	45.6
5	2	4	3	4	48
6	3	2	2	2	39.9
7	2	1	1	1	42.3
8	3	3	3	3	25.8
9	5	1	2	1	56.7
10	3	1	2	1	57
11	2	2	3	2	40.3
12	1	1	2	1	39.2
13	5	4	5	4	40.1
14	1	1	1	1	59.9
15	5	5	5	5	29.1
16	4	5	5	5	21.4
17	3	4	4	4	25.1
18	3	4	3	4	18.2
19	4	4	4	4	36.2
20	3	4	3	4	51
AVERAGES	3.1	3.05	3	3.05	42.255

Figura 4.9. Estimation of Exercise Difficulty.

Regarding the difficulty of the textual descriptions of the exercises, the average FK ease score was 42.3 out of 100. This suggests that, on average, the textual descriptions were moderately difficult to understand and required a reading grade level equivalent to that of a college student.

In summary, the evaluation of exercise difficulty revealed a moderate level of challenge for the exercises under investigation, supported by consistent assessments from the multiple evaluators. Additionally, the textual descriptions were found to be moderately difficult, underscoring the need for readers to possess a certain level of literacy to comprehend the content effectively.

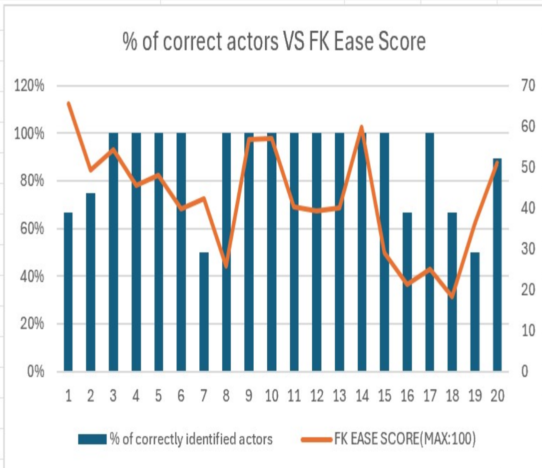
4.3 Correlation Analysis

One of the objectives that we tried to achieve for this work was identify or just analyze the relationship between the different variables that may affect the investigation, als by influencing the way in which Use Case Diagrams may be generated by the AI. Having this as a goal, I established some correlation analysis that were investigated and ahown more deeply in the flowing paragraphs, that somehow allows to understand the relationships between the factors. 4 different correlations were studied, with respect to the factors that contribute to the accuracy and efficiency of ChatGPT's output.

4.3.1 Correlation with FK Ease Score

The FK ease score was on important factor from the results of this investigation, given the importance of textual description for our study. That's why the first correlation analysis undertaken investigated the associations related to the Flesch-Kincaid (FK) Ease Score, which, as mentioned also before, measures the readability of textual descriptions. To this aim, three distinct tables were constructed, which showed the FK Ease Score with a different variable: the percentage of correctly identified actors, the percentage of correctly identified use cases, and the number of prompts required for each exercise. When the Pearson correlation factor was calculated, the result for each of the pair of data described a positive but low correlation.

EX.	% of correctly identified actors	FK EASE SCORE(MAX:100)
1	67%	65.6
2	75%	49.3
3	100%	54.4
4	100%	45.6
5	100%	48
6	100%	39.9
7	50%	42.3
8	100%	25.8
9	100%	56.7
10	100%	57
11	100%	40.3
12	100%	39.2
13	100%	40.1
14	100%	59.9
15	100%	29.1
16	67%	21.4
17	100%	25.1
18	67%	18.2
19	50%	36.2
20	89%	51
AVERAGES	88%	42

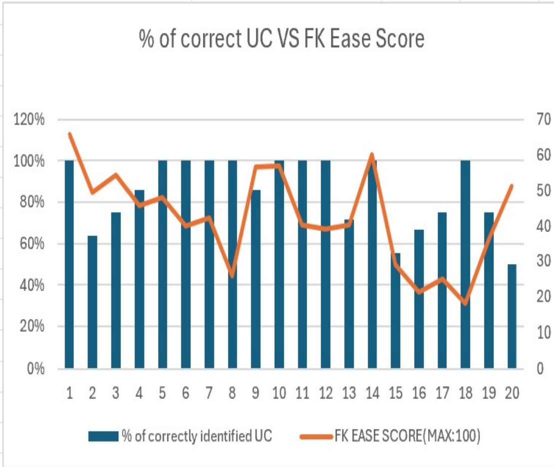


CORRELATION 0.16207497

Figura 4.10. Correlation of FK Score and % of Correctly Identified Actors.

To be more specific, the correlation between the FK Ease Score and the percentage of correctly identified actors was found to be 16%, indicating a weak positive relationship. Something very similar happened with the correlation between the FK Ease Score and the percentage of correctly identified

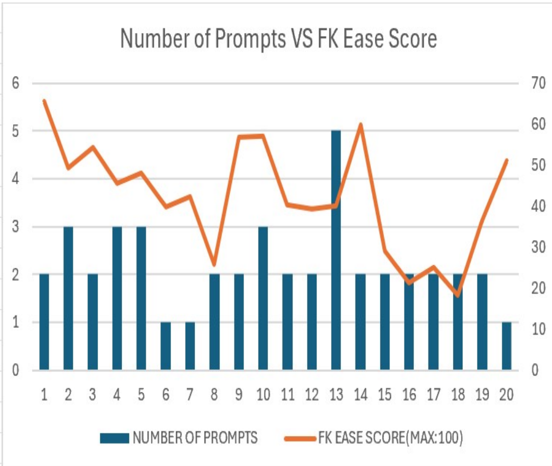
EX.	% of correctly identified UC	FK EASE SCORE(MAX:100)
1	100%	65.6
2	64%	49.3
3	75%	54.4
4	86%	45.6
5	100%	48
6	100%	39.9
7	100%	42.3
8	100%	25.8
9	86%	56.7
10	100%	57
11	100%	40.3
12	100%	39.2
13	71%	40.1
14	100%	59.9
15	56%	29.1
16	67%	21.4
17	75%	25.1
18	100%	18.2
19	75%	36.2
20	50%	51
AVERAGES	85%	42.26



CORRELATION 0.15312328

Figura 4.11. Correlation of FK Score and % of Correctly Use Cases.

EX.	NUMBER OF PROMPTS	FK EASE SCORE(MAX:100)
1	2	65.6
2	3	49.3
3	2	54.4
4	3	45.6
5	3	48
6	1	39.9
7	1	42.3
8	2	25.8
9	2	56.7
10	3	57
11	2	40.3
12	2	39.2
13	5	40.1
14	2	59.9
15	2	29.1
16	2	21.4
17	2	25.1
18	2	18.2
19	2	36.2
20	1	51
AVERAGES	2.2	42.26



CORRELATION 0.078953092

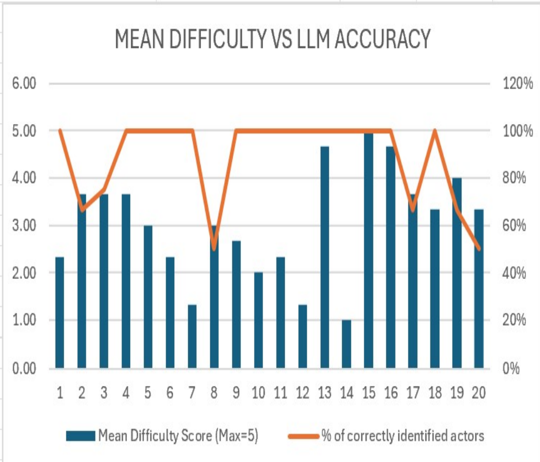
Figura 4.12. Correlation of FK Score and Number of Prompts.

use cases was 15.3%, showing a comparable level of correlation. Conversely, the correlation between the number of prompts and the FK Ease Score was even lower at 7.8%, which suugests us somehow that the reading difficulty of an exercise does not neccessairly have a big impact to the number of prompts required to generate an accurate diagram.

4.3.2 Correlation with Exercise Difficulty

On the other hand, and also considering it a very important subject of study, the second type of correlation analysis undertaken was focused on the relationship between exercise difficulty, as estimated by the three evaluators, and various variables. To this aim, again, three separate tables were created, each showing the mean difficulty score for an exercise with one of the following: the percentage of correctly identified actors, the percentage of correctly identified use cases, and the number of prompts. Surprisingly, two negative correlations emerged from this analysis.

EX.	Mean Difficulty Score (Max=5)	% of correctly identified actors
1	2.33	100%
2	3.67	67%
3	3.67	75%
4	3.67	100%
5	3.00	100%
6	2.33	100%
7	1.33	100%
8	3.00	50%
9	2.67	100%
10	2.00	100%
11	2.33	100%
12	1.33	100%
13	4.67	100%
14	1.00	100%
15	5.00	100%
16	4.67	100%
17	3.67	67%
18	3.33	100%
19	4.00	67%
20	3.33	50%
AVERAGES	3.05	89%



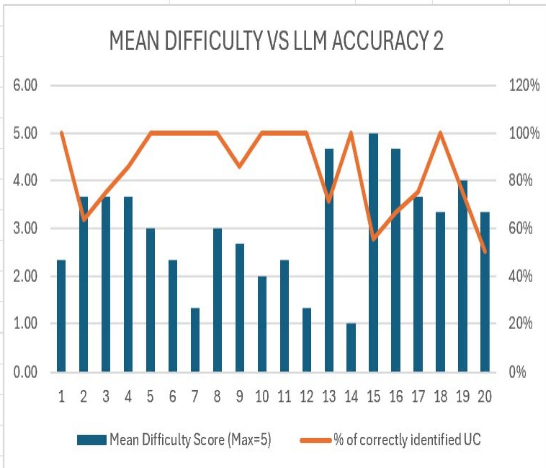
CORRELATION -0.251626949

Figura 4.13. Correlation with % of Correctly Identified Actors.

Firstly, the correlation between the mean difficulty score and the percentage of correctly identified actors was negative (-25%), suggesting that as exercise difficulty increased, ChatGPT's accuracy in identifying actors decreased, which is of course, a coherent result. Secondly, the correlation between the mean difficulty score and the percentage of correctly identified use cases was notably negative (-75%), which one again shows an expected result, meaning that when the exercise difficulty is higher, it is related to the lower accuracy of ChatGPT in this field. Conversely, a positive correlation of 37% was observed between the mean difficulty score and the number of necessary prompts, indicating that more challenging exercises tended to require additional prompts for satisfactory outcomes. This was also an expected result from a common-sense point of view.

These results highlight the complexity of the relationship between exercise difficulty and the variables influencing ChatGPT's performance, but at the same time allows the user to understand ChatGPT's expected outcomes with respect to the exercise difficulty, taking into account the number of required actors, use cases and associations.

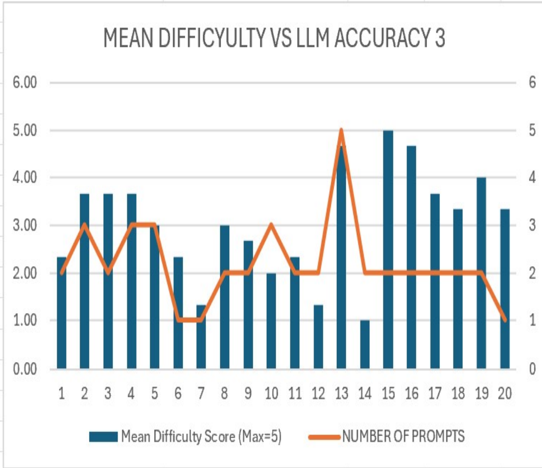
EX.	Mean Difficulty Score (Max=5)	% of correctly identified UC
1	2.33	100%
2	3.67	64%
3	3.67	75%
4	3.67	86%
5	3.00	100%
6	2.33	100%
7	1.33	100%
8	3.00	100%
9	2.67	86%
10	2.00	100%
11	2.33	100%
12	1.33	100%
13	4.67	71%
14	1.00	100%
15	5.00	56%
16	4.67	67%
17	3.67	75%
18	3.33	100%
19	4.00	75%
20	3.33	50%
AVERAGES	3.05	85%



CORRELATION -0.75794224

Figura 4.14. Correlation with % of Correctly Identified Use Cases.

EX.	Mean Difficulty Score (Max=5)	NUMBER OF PROMPTS
1	2.33	2
2	3.67	3
3	3.67	2
4	3.67	3
5	3.00	3
6	2.33	1
7	1.33	1
8	3.00	2
9	2.67	2
10	2.00	3
11	2.33	2
12	1.33	2
13	4.67	5
14	1.00	2
15	5.00	2
16	4.67	2
17	3.67	2
18	3.33	2
19	4.00	2
20	3.33	1
AVERAGES	3.05	2.2



CORRELATION 0.36960552

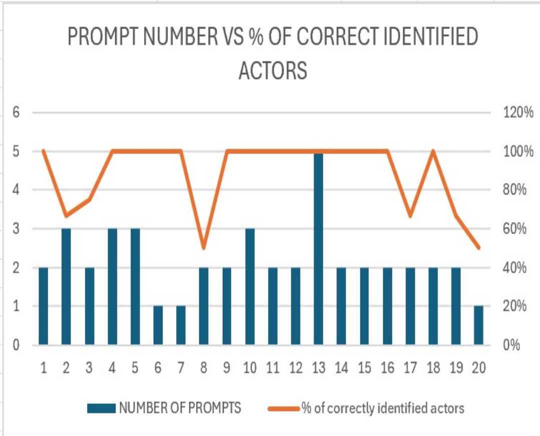
Figura 4.15. Correlation with Number of Prompts.

4.3.3 Correlations Related to Number of Prompts and ChatGPT's Accuracy

Moreover, another question that the reader may have regarding the investigation is related to the quality and accuracy of ChatGPT, related to the number of necessary prompts that were necessary for each exercise. Akcnowlidging this, two additional correlation analyses were conducted to in order to understant such relationship between the number of prompts and ChatGPT's accuracy in identifying actors and use cases. For this, separate tables were created to pair the number of prompts required for each exercise with the percentage of correctly identified actors and the percentage of correctly identified use cases, respectively. The results from the analysis were very interesting to analyze and revealed important insights into the interaction between prompt usage and ChatGPT's performance.

In the correlation analysis between the number of prompts and the percentage of correctly identified actors, a correlation of approximately 20% was observed. This result shows that in fact there is not such a strong correlation between the number of prompts provided and the accuracy of ChatGPT in identifying correct actors. This modest correlation was very important

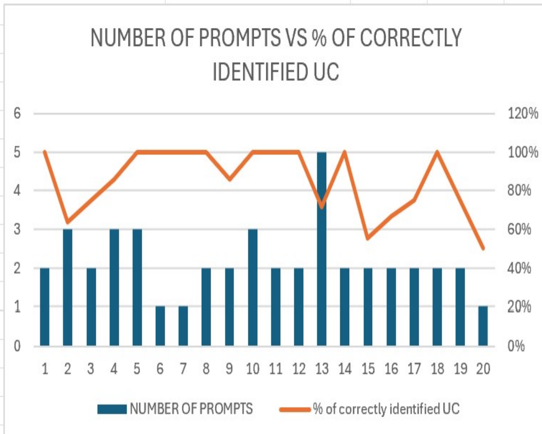
Exercis	NUMBER OF PROMPTS	% of correctly identified actors
1	2	100%
2	3	67%
3	2	75%
4	3	100%
5	3	100%
6	1	100%
7	1	100%
8	2	50%
9	2	100%
10	3	100%
11	2	100%
12	2	100%
13	5	100%
14	2	100%
15	2	100%
16	2	100%
17	2	67%
18	2	100%
19	2	67%
20	1	50%
AVERAGE	2.2	0.8875



CORRELATION 0.19729756

Figura 4.16. Correlation between Number of Prompts and % of Correctly Identified Actors.

Exercis	NUMBER OF PROMPTS	% of correctly identified UC
1	2	100%
2	3	64%
3	2	75%
4	3	86%
5	3	100%
6	1	100%
7	1	100%
8	2	100%
9	2	86%
10	3	100%
11	2	100%
12	2	100%
13	5	71%
14	2	100%
15	2	56%
16	2	67%
17	2	75%
18	2	100%
19	2	75%
20	1	50%
AVERAGE	2.2	0.851857864



CORRELATION -0.092752678

Figura 4.17. Correlation between Number of Prompts and % of Correctly Identified Use Cases.

and notable, since it contradicts the assumption that one could suppose, that more prompts necessarily lead to better outcomes, indicating a nuanced relationship between prompt usage and accuracy.

Continuing with the correlation analysis, the the results furthermore also revealed that between the number of prompts and the percentage of correctly identified use cases , there was incredibly a negative correlation of -9%. This shocking and unexpected result suggests us that there is an inverse relationship between prompt usage and the accuracy of identifying use cases. This result could be possibly understod subject to further analysis if it is prooved that exercises that require more prompts tend to be more complex, leading to decreased accuracy in identifying use cases despite additional input.

4.3.4 Correlations Related to Number of Prompts and Number of Necessary Actors/UC/Associations

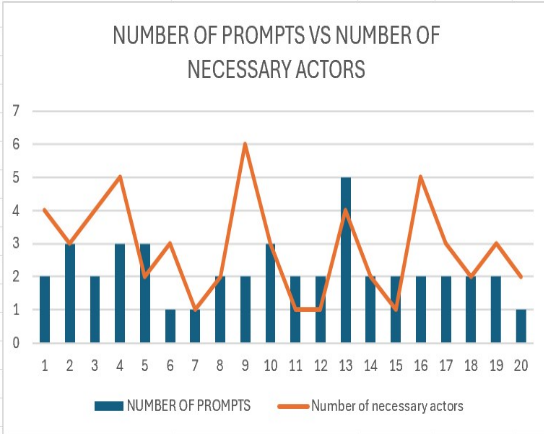
Another interesting insight that I wanted to explore in order to have a better understanding of the study, was to investigate if there existed correlations between the number of necessary prompts, and the number of necessary Use Cases, actors or associations. To this extent, in a deeper exploration of the relationship between prompt usage and the complexity of exercises, three additional correlation analyses were realized. These analyses paired the number of prompts with the number of necessary actors, use cases, and associations for each exercise, respectively.

Regarding the correlation between the number of prompts and the number of necessary actors, the results from analysis showed a positive correlation of 30%, which suggests us that the exercises whic require more actors are tending also also to need more prompts in order to achieve better outcomes. It is very important to highlight this, since it als gives an insight regarding the importance of providing adequate input to ChatGPT when dealing with exercises that involve numerous actors.

Surprisingly, the correlation between the number of prompts and the number of necessary use cases yielded a slightly negative correlation of -1.7%, which is a very unexpected result, but at the same time very valuable, which suggests that exercises requiring fewer use cases may paradoxically require more prompts to achieve accurate diagrams, which forsure is not something expected.

Finally, the correlation between the number of prompts and the number of necessary associations upshoed us a correlation which was stroner and expected, of 43%, which reveals the very important role of associations in

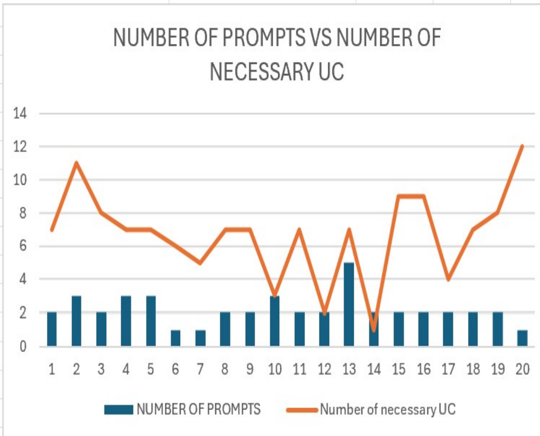
Exercis	NUMBER OF PROMPTS	Number of necessary actors
1	2	4
2	3	3
3	2	4
4	3	5
5	3	2
6	1	3
7	1	1
8	2	2
9	2	6
10	3	3
11	2	1
12	2	1
13	5	4
14	2	2
15	2	1
16	2	5
17	2	3
18	2	2
19	2	3
20	1	2
AVERAGE	2.2	2.85



CORRELATION 0.306123288

Figura 4.18. Correlation between Number of Prompts and Number of Necessary Actors.

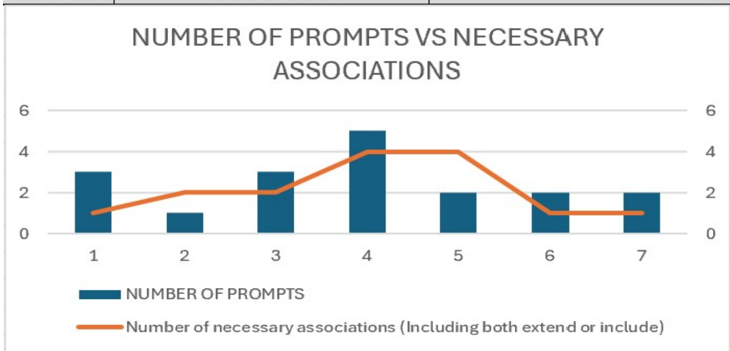
Exercise	NUMBER OF PROMPTS	Number of necessary UC
1	2	7
2	3	11
3	2	8
4	3	7
5	3	7
6	1	6
7	1	5
8	2	7
9	2	7
10	3	3
11	2	7
12	2	2
13	5	7
14	2	1
15	2	9
16	2	9
17	2	4
18	2	7
19	2	8
20	1	12
AVERAGE	2.2	6.7



CORRELATION -0.017329825

Figura 4.19. Correlation between Number of Prompts and Number of Necessary Use Cases.

Exercise	NUMBER OF PROMPTS	Number of necessary associations (Including both extend or include)
5	3	1
6	1	2
10	3	2
13	5	4
15	2	4
16	2	1
17	2	1
AVERAGE	2.571428571	2.142857143



CORRELATION 0.579519951

Figura 4.20. Correlation between Number of Prompts and Number of Necessary Associations.

determining the need for additional prompts. From this findings, we are able to state that ChatGPT showed a limited proficiency in identifying and incorporating associations often necessitated multiple prompts to ensure that they were finally included in the diagram.

These correlation analyses offer valuable insights into the nuanced relationship between prompt usage, exercise complexity, and ChatGPT's performance in generating accurate use case diagrams. Understanding these dynamics can inform strategies for optimizing the interaction with ChatGPT to achieve desired outcomes effectively.

4.4 Discussion and Interpretation

The results presented in the preceding sections focused on the capabilities and limitations of using ChatGPT for generating use case diagrams and narratives. This discussion aims to delve deeper into the implications of these findings and provide insights into the effectiveness of leveraging AI in software development tasks.

4.4.1 ChatGPT's Proficiency in Generating Use Case Diagrams

ChatGPT demonstrated to have a good proficiency in identifying actors and use cases across the exercises analyzed. The analysis revealed that, on average, ChatGPT correctly identified approximately 90% of necessary actors and 85% of necessary use cases. These results underscore the AI's ability to comprehend textual descriptions and translate them into structured diagrams effectively.

However, challenges emerged in accurately identifying associations, particularly "extend" relationships. The analysis showed a considerable deficiency in identifying "extend" associations, with all exercises containing such relationships showing missing associations in the generated diagrams. This limitation highlights a gap in ChatGPT's understanding of more complex UML structures, showing the need of further investigation and improvement.

The correlation analysis provided valuable insights into the factors influencing ChatGPT's performance. While there was a modest correlation between exercise difficulty and the accuracy of actor identification, unexpected trends appeared in the correlation between prompt usage and accuracy. The findings suggest that the relationship between prompt usage, exercise complexity, and diagram accuracy is very variable, requiring careful consideration in practice.

4.4.2 Quality Evaluation of Use Case Narratives

Regarding Use Case Narratives exercises and their quality, one can say that all of them were able to show excellent results, with all exercises passing the quality evaluation test. The level of proficiency from ChatGPT was very high in generating coherent and comprehensive narratives, and this is clearly evidenced by the average score of 93% across exercises, which underscores that the AI can extract very well the input from use case scenarios and generate narratives which are of high quality which are able to help users with software development tasks.

Moreover, this correlation analysis also deeped into the relationship between exercise difficulty, readability, and narrative quality. Something important to highlight and that was actually not a surprise, is the negative correlation observed between exercise difficulty and the accuracy of identifying actors and use cases. Actually, this result was expected because it suggests that ChatGPT can have higher difficulty in providing a better quality narrative for more challenging exercises, and this could be potentially due to the richer contextual information provided.

4.4.3 Implications and Recommendations

If we analyze the results and findings from this section, we can understand that they may have several implications for any person who cud possibly make use of any LLM for the automation of diagrams or narratives, specially for practitioners and researchers in the field of software engineering and natural language processing. This is sustained from the fact that the results are clearly able to show the potential of AI, specifically ChatGPT, to be a valuable tool for automating certain aspects of software development, such as generating use case diagrams and narratives which are the subject of study.

Even though the investigation show this valuable insight, I think it is highly recommended to understand the limitations of the ChatGPT at least on the current days, in the case when more complex UML structures are given, and particularly regarding to the inclusion of associations. Future research related to this study should focus on improving the LLM's comprehension of more complex software engineering concepts to improve the quality and completeness of the output generated. This is why it is very important to consider as a user to have at least some knowledge on the field, and not just to rely on the fact that ChatGPT will be able to help in the development of tasks. It is very important for the user to be able to at least understand the quality and completeness of the output generated by the AI. In other words, while ChatGPT can be really helpful and advantageous in terms of the time that as a user you could be able to save for the development process, human oversight and validation remain critical to ensure the quality and integrity of the final deliverables.

4.4.4 Future Directions

Something that I believe would be very important for the future related to the investigation, is to try to find new ways in which the AI could be able to help human beings in developing any kind of software development tasks. Some ways could be to potentially enhance the actual the models that we know, or even better, to develop new ones that could be better understanding software engineering stuff.

I think it's also important the opinion of software related people, like for example software developers, in understanding if this kind of models are really able to help them with their work or not, for this, I think it could be a great idea to carry on surveys or any other way that could let us analyze the actual help that this kind of models are able to support expert and basic users in their tasks . The fact of understanding over time how people use them for their "day a day" life, could probably let us learn more about what it's really like to use AI in software development.

4.5 Limitations and Future Work regarding UCD Exercises

Regarding the Use Case Diagram exercises, the investigation in overall suggests us that ChatGPT is a very capable tool in generating the diagrams, even though that it clearly demonstrated different levels of accuracy and completeness in its output. As presented before, some exercises were able to showcase almost a perfect diagram, while in other cases it failed to generate a good outcome and in this cases, it was required to prompt more times in order to generate a refined diagram. It could be said that when there were textual descriptions with some implicit information that should be taken into account, ChatGPT failed to detect it and to include it in the diagram, but the most important taking is that it was almost never able to detect and identify the correct associations and include them in the diagrams.

4.5.1 Heterogeneity of Exercise Sources and Difficulties

One big problem was that the exercises that were taken into consideration for the investigation were gathered from all sorts of different places, so they were not all the same in terms of their difficulty and detail. This mix of exercises makes the study more complete, but it also makes it harder to compare them directly. Future studies might work better if the exercises chosen for the the analysis are all about the same difficulty or from the same source, so it's easier to compare them.

4.5.2 Limited Association Coverage

One of the issues that I discovered after evaluating completely all of the exercises that I chose for the study, was that very few exercises had associations, and most of them didn't contain even one necessary association. This can be signalled as something to take into account, since it makes it hard to see how well ChatGPT can find and show associations. To this extent, it would be important for future studies to include more exercises with lots of associations to better test ChatGPT's skills in this area.

4.5.3 Dependency on ChatGPT

Another thing to highlight from our study is the fact that we only used ChatGPT an not any other LLM for the generation of the outcomes needed for our diagrams. This is important also, since even though that ChatGPT is one of the most known LLM now a days, other models could have had different results that would have been also good to study and that could change the results from the investigation. Of course this was not done because of lack of time to perform the same tests with many different models, but it could be a good idea for future investigations to perform this kind of analysis also with other models in order to test them, also, because software technology is very dynamic and we do not know the changes that this models may have in the medium to long term.

4.5.4 FK Ease Score Precision

Using the Flesch-Kincaid (FK) Ease Score to measure how hard the text is can be a bit off. This score just looks at sentence length and syllables, missing other things that make text complex. Future studies could use more detailed measures to better judge text difficulty.

4.5.5 Subjectivity in Exercise Difficulty Assessment

Another important thing that could be seen as a limitation is the fact that even if the 3 evaluators which provided a difficulty level for each exercise were supposed to have knowledge enough to show a reliable difficulty level, this is also a little subjective, in terms of the evaluation, because each person can consider different factors in order to give a score, and this clearly is something that can affect how results are interpreted. So, regarding this evaluation, even though in overall one could say that they are useful in order to have a better assessment of the study, future research could standardize the evaluation process or use multiple evaluators to make the difficulty assessments more reliable.

4.5.6 Evaluation of Other Diagram Types

In order not to generate confusions to the reader regarding the models reliability and usefulness in helping users with the automatization of requirements, it is highly important to remind that this study only looked at generating UCDs, meaning that one potential thing to do for future research could also see how ChatGPT performs with other diagram types, like class diagrams or sequence diagrams in order to give a better understanding of its strengths and weaknesses in creating different diagrams.

4.6 Limitations and Future Work regarding UCN Exercises

The review of the use case narrative exercises showed both strengths and areas for improvement. Each exercise was checked for clarity, completeness, and how well it matched with standard use case modeling practices. Some key points from the study are highlighted below.

4.6.1 Consistency and Completeness

Most of the exercises chosen for the investigation aimed to be consistent and complete, describing the important steps and interactions in each use case. Common elements like actors, triggering events, preconditions, and postconditions were always included, helping to understand the system's behavior.

4.6.2 Level of Detail

There were differences in how detailed the provided solutions and ChatGPT's solutions were. Some had detailed breakdowns of activities and alternative scenarios, while others were more brief. The right balance between detail and brevity depended on each exercise's needs.

4.6.3 Structural Clarity

The clarity of structure varied. Some solutions had a clear and organized flow of activities, while others could be clearer. Adding related use cases, stakeholders, and clear actor-system interactions made the narratives easier to understand.

4.6.4 Integration with Use Case Diagrams

A big issue was the lack of alignment between the use case narratives and their diagrams. This disconnect made it hard to evaluate the exercises as a whole, as differences between the text and diagrams hindered the full assessment of system requirements.

4.6.5 Recommendations for Improvement

Alignment with Use Case Diagrams

A mentioned before in other sections, and as presented, not all of the use case narrative exercises selected for study matched with the exercises about Use Case Diagrams, which putted a limitation in the overall assessment of the investigation, since other areas for analysis and for study could have been approached. To this extent, future exercises should make sure the use case narratives match their diagrams in order to help understanding in a better way system requirements and interactions.

Clarification of Roles and Responsibilities

Clearly defining actor roles and their responsibilities would help in understanding stakeholder interactions and system behavior better.

Enhanced Detailing

Finding the right balance between brevity and detail is key. Exercises should give enough detail to capture the system's behavior without being too wordy.

The evaluation showed the importance of clarity, completeness, and matching with standard modeling practices. By fixing the areas needing improvement and making sure text descriptions match the diagrams, future exercises can better convey system requirements and interactions

Capitolo 5

Conclusion and Future Work

The investigation tried to show the reader that for the software development process, the available tools and methods that we have at the reach of our hands, can be super important for human users to make things run smoothly and using resources wisely to take better profit of time, which is a limited resource that we should try to allocate in the best possible way. Use Case Diagrams and Use Case Narratives are very important parts of this, helping with gathering requirements, designing systems, and estimating work. These diagrams and narratives show how the system works and how users interact with it, which affects how successful and cost-effective software projects are.

As we know that Large Language Models now a days are very popular and highly utilized by human for many different kind of activities, in our specific case, we wanted to prove the capability and possibility of making use of Large Language Model, and in this specific case, ChatGPT, in automating Use Case Diagrams and Narratives. The main goal was to see if these models could save time and make human users more productive in software development.

The evaluation of the exercises and the analysis from the results of the outcomes from ChatGPT showcased both strengths and weaknesses in using LLMs for this task. ChatGPT was good at creating clear representations of system functions, but there were differences compared to those made by people. These differences show that human review and tweaking are still needed to make sure everything is accurate and complete.

The study also highlighted the partnership between LLMs and humans in software development. The important finding from here is mainly that even if LLMs can of course, generate initial diagrams and narratives much faster than a human user, the verification and knowledge about the area from human users are essential for refining and validating these outputs. Using LLMs in the development process can speed things up, helping with requirements and design work.

The benefits go beyond just saving time. Quick creation of Use Case Diagrams and Narratives can help with estimating the effort needed for software projects. Also, if we are able to have diagrams that showcase more quality in terms of correctness and completeness, it could really help in planning resources and timelines better, reducing uncertainties and risks in project planning.

To close up, this investigation has helped to show that the use of large language models by human beings is extremely helpful in the automation of the diagrams and narratives which were subject of study , but their real value comes from working together with human experts. What we are trying to say with this is that the real work should be done by the humans. We, as humans, must be extremely cautious about the use of these models, and most of all, we must understand that we dominate them, we instruct them, and they are just a tool to assist us. By using LLMs as tools to assist, software development teams can get the advantages of automation while keeping the deep understanding and creativity that humans bring. This teamwork approach can lead to more efficient, innovative, and successful software development.

Bibliografia

[1] A. Cockburn, Y. 2000. *Writing effective use cases*, Addison-Wesley Proffessional.

[2] K. Lano, Y. 2009. *UML 2 Semantics and Applications.*, John Wiley and Sons.

[3] K. Bittner e I. Spence, Y. 2003. *Use Case Modeling*, Addison-Wesley Professional.

[4] Elizabeth Larson, Cbap, Pmi-Pba, 2021. A PM#&8217;s *Guide to Use Cases Part 3: Use Case Narratives.* Available at: https://www.watermarklearning.com/blog/a-pms-guide-to-use-cases-part-3/.

[5] S. Shendall & A. Strohmeier, Y. 2001. *Guide to Use Cases Part 3: Use Case Narratives.* Available at: https://www.watermarklearning.com/blog/a-pms-guide-to-use-cases-part-3/.

[6] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, Y. 1992. *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley.

[7] A. Sharma & N. Chaudhary, Y. 2023. *Prediction of Software Effort by Using Non-Linear Power Regression for Heterogeneous Projects Based on Use case Points and Lines of code*, Elsevier. Available at:https://www.sciencedirect.com/science/article/pii/S1877050923001382

[8] Y. Mahmood & N. Kama & A. Azmi, Y. 2019. *A systematic review of studies on use case points and expert-based estimation of software development effort*, Wiley. Available at: https://onlinelibrary.wiley.com/doi/epdf/10.1002/smr.2245

[9] A. Effendi & R. Setiawan & Z. Erlisa Rasjid, Y. 2019. *Adjustment Factor for Use Case Point Software Effort Estimation (Study Case: Student Desk Portal)* , Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S1877050919311342

[10] M.Ochodek & J. Nawrocki & K. Kwarciak, Y. 2010. *Simplifying effort estimation based on Use Case Points*, Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S095058491000176X

[11] He, H. Y. 2023. *Robotgpt: from chatgpt to robot intelligence.*. https://doi.org/10.36227/techrxiv.22569247

[12] Rana, S. Y. 2023. *Ai and gpt for management scholars and practitioners: guidelines and implications.* FIIB Business Review, 12(1), 7-9. https://doi.org/10.1177/23197145231161408

[13] L.Fan & L.Li & Z.Ma & S.Lee & H.Yu & L. Hemphill. Y. 2023. *A Bibliometric Review of Large Language Models Research from 2017 to 2023.* School of Information , University of Michigan, Ann Arbor,MI.

[14] Wayne Xin Zhao,Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yi fan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Rui yang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian Yun Nieand Ji-Rong Wen. Y. 2023. *A Survey of Large Language Models.* GitHublink:https://github.com/RUCAIBox/LLMSurvey

[15] Faruk Enes Oguz, Mahmut Nedim Ekersular, Kubilay Muhammed Sunnetci & Ahmet Alkan. Y. 2023. *Can Chat GPT be Utilized in Scientific and Undergraduate Studies?* BIOMEDICAL ENGINEERING SOCIETY.

[16] Xin Liu, Daniel McDuff, Geza Kovacs, Isaac Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, Shwetak Patel. Y. 2023. *Large Language Models are Few-Shot Health Learners.*

[17] Laura Ruis ,Akbir Khan, Stella Biderman, Sara Hooker, Tim Rocktäschel1 , Edward Grefenstette. Y. 2022. *LARGE LANGUAGE MODELS ARE NOT ZERO-SHOT COMMUNICATORS.* University College London,EleutherAI, Booz Allen Hamilton.

[18] B.Andersson, M.Bergoltz & P.Johannesson. Y.2002. *Natural Language Processing and Information Systems.*, 6th International Conference on Applications of Natural Language to Information Systems, NLDB 2002. Stockholm, Sweeden, June 2002. Ed. Springer.

[19] E.Metais. Y.2002. *Enhancing information systems management with natural language processing techniques.* Data & Knowledge Engineering. Elsevier.

[20] L.Zhou, M.Shaikh & D.Zhang. Y.2007. *Natural Language Interface to Mobile Devices.* University of Maryland, Baltimore County.

[21] Akyildiz, I. F. and Guo, H. Y. 2022. *Wireless communication research challenges for extended reality (xr). ITU Journal on Future and Evolving Technologies.* https://doi.org/10.52953/qgkv1321

[22] A.Fan & B.Gokkaya & M.Harman & M.Lyubarskiy & S.Sengupta & S.Yoo & Jie M.Zhang. Y.2023. *Large Language Models for Software Engineering: Survey and Open Problems* Cornell University, arXiv:2310.03533v4 [cs.SE] 11 Nov 2023

[23] I.Ozkaya. Y.2023. *Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications.* IEEE SOFTWARE | PUBLISHED BY THE IEEE COMPUTER SOCIETY

[24] H.Rukshan Piyumadu Dias & C.S.L Vidanapathirana & Rukshala Weerasinghe & M.D Asitha Manupiya & R.M.S.K. Bandara Y.P.H.W. Ranasighe. Y. 2023. *Automated use case diagram generator using NLP and ML.* Ithaca: Cornell University Library, arXiv.org

[25] Miranda, Márcio Assis ; Ribeiro, Marcos Guilherme ; Marques-Neto, Humberto Torres ; Song, Mark Alan Junho. Y.2017. *Domain-specific language for automatic generation of UML models.*The Institution of Engineering and Technology.

[26] https://plantuml.com/faq

[27] John W.Satzinger & Robert B.Jackson & Stephen D.Burd. Y. 2015. *Systems Analysis and Design in a Changing World.* Cengage Learning.

[28] https://www.scribd.com/document/18122626/How-to-Use-Use-Cases

[29] https://ar5iv.labs.arxiv.org/html/1401.3069

[30] M. Usman Hadi & Rizwan Qureshi, & Abbas Shah, & Amgad Muneer, & Muhammad Irfan, & Anas Zafar, & M. Bilal Shaikh, & Naveed Akhtar,& Jia Wu, & S. Mirjalili. Y.2023. *A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage.* https://doi.org/10.36227/techrxiv.23589741.v2

[31] https://eulynx.eu/storage/simple-file-list/General-Documents/Modelling-Standard/20220502-Modelling-Standard-Eu_Doc_30-v4_0-0_A.pdf