Virtual Suspect

A Lying Virtual Agent

Diogo da Fonseca Caetano Rato

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Ana Maria Severino de Almeida e Paiva
           Prof. Rui Filipe Fernandes Prada

Examination Committee

Chairperson: Prof. Miguel Nuno Dias Alves Pupo Correia
Supervisor: Prof. Ana Maria Severino de Almeida e Paiva
Member of the Committee: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur

October 2016
Acknowledgments

Firstly, I want to thank to all the people that somehow collaborated in this work.

I would like to acknowledge my dissertation supervisors Prof. Ana Paiva and Prof. Rui Prada for the opportunity to work in such an interesting and challenging project with all the amazing people that I met in this journey. I also want to thank my advisors for their support, for the advices, the motivation and the sharing of knowledge throughout this dissertation.

I want to thank IST and its professors for all the things I have learnt in the past 5 years and to GAIPS’ team for the availability and great work environment that helped making this Thesis possible.

To my parents for their friendship and caring over all these years, for making me believe in myself, in my work, thank you. For always being there for cheering me when I was having a rough time, I am deeply thankful!

I would also like to thank all my friends and colleagues that helped me grow as a person and were always there for me, for the friendship, advice and the great support, thank you all!

Last but not least, I would like to thank Raquel for all the friendship, love and encouragement during the development of this Thesis. I am very grateful to had you alongside me in this adventure!
Abstract

Humans lie every day, from the least harmful lies to the most impactful ones. It is part of the strategies we use in our interactions to deal with conflicting situations like negotiation or sharing compromising information. Therefore, in an attempt to design virtual agents endowed with advanced decision-making abilities, researchers not only focused their effort in designing cooperative and truthful agents but also deceptive and lying ones. In this document we propose a model capable of engaging an agent in an uncooperative misleading dialogue with a user. This model gives to an agent the ability to reason about its knowledge and then autonomously adjust the story it tells depending on what its interlocutor might know and on how sensitive it considers the conversation topic to be. Such a model allows a story’s author to focus on the main narrative, letting the model handle the generation of alternatives. We implemented the model in an agent called the Virtual Suspect, used in an Interrogation Game, The game portrays an agent acting as a criminal suspect trying to mislead the police by concealing information about its past events. Using the Interrogation Game, a preliminary experiment was conducted to investigate the conditions in which the agent produces more lies. The results showed that participants who used interrogation skills similar to the one the police use, that is, asking open questions first and closed questions after, made the agent adapt its story more often, leading the interrogators to identify the deceptive behaviour of the agent.

Keywords

Adaptive Storytelling, Human-Agent Interaction, Deceptive Communication, Autonomous Agents
Resumo

Os seres humanos mentem todos os dias, das mentiras menos nocivos até às que têm maior impacto nas nossas vidas. É uma das estratégias usadas nas nossas interações diárias para lidar com situações com conflitos como negociações ou partilha de informação comprometedoras. Desta forma, com o objectivo de desenhar agentes virtuais com melhores capacidades de tomada de decisões, os investigadores não se focam apenas em desenvolver agentes cooperativos e honestos mas também em dotá-los com métodos para ocultarem informação e mentirem sobre ela. Neste documento propomos um modelo que confere ao agente virtual que o implemente a capacidade de participar num diálogo não cooperativo com o objectivo de enganar o utilizador. Este modelo dá a um agente os mecanismos necessários para racionar sobre o seu conhecimento e ajustar autonomamente a história que partilha com base no que o interlocutor sabe e no quão comprometedor é o tópico da conversa. Este modelo permite ao autor que escreve a história focar-se na narrativa principal, deixando para o modelo a tarefa de gerar as alternativas. Implementámos o modelo num agente chamado Virtual Suspect, usado num jogo chamado Interrogatório. Este usa o agente virtual como um suspeito de um crime tentando enganar um polícia ao esconder informação sobre o seu passado. Usando o Interrogatório, realizámos uma experiência preliminar para investigar as condições nas quais o agente produzia mais mentiras. Os resultados mostraram que os participantes que utilizaram técnicas de interrogatório semelhantes às usadas pela polícia fizeram o agente adaptar a sua história mais frequentemente.

Palavras Chave

Ajustamento de Histórias, Geração de Mentiras, Interação Humano-Agente, Agentes Autónomos
# Contents

1 Introduction ................................................. 1
   1.1 Motivation ............................................ 5
   1.2 Problem ............................................... 5
   1.3 Contributions ......................................... 6
   1.4 Document outline ..................................... 6

2 Background .................................................. 9
   2.1 Deception ............................................. 11
   2.2 Cognition of Deception ................................. 12
   2.3 Theory of Mind ......................................... 12
   2.4 Police Interview ....................................... 13

3 Related Work ............................................... 15
   3.1 Virtual Suspect ........................................ 17
   3.2 Interactive Storytelling ............................... 19
      3.2.1 Agent’s Interaction in Virtual Storytelling .... 19
      3.2.2 The Virtual Storyteller: Story Creation by Intelligent Agents .... 20
      3.2.3 Narrative Mediation ............................. 21
      3.2.4 Game AI as Storytelling ......................... 21
   3.3 Memory architectures for agents .................... 24
      3.3.1 I Know What I Did Last Summer ................ 24
      3.3.2 MAY: my Memories Are Yours ................... 25
   3.4 Deceptive Virtual Agents .............................. 25
      3.4.1 GOLEM ........................................... 26
      3.4.2 MOUTH-OF-TRUTH ............................... 27
      3.4.3 Deception Planner ............................... 28
      3.4.4 Great Deceivers .................................. 28
4 Agent's Model

4.1 Story Representation

4.2 Interaction

4.2.1 Question

4.2.2 Answer

4.3 Architecture

4.3.1 Update Theory Of Mind with Question

4.3.2 Select Strategy

4.3.3 Adjust Story

4.3.4 Filter Answer

4.3.5 Update Theory Of Mind with Answer

4.3.6 Retrieve Similar Entities

4.3.7 Process Query

5 Use Case: a Virtual Suspect

5.1 Police Protocols

5.2 Implementation

5.2.1 Knowledge Base

5.2.2 Query Engine

5.2.3 Layers

5.2.4 Assembly the Virtual Suspect

5.3 Natural Language Generation

5.4 Interaction's Interface

5.4.1 Interaction's Flow

5.4.2 First Prototype

5.4.3 Second Prototype

5.4.4 Prototypes Remarks

5.5 Who is Peter Barker?

6 Demonstration

7 Evaluation

7.1 Sample

7.2 Measurements

7.3 Methodology and Procedures

7.4 Results

7.5 Empirical Results

7.6 Discussion
8 Conclusion
  8.1 Future Work ................................................................. 78
A Project Source Code Samples 85
  A.1 Knowledge Base Code ..................................................... 85
  A.2 Query Engine Code ....................................................... 86
  A.3 Virtual Suspect Code ..................................................... 87
B User Studies ................................................................. 89
List of Figures

3.1 The flow of information in the system ........................................ 18
3.2 Story’s internal representation using AND/OR graphs ................... 20
3.3 State Space with multiple deviation from the expected trajectory ......... 22
3.4 Main branch and contingency plans for unexpected user’s action ......... 23
3.5 An event constructed on a episode where Luke hits John .................. 24
3.6 May’s Memory System .......................................................... 26

4.1 Entities Example ................................................................. 32
4.2 Event Example ................................................................. 33
4.3 Story with one event example ................................................ 34
4.4 Timeline with conflicting events .............................................. 34
4.5 Section of the stories’ timeline, with two shared events ................. 35
4.6 Condition representing a constraint between the “Location” and the Value “Técnico” ................................. 37
4.7 Question Example .............................................................. 37
4.8 Answers Example .............................................................. 38
4.9 The virtual agent’s vertically layered two-pass architecture. By definition [1] the architecture is represented vertically, however, for space reasons, it is drawn horizontal with the bottom layer on the left. ................................................ 39

5.1 Agent’s graph with multiple nodes associated in events .................. 49
5.2 Interaction’s Flowchart ......................................................... 54
5.3 First Prototype’s interface using WPF ...................................... 55
5.4 Second Prototype’s interaction window sketch ............................. 56
5.5 Second Prototype’s interface implemented in WPF ......................... 56

6.1 Query representation for the question “Did you go to the jewelry store on March 3rd or 4th?” .................................................. 61
6.2 Story section with events satisfied by the question “Did you go to the jewelry store on March 3rd or 4th?” 62
6.3 Duplicated events that replace the compromising section of the Real Story 62
6.4 Query result representing the answer 63
6.5 Virtual Suspect’s stories timelines before and after the question was proposed (The dotted line represents the shared story and the solid line the original story) 63

7.1 Graph presenting the ratio of Yes and No answers for each accomplices 71
7.2 Graph presenting the percentages of each answer to question Q2.8 71

B.1 Questionnaire Page 1 90
B.2 Questionnaire Page 2 91
B.3 Questionnaire Page 3 92
List of Tables

4.1 Available methods to adjust the story (grouped by strategies) .......... 42

7.1 Descriptive Statistics for Questions Q1.3 and Q1.4 ......................... 70

7.2 Answers to the questions Q2.1 and Q2.3 ................................. 70

7.3 Results of the Virtual Suspects Logs divided in the two groups identified .... 72
# List of Algorithms

1. Algorithm representing the agent's cycle ........................................ 40
2. Procedure to update the Theory of Mind using the question asked .......... 41
3. Procedure to retrieve a list of entities similar to another entity ............ 43
4. Procedure to process the Knowledge Base using the question's query .... 44
Listings

A.1 XML file structure representing the Agent's Story. . . . . . . . . . . . . . . . . . . 85
A.2 "Location Filter implementation" . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
A.3 Time Focus Predicate implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
A.4 Code to query the Knowledge Base with the question asked" . . . . . . . . . . . . 87
A.5 "Agent's initialization code" . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
A.6 "ProcessQuestion’s body” . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
Introduction

Contents

1.1 Motivation ................................................................. 5
1.2 Problem ................................................................. 5
1.3 Contributions ......................................................... 6
1.4 Document outline .................................................... 6
As humans, we use deception daily as a tool to avoid problems in a specific context or in long-term scenarios, but producing a lie is not a straightforward task, it demands an effort way bigger than telling the truth [2]. A person that wants to deceive someone must go through a mental process to successfully create a robust and consistent lie that does not contradict itself when compared with other statements [3]. A lie is easier to maintain if it is simple, but an intricate deceit that has multiple dependencies and relates different events is harder to keep credible due to the increased number of people that, because they are connected to the lie, can detect flaws on its content. In addition to the awareness of the people involved by the false statements, the planning needed to make a lie solid enough to avoid being exposed by unpredictable future events is very hard and could even be impossible. Alongside the task of building a perfect lie, it is important to make it believable when the deceiver tells it, a person looking nervous may not transmit the confidence needed to support the created statement.

Creating a lie is already a hard task, but maintaining it throughout time requires a big mental effort since the liar needs to keep track of it in parallel with the real events [4]. This task is demanding for humans to perform because there is the necessity to memorize two different stories about the same situation, where one is fake, it is not supported by real events making it harder to remember, and the other is real, it is factual and happened [5].

Empowering a virtual intelligent agent with deceptive capabilities can improve their social interaction skills allowing it to decide whether to say the truth or to lie. An agent capable of lying can be useful in different situations, such as in tutoring scenarios to teach trainees techniques to identify deception, in interactive storytelling to create a richer plot, or to allow NPCs in video games to mislead the player, creating a challenging and harder experience.

There are several virtual agents that use deceptive techniques in their interactions with other agents and with users, either to detect deception [6] [7] or to negotiate resources between different entities using lies [8] [9]. The environments used to test these agents are mainly games, and they had a close and well-known domain allowing both parties of the interactions to be aware of the actions and their possible consequences.

While games have a specific set of actions the players can choose from, dialogues between humans can be about everything and demand an ample world knowledge. Recently, researchers focused their effort in creating virtual agents capable of engaging in a conversation with humans about its previous actions and their interactions. A topic addressed in these investigations is the memory architecture used to store the information to be shared later. Some implementations of virtual agents base their memory on the concept of autobiographic memory [10] to create agents capable of recalling and talking about their past experiences [11] and virtual companions that remember previously shared memories [12].
To endow virtual agents with the capacity of not only talking about previous actions or stories but also to hide some of its parts, it requires the use of mechanisms to adjust the information shared with the users. In Interactive Storytelling, several researchers developed methods to adjust narratives based on the users actions [13] [14]. Some work focused on dealing with unexpected actions and others on assuring some paths were followed by the user, but across all the systems studied we found a common goal: mitigate the task of explicitly dealing with all the branching a story could have. These support tools help the author focus on the story itself and not on dealing with user's behaviour.

As already mentioned, a conversational agent with deceptive capabilities could be used in training games and, with the expertise gathered from different fields of artificial intelligence, a scenario that uses an intelligent agent can be a challenging one, producing the necessary practice environment to develop students’ skills. A virtual agent with such capabilities must communicate with the user. Research such as Virtual Suspect: Response Model by Merijn Bruijnes et al. studied the interaction between police investigators and suspects during an interrogation to propose a model for a question-answering interaction [15].

Long-term, recalling everything from his past is difficult for the human being to do and it is even more to remind fake situations. Computers, however, have efficient and reliable mechanisms to store information and access those records, concealing one of the major reasons humans are caught lying, temporal inconsistency [16]. For humans it is challenging to infer a deceit in a complex context that is consistent with other’s knowledge due to the huge number of possible outcomes, but machines can use planning and search algorithms to evaluate and choose a lie that best suits a specific situation. On the other hand, a computer has limited knowledge about the world with a reduced capability to increase its domain while humans are able to learn very fast about almost any domain.

If we were to develop an agent capable of lying about everything in every context, the domain known had to be unlimited. For this reason, the work to be done will focus in developing a virtual agent that has to deceive a police officer in an interrogation scenario. This context gives a limited, yet varied enough, domain to test the interaction between a virtual agent and a human being. The deceiver will take part in the conversation as a criminal that needs to hide the real story about a crime in order to avoid getting caught by the detectives while the user will be the officer questioning the suspect about his actions regarding the crime being investigated.
1.1 Motivation

The development of a virtual agent that is able to share and modify a false story to match the users’ actions and knowledge requires the story's author to explicitly describe how the narrative should be modified for each one of the user's possible interactions. This process forces the author to invest a lot of effort in predicting the possible courses the main story can take and specifying the corresponding alternatives, instead of focusing on writing the original narrative. We intend to reduce the authoring process associated with this issue by designing a model capable of generating alternative stories without being previously scripted.

1.2 Problem

This dissertation proposes a model of a virtual agent capable of lying about a story, this is to say a sequence of events, without predefined alternative stories. The approach chosen was to develop a virtual agent's model that is able to create and share false events that hide compromising sections of the original story during a question-answering interaction with a user. With this set of requirements an objective was defined:

*How can we build a model for a lying virtual agent that is capable of maintaining in memory a real story and in parallel adjust a false story to match the user's knowledge without revealing incriminatory information?*

To achieve the objective we aimed for, we were concerned with the adjustment and maintenance of the lies created during the interaction: the memory and its fragments need to be changeable and replaceable, the lie generation must be fast to make the interaction seamless, the alternatives generated must be consistent with the agent's internal state and the user's knowledge, and the required interaction must be flexible and structured. With all these concerns to address, we tried to prove the following *hypothesis*:

*If a virtual agent, that has knowledge about other's beliefs, knows the information that should not be revealed, and can compute alternative stories during a conversation, it will mislead the user by sharing a parallel story that hides compromising events about its past actions.*

To validate our hypothesis, we studied theories about human's cognitive process when generating a lie, and how police officers use this information to conduct their interviews. Then we investigated how memory is stored in agents, the techniques to deal with users actions and the impact of Theory of Mind in varied virtual agent's performance. After that, we designed a model for a conversational agent and implemented it using a crime suspect point of view as the agent's
story. Finally, we evaluated the agent in sessions with users where we asked them to discover some details about its story that were missing. With the results of these experiments, we tried to verify the hypothesis.

1.3 Contributions

By the end of the project, we had the following contributions:

• A Virtual Agent model capable of creating and adjusting false events that replace compromising sections of a story without previously scripted alternative stories.

• An implementation of the model proposed, the Virtual Suspect, that plays the role of a jewelry robbery's suspect in a police interview.

• An interface for the interaction between users and the Virtual Suspect used for the user studies.

• A study to test if the Virtual Suspect was able to mislead the participants and to analyse how the agent’s performance would change in function of the question sequence asked by the interviewer.

1.4 Document outline

The agent's model design and implementation demanded a research about different concepts such as Deception and Theory of Mind. We also investigated how Police Interviews were conducted and what were their protocols and guidelines. This study is presented in Chapter 2.

Already familiarized with the concepts involved in our project, in Chapter 3 we reviewed related and relevant work to our agent's model from other deceptive virtual agents to interactive storytelling systems. We also studied which memory architectures are used in agents and how narratives are computationally represented.

The following chapter, Chapter 4, describes the virtual agent's model proposed. We present its memory structure and how the story is decomposed. Then we elaborate on the interaction used between the virtual agent and the user and, after that, we cover the proposed architecture and the mechanisms to endow a virtual agent with the desired functionality.

Chapter 5 explains how the Virtual Suspect was implemented using the virtual agent’s model considering the police protocols studied. This chapter also covers the prototypes developed for the virtual agent's interaction with users and the crime script used to test the agent.
To show how the Virtual Suspect and its components would process the questions proposed, we did a demonstration in Chapter 6. After that, in Chapter 7, we described the evaluation done with users and discussed the results observed.

Finally, Chapter 8 has the conclusion of this dissertation and some proposals for future works based on the virtual agent’s model.
2 Background

Contents

2.1 Deception ................................................. 11
2.2 Cognition of Deception ................................. 12
2.3 Theory of Mind ........................................... 12
2.4 Police Interview ........................................ 13
In the following sections, we present the study done about several theoretical concepts related with our work. We cover different forms of deception and how is this concept defined. Then we present theories that explain the process humans use to create lies. After that, we present a brief description of the concept of Theory of Mind and how it can be incorporated.

Finally, we review how police interviews were conducted and the approaches used by the interviewers to extract the maximum amount of information from the interviewees.

2.1 Deception

Deception is a common technique used by all living forms to achieve their goals: some flowers deceive pollinators by looking like other plants that have rewarding nectars [17], shrimps that lost their exoskeleton are very vulnerable yet they behave very aggressively towards offenders to intimidate them [18] and humans lie about everything in casual conversations with others. While in nature using deception happens when a living being has to fight for his life or for its species’ future, as humans we use deception to avoid situations of almost any type, from the ones with less impact in our life, such as lying about leaving a window opened in a rainy day, to more serious offences like having an affair with someone or about acts of violence [19]. Also, animals in life-threatening situations rely on their instincts, learned over hundreds of years of evolution, to deceive successfully, whereas humans weight the pros and cons of lying considering its impact on what others think about them, on their self-esteem and if it gets them closer to their objective.

Most justifications people gave to their everyday lies are benign, however the use of serious lies, that is, lies that affect deeply people’s life, is motivated by self-serving reasons with the intention of hiding bad behaviours or avoiding shameful situations [20]. When creating a lie to deliberately deceive another human being, the liar is aware that he is creating a false version that diverges from the truth with the objective of misleading someone. This deceptive communication requires two parties to be involved, a conversational context and their interpersonal relationship [21]. In addition to the interpersonal component of deceiving, lying requires different cognitive mechanisms to cooperate and demands a big mental effort [22].

A lie that is plausible needs to be internal consistent, its logic should be solid and have no contradictions, and external consistent, the statement should agree with others’ knowledge [3]. Aside from the social interaction between both parties in the conversation and the cognitive processes used to produce a lie, the use non-verbal communication skills to align the content of the false statement with the emotions of the speaker can make the deception more believable and credible, but this task requires the activation of even more cognitive processes [2] [3].
2.2 Cognition of Deception

A model to explain how humans’ cognitive processes work when generating a lie was present by Walczyk, Roper, Seemann, and Humphrey in 2003. They described deception as a cognitive process with three main stages: Activation, Decision and Construction [22]. This process activates two types of memories: the Working Memory (WM), that has information regarding the current process, and the Long-Term Memory (LTM), a component that stores general information about the world and cognitive knowledge concerning the beliefs of others, also called Theory of Mind. To lie in an answer, the deceiver perceives the question, activates truthful information about it, and transfers the related memories from the LTM to the WM. While the activation step can be automatically activated, the next phases, decision and construction, are intentional. Based on the WM and links to the LTM, the potential liar decides whether or not to answer with a lie. Whatever the choice, the construction component is activated. If the person decided to answer honestly it will pursue pointers to the LTM and generate a response based on this memory. On the other hand if he decided to lie, the liar needs to construct a plausible and consistent lie on the WM.

Instead of describing the flow of brain’s activations, in 2007, Victor Gombos proposed a theory to explain how a person uses his components of cognition to produce lies [5]. The theory relates the role of different executive processes, like the attention and meta cognition, the working memory and inhibition, with the task of creating lies, suggesting that they all need to work together to “inhibit inappropriate representations or responses and activating an appropriate one”. His work results determined that children’s ability to deceive rely more on the inhibition of information and the executive functions efficiency decrease with old age making it harder to deceive successfully. It was also observed that the need to deceive in complex situations required an higher mental effort with the activation of multiple cognitive processes.

2.3 Theory of Mind

According to Baron-Cohen, Theory of Mind (ToM) is a process used to model what others are thinking, that is, the ability to infer about other’s mental states like beliefs, desires, intentions, imagination and emotions [23]. A person is considered to have the best level of ToM if he can represent its belief about what someone else is thinking, and he has the second level of ToM if he is capable of modelling what someone is thinking about other’s beliefs.

According to Meyer et al., there are two different approaches to incorporate ToM capabilities: theory-theory, an agent has to use declarative models to represent the reasoning of other agents, and simulation-theory, an agent can simulate reasoning over its own model about others’ beliefs [24].
Knowing other’s belief about him, a lying virtual agent can prevent the detection of future vulnerabilities in his false statements. He can infer about other’s mind content, giving additional information to help him decide whether or not to lie and what to lie about. Being able to reason not only over its mental model but also about what others are thinking improves the chances of creating a solid lie consistent with everyone’s knowledge [3].

2.4 Police Interview

In order to develop a virtual agent capable of lying on a game about police interviews, it is important to understand how the task is performed by law enforcement around this world, who are the stakeholders and the strategies used by each side of the conversation.

A police interview, also known as questioning, cognitive interviewing and, more commonly known as interrogation, is an example of non-cooperative dialogue between two sides with divergent goals that, using verbal and non-verbal communication, interact with each other. On an interrogation, there are two sides easily identifiable, the group that interviews, that has one or more members from the authority responsible for the investigation, and facing them there is a person being interrogated, to whom the questions are addressed, that can be advised by a lawyer.

From the interviewer’s perspective, the job is to retrieve as much as possible information relevant to the case without breaking the rules imposed by the police. This set of rules, called a protocol, is used not only to guarantee that the evidence and testimonials gathered in the interview are admissible in court, but also to help the interviewer during the dialogue by providing him a set of best practices with the view to establish rapport and gain his trust [25]. Besides the protocol, the police should be aware that the person being questioned might be aggressive with no intentions of cooperate or could be someone so submissive willing to admit something that he did not do just to avoid more trouble. In both scenarios, police should adapt their methods to maximize the cooperation between both parties [26].

There are two types of methods used in cognitive interviews: information gathering methods and accusatory methods [27]. Information gathering methods try to establish rapport with the interviewee such that a good interpersonal relation can be built. After that relationship is established, the interviewer asks questions about the questionee’s lifestyle and personality, allowing him to speak as much as possible. The accusatory methods, as the name suggests, try to get a confession from the suspect by judging his actions, exposing lies in his answers and confronting him with evidence that can make him look guilty. The two approaches are not independent, most of the times they are combined such that on the rest stage of the interview uses information-gathered methods while in the end accusatory methods are used.
Even though the job done and strategies used by the police on the interrogation offers a lot of information, the most helpful source of knowledge to this project is collected after analysing the interviewee’s point of view. The person being interrogated does not have to be a criminal or a suspect, he can be a witness with useful information about a crime, but besides his position in the investigation’s context, the biggest obstacle appears when he is not willing to cooperate with law enforcement, either by remaining silent or telling lies to hide the real story [28].

When the person being interviewed knows if he is a suspect or not, he can adapt his behaviour according to that information but should also be aware that with the discovery of new facts, he might turn into a suspect. Most of the times, while being interrogated, a witness is unwilling to cooperate because he thinks his privacy is threatened. In this scenario, the task on the police is to gain the witness’ trust and create a situation where he feels comfortable to talk. Beyond feeling like a witness, a suspect knows that, to the police, he might be a criminal. The safest approach would be remaining silent and deny any accusation, however, the best way to avoid further problems is proving his innocence. Cooperation with law enforcement is key on this situation, answering the interviewer question honestly, will grant him the confidence of the police. If the interviewee is a criminal, there are only two possible ways of ending the investigation: confessing the crime or deceiving the police about his true actions. To avoid getting caught, he must deny answering all question or lie to the police to make them follow wrong clues. The main problem about choosing the second method comes up when the interviewer exposes those lies with evidence and the criminal has no other way of misleading the police.

As seen in 2.1, a person has a increased cognitive load when is lying and, on the context of a cognitive interview, the criminal also have an higher mental effort to create a believable lie. A criminal that has to mislead the police in a interrogation needs to be aware of multiple aspects regarding the conversation content and the beliefs of the interrogator. He needs to make up a story and remember it so that it is credible and coherent to the police’s knowledge. Also a liar is less likely to take his credibility for granted [16], he needs to monitor and control his reputation. A criminal needs to observe the interviewer’s reactions to know if his lies are credible from the police’s perspective and use this feedback to improve his act and role-play. Also, when describing real events, the activation of the executive processes is automatic, while the creation of lies needs intention and deliberation [22]. The combination of all this aspects results on a cognitive demanding situation for the criminal when trying to build a solid alibi that seems credible to the police [2].
## Related Work

### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Virtual Suspect</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Interactive Storytelling</td>
<td>19</td>
</tr>
<tr>
<td>3.3 Memory architectures for agents</td>
<td>24</td>
</tr>
<tr>
<td>3.4 Deceptive Virtual Agents</td>
<td>25</td>
</tr>
</tbody>
</table>
Creating a virtual agent capable of deceiving is a job that requires the expertise from multiple fields of computer science. To better understand the work already done in each subject, this section groups the research in different categories. First we explore the work done on the Virtual Suspect project to find relevant aspect of its research. After that, we study interactive storytelling systems and different agents’ memory architectures. Then we review some deceptive agent’s implementations.

3.1 Virtual Suspect

"Keeping up Stories: Design Considerations for a Police Interview Training Game" is a discussion paper about the problems that arise when developing a virtual suspect to teach the Dutch police about the social skills needed in police interviews [29]. The main goal of the work done by Merijn Bruijnes et al. was to train Dutch police in social and non-cooperative interactions. They focus on filtering the information told by the virtual suspect in function of the interpersonal relationship between both parties of the conversation, but the model does not elaborate on the deception techniques the suspect uses to hide his story. Despite choosing different approaches, some of our virtual agent's problems are related with the ones addressed in theirs.

One of the fields the research touches that is relevant to our work is the Dialogue Manager. For our work we will abstract its natural language component, but some concepts are worth mentioning. They propose a turn based dialogue in which both participants interact using actions and these affect directly the information state of the action's target. They used a paired structure for each action, it contains an action type, such as make statement, infer, accuse and respond, and a questioning strategy. The information state's formalization design is based on the architecture proposed by S. Larson and D. Traum [30]. Their approach involves formalizing actions and information, so that the system is able to define goals for agents and plan sequences to reach these goals. It is also worth considering the way they suggest to deal with the limited domain available. Because the system can not deal with all the possible actions in the world and the interviewer does not know about the completeness of the agent's knowledge, they recommend the use of a wizard that shows all the actions a user can chose from.

The other problem they elaborated was the Virtual Suspect's Response Model [15]. Merijn Bruijnes et al. studied the aspects of the question asked by the police and how the suspect answers that question in order to create a response model that includes these aspects. The response model depends on the suspect's personality and the dynamic state of the non-cooperative dialogue involving him and the police. The response model for the virtual agent receives the question using a set of nine input-variables, called Question Frame. To create his answer, the virtual
agent's uses the information perceived and the response model's internal state. This state includes the persona specification, also called Personality, the Interpersonal Status between the police and the suspect, and the Suspect Stance. Combining all these values, the agent's model calculates a new Answer Frame and, before giving it to the police, it updates the interpersonal status and a new stance is created for the suspect. Figure 3.1 has a detailed graphic representation of the response model's information ow described.

![Figure 3.1: The flow of information in the system](image)

The model created was based on observations of Dutch police interviews. Using this system, the police trainees can compare their intentions to the way the agent interpreted his intentions by looking at the model's frames and evaluate if the type of interaction used was the most appropriate one. Merijn Bruijnes et al. built an embodied conversational agent to play the role of a suspect in a crime investigation [31]. This agent had changeable personalities and would be used to evaluate if a user would perceive that each persona had a different behaviour. The agent was implemented with components from the Virtual Human Toolkit [32] and the interaction was done using natural language. Most of the participants were able to guess correctly which persona was the agent and some users detected changes in the virtual suspect's behaviour throughout the interaction, supporting the agent's ability to dynamically change his stance with the questions proposed.
3.2 Interactive Storytelling

To construct a lie it is necessary to create and maintain a parallel and false story, as described in 2.1. Performing this tasks requires the liar to shape a narrative and the field of virtual agents that studies the generation of narratives is Interactive Storytelling (IS). The research done in this subject can be grouped in two different models: a plot-centered approach and a character-centered approach [33]. The rest one focus on creating an environment with a set of rules used to plan a story plot. On the other hand, a character-centered approach enables the participants of the story to decide their future. For the purpose of this work, we will only focus on some IS systems that follow a character-centered approach and are a helpful contribution to the development of our virtual agent's story module.

3.2.1 Agent’s Interaction in Virtual Storytelling

Despite not interacting with users, the prototype implemented in 2001 by Cavazza et al [33] used the interaction between several virtual characters, each one with a goal, to guide the plot of a story in a three dimensional environment. To develop this system, several issues had to be addressed, such as how to structure and represent actor's behaviours and how to update the story with the contributions of different characters. From a top-level perspective, characters have goals that can be described as groups of nested sub-goals ending with terminal actions, for example 3D animations.

Figure 3.2 describes part of a story using an AND/OR graph. It represents the action of Leave Workplace as an OR node, so only one of the two child nodes needs to be satisfied, either the Go Home node or the Go to John's Pub. Both actions are represented using AND nodes, this way all of their child nodes need to be satisfied. Assuming that the terminal nodes Walk to the parking lot, Drive Home and Park car are satisfiable, the action Go Home is also satisfiable, so Leave Workplace is also satisfied. This prototype uses a branched graph representing the narrative created a priori. Using a planner algorithm, a search process is carried out over that branched graph and it creates a state-space graph. Later, the search aims to get only the next action, because further interactions with different characters might influence the path to the main goal. Avoiding to look for actions that can be changed in the future, the prototype empowers the actors with re-planning capabilities between the execution of the selected action and the search for the next behaviour to be performed.

Apart from lacking on user interaction when creating the narrative, the work done by Cavazza et al. solved several problems related to storytelling. From the character point of view, he needs to adapt to the changes in the environment that surrounds him and making a big effort searching
for the optimal solution in the begin of the narrative seems to be a very optimistic approach when so many variants are involved. By endowing the actor with the option to change the path selected earlier to a better one according to the current state of the narrative, the prototype implemented models a storytelling environment responsive to changes and rich in story variability. Regarding the creation of a lying virtual agent, this work offers helpful insight about computational techniques used to adjust a story when changes on the agent’s surroundings occur, for example when answering a question, the lying agent might need to change some parts of the false story created in order to keep the consistency between it and the knowledge of the questioner.

3.2.2 The Virtual Storyteller: Story Creation by Intelligent Agents

The Virtual Storyteller [34] is a framework for story creation in a multi-agent system. The two relevant aspects of this project to our work are the automatic plot development by characters as intelligent agents and the control of those characters to get a well-structured plot. Similar to the work done by Cavazza et al [33] in 2001, the characters in the Virtual Storyteller have their own goals and control which actions they find the best to perform, but their behaviour, before executed, is judged by a director agent, an entity responsible to evaluate the impact of these actions on the plot. With those stakeholders, the process of creating a plot is defined as an hybrid approach, using a combination of character-centered and plot-centered models. This way, the director is able to guarantee that certain requirements are met in order to create a “good” plot, such as surprise, tragedy and drama.

The Virtual Storyteller does not propose a solution about the computational approach for planning and creating a plot, yet the hierarchical structure between agents and the well-defined role of each one in the creative process allows to generate a better and consistent narrative. Despite a lying virtual agent not looking forward to introduce surprise or drama in his fake story, having the supervision over the process of creating a story guarantees that certain criteria are met, like the internal and external consistency of a lie.
3.2.3 Narrative Mediation

In 2003, Riedl, Saretto, and Young introduced a new technique for interactive storytelling called narrative mediation [13]. In this alternative approach, the story is represented as a linear progression of events with anticipated user's actions, this way for every possible inconsistency the user makes to the story plan, there is an alternative plan to deal with that violation. This technique was implemented in Mimesis system, an architecture for building and maintaining interactive adaptive narratives. This approach is at least as powerful as an interactive narrative system that have acyclic branching stories [35].

Before the begin of an interactive session, the system generates a linear narrative with the ideal story, formally represented by a plan of actions connected by casual links, and, for every possible action from the user, it dynamically creates an alternative sideline to be used when an action that threatens the ideal storyline is chosen. The system defines the user's actions as one of three types: an action is consistent with the plan, an action is consistent with plan but its effects do not interact with the plan's remaining actions, or the action is not consistent with the plan and its effects threaten the ideal storyline, launching an exception. An exceptional action on the plan breaks a casual link needed to connect the begin of the ideal story to its end, so, to deal with this situation, Mimesis uses one of two mediation strategies. One strategy is accommodation, in which the exceptional action is executed and the narrative is re-planned, according to the users interaction, re-connecting the threatened causal connections. The other strategy, called intervention, inhibits the exceptional action chosen by the user and replaces it with a failure mode action, this type of action produces effects that are consistent with any causal structure of the plan.

The exceptional actions for the storyline previously generated, can be seen as the non-planned decisions humans make in their daily life and, when lying, improvisation might be needed instead of selecting a planned lie due to unexpected knowledge gain by the deceiver.

3.2.4 Game AI as Storytelling

When creating a game, designers focus on improving player's experience with pre-scripted plots. Riedl et al. proposed a system that, using artificial intelligence techniques, was able to adapt the game's narrative according to the player's decisions [14]. This approach was motivated by the lack of player's agency - the ability for the player to do whatever he or she wants at any time - and the limited customization of the game's play style.

The concept of narrative used in this work was defined by Prince in 1987 [36] and it says that a narrative is "the recounting of a sequence of events that have a continuant subject and constitute
a whole”. Based on this definition, their system plans different trajectories for a narrative and has to computationally search for the trajectory that maximizes the player’s experience. Assuming that the right internal side of the circle contains terminal states, figure 3.3 shows the possible branching, caused by different player’s decisions, deviating from the expected trajectory.

![Figure 3.3: State Space with multiple deviation from the expected trajectory](image)

There are two main problems when addressing the story generation system proposed by Riedl et al.: how to computationally model a narrative structure and how to process the construction and maintenance of the narrative based on user’s actions.

To solve the rest one, the system uses Partial-Order Causal Link (POCL) to represent a sequence of events that compose a narrative, adding the concept of author goals - an intermediate state, defined by the narrative designer, that must be contained in the story. The nodes of a POCL plan are operations that change the story’s world allowing future events to occur. Each operation has a set of preconditions that need to be satisfied in order to generate the event’s effects. There are three special concepts in this representation: the initial state (a representation of the story world as a set of logical propositions), the outcome (a description of how the story should be different when it is completed) and the author goals (an intermediate state that must be achieve in order to consider the story completed).

The solution for the second problem uses an extension of the Partial-Order Planning (POP) algorithm to include author goals and, instead of only choosing a valid sequence of operators that connects the initial state to the outcome, it will maximize the story value regarding the player’s experience. To accept author goals, the extended version of POP needs to prune all the solutions that do not satisfy all author goals. These cuts enforce the story to meet the game designer’s intent. Using plan structures to model the narrative, it is possible to identify points between two operations where the user’s action might change the story’s world. This way the algorithm can look into future changes and create an alternative narrative plan for each possible action. To construct these options, this version of POP assumes that the story will occur without threats to the narrative’s expected progress. But when an action that deviates the story from the conventional trajectory might occur, the algorithm assumes the worst case scenario and creates a new alternative narrative plan from that point on. This cycle is performed for every possible action until one
trajectory reaches the outcome.

Figure 3.4 has the representation of a story that contains one main branch, plan 0, with two author goals, \textit{a.goal 1} and \textit{a.goal 2}, and the alternative plans for the user’s actions that may threaten the main narrative, plan 1, 2 and 3.

This approach is worth mentioning because it is able to “predict” and deal with the user’s behaviour that deviates from the expected story trajectory. Contingency plans can also be used in our virtual agent to make him capable of dealing with new information told by the user that was already expected, avoiding to re-plan the whole story at the interaction’s time.
3.3 Memory architectures for agents

According to Brom and Lukavsky (2009), the concept behind an intelligent virtual agent is its believability, and to make them mimic a human-like behaviour they must have memory [37]. Agent’s capability to remember details about previous interactions has been a research topic addressed in multiple works, for instance to develop virtual companions capable of remember people’s actions [12], to create virtual teachers that help students learn subjects such as Geography based on past interactions [38] or virtual real estate agents that recall the customer preferences and previous conversations [39].

3.3.1 I Know What I Did Last Summer

Aiming to create an agent capable of sharing its beliefs and past experiences, in 2007, Dias et al. incorporated on FearNot’s synthetic characters the capability to recall and talk about their past experiences [11].

I Know What I Did Last Summer: Autobiographic Memory in Synthetic Characters adds to FAtiMA emotional agent architecture a component to link the emotions the agent had when an event was perceived with the event itself. In FAtiMA, personality and emotions take a central role in influence the characters’ behaviour however, for the purpose of our work, we will focus on the memory and structure used to store the agent’s events on the autobiographical memory.

FatiMA organizes its autobiographical memory in different episodes, each one containing actions or events about a particular location or time. Each episode has three main components: Abstract, Narrative and Evaluation.

![Figure 3.5: An event constructed on a episode where Luke hits John](image)

Figure 3.5 shows the representation of an event in John’s memory. The Narrative component of this event contains its time and location, the people and objects involved, and also details and feelings about the actions related. The Abstract is a summary of the actions with strongest
emotional impact and Evaluation contains the agent’s psychological interpretations of each cause-effect action in the episode.

3.3.2 MAY: my Memories Are Yours

MAY: my Memories Are Yours proposes a model for shared memories to be used in artificial companions that engage on a dialogue with users [12]. The virtual agent’s memory is based on the concept of autobiographical memory defined by Conway [10] stating that semantic knowledge can be specified in three layers: Lifetime Periods, General Events and Memory Line.

The model developed defined a shared memory by a tuple \( \langle L, G, E \rangle \). A Lifetime period, \( L \), can be a fixed lifetime period, referring an interval common to everyone’s life; or a specific time that only make sense to the speaker. The General Event contains 6 aspects: the action represented by the infinitive verb shared in the memory, who participated in the event, where and when did the event occurred, and other complements that does not fit in the other characteristics. It also contains a reference for the event itself, linking its underlying complement with the specific action and a link to another related General Event. The Memory Line has the context of the events. It is defined by the text that described the event, the instant or time interval of the event, the emotion state inferred from the text, an image that can be used to describe the event and a sound related to the event.

The model for shared memories is composed by three subgraphs, one for each layer previously described. Each graph has a specific view of the events but all levels are interconnected (See Figure 3.6). Between the three layers, the model’s main level is the General Event and the other two are used to contextualize and add personal details to it. To retrieve a memory, the virtual companion uses a structure that includes the reference for a specific event, its context and the complements to be retrieved.

To test if the agent was able to establish a relationship with the user using the proposed shared memory model revealed that the users considered friendship’s intimacy and companionship as more positively. The outcome helps validate the argument that the shared memories process contributes to the maintenance of a relationship [40].

3.4 Deceptive Virtual Agents

There is an extensive research in the virtual agent’s field and psychology regarding the detection and prevention of deception [6] [41], but, for the purpose of our work, we explore some implementations of deceiver agents that interact with other agents or humans using different forms of deception.
3.4.1 GOLEM

Castelfranchi built a multi-agent environment to research about social attitudes in agent's communication, mostly regarding cooperation and deception [8]. GOLEM is the world created to test these interactions between agents. It is based on the Blocks planning problem of AI. An agent has the goal to build a structure of blocks using small blocks, large blocks or both. Because each agent in this environment will have their own goals, conflicts between their objectives will appear. To successfully achieve his goal, an agent will have to compete and/or cooperate with other agents. Each agent has two types of preferences: task delegation and task adoption. For example, an hyper-cooperative agent will always assist others if he can while an hanger-on agent never acts by himself, in this example the task delegation preference is higher on the second type of agent but the task adoption preference is higher on the rest type of agent. In GOLEM agents are able to deceive in order to reach their objective.

There are three types of deception used for obtaining help: deception about capabilities, deception about personality, or deception about goals and plans. The example used in “Deceiving in GOLEM” [8] describes a scenario where Eve, an lazy agent, wants to delegate to Adam an action a that she is able to perform. Eve knows that Adam is a supplier agent, that is an agent with high adoption preferences but will only perform a task if he knows that the requester can not do it, and wants to make him believe that she is not able to perform the task in order to get his help. In this
case of deception there are three possible scenarios:

• Adam believes that Eve cannot do a: either Adam thinks that Eve is not capable of doing the task because never saw her doing it before, or because he thinks Eve’s personality is *delegating-by-necessity* thus she would ask for his help when she really can not perform the action. In both possibilities, Eve can exploit this error and used deception about her personality or about her capabilities to induce Adam to accept her request.

• Adam believes that Eve can do a: Eve has to convince Adam that she really need his help to accomplish her goal. She might say directly to Adam that can not perform the action by herself, or might try to change his beliefs about her personality, making him think she is has a *delegating-by-necessity* personality. Both these approaches involve lying to Adam deliberately and hope that he will change his opinion. If she is unsuccessful, she will possibly ask him again later to perform the action, hoping that he already review his beliefs about her.

• Adam has no beliefs about Eve’s capacity: Eve can use the same strategy mentioned in 2.

The multi-agent world created by Castelfranchi evaluates how agents use cooperative and deceptive interactions to achieve their goals. For the purpose of our work, it is worth analysing the deceptive interactions and their domain. The approach used to convince another agent to cooperate towards a common goal is to make an agent reason about what others think are his goals, personality, and capacity, and select the best option according to the result of that reasoning. He can either chose to directly convince the other agent to change his beliefs or to hope that, later, he will review his beliefs and behave differently. In both cases, the use of ToM is useful to make a better decision considering not only the agent’s beliefs but also what the target agent thinks about his beliefs.

### 3.4.2 MOUTH-OF-TRUTH

The system developed by Caroglio and de Rosis [7] uses the concept of uncertainty to determine the probability of a target believing one fact if some other fact is believed. This causal relation is modelled using Bayesian belief-nets [42]. The beliefs are the nodes of a network, where each one has a probability representing the belief’s certainty. When the target says something that can imply a belief, the value representing that belief’s probability is increased in the node. This way, the sender can make a statement with a degree of certainty according to the implied beliefs probability. This approach allow for the liar to have a representation of what the target’s beliefs are and create a lie with a degree of certainty.
3.4.3 Deception Planner

The Deception Planner is a modified version of the Local Search for Planning Graphs planner built by Christian and Young [9]. This algorithm creates a set of statements that can be told to a target agent with the goal of convincing him to do an action. The sender agent creates a plan that satisfies a set of desired states to achieve the target's goal. The Deception Planner eases the search for these plans by changing the initial state of the planning world. The agent adds lies to the initial state in order to increase the valid solution for the target's goal.

The Deception Planner was built with modularity in mind. It represents the world using first-order logic from Planning Domain Definition Language (PDDL) [43] and outputs, also, in first-order logic. This way, it can be used in contexts besides the virtual agents field. Despite the motivation for this work was to build an agent capable of convincing a target agent to perform some action using deception, the planning algorithm does not search for a solution that takes into account the target’s belief using ToM. This could be an improvement to consider if we were to build an agent capable of lying avoiding inconsistencies with the target’s knowledge.

3.4.4 Great Deceivers

In Great Deceivers, João Dias et al. propose a model for virtual agents able to deceive, using n-level ToM [44]. Their agent’s model had the ability to reason about other’s beliefs and actions as if they were their own. The evaluation of the agent's performance was carried out using the social game MIXER [45] with the integration of the n-Level ToM mechanism inside the character Werewolf, that needed to deliberately lie in order to be successful. The first test showed that the Werewolf performance increased when it was running a two-level of ToM instead of a single-level of ToM against Villagers with single-level of ToM. The second test showed that users perceived the second-level of ToM Werewolf as more intelligent than a single-level of ToM character.

The work described on this paper focused on the ToM aspect of deception, successfully proving that an agent with an higher level of ToM not only seems more intelligent to the people that interact with him, but also has a better performance when playing against an agent with a lower level of ToM. This paper is worth mentioning because supports the integration of ToM inside our deceiver virtual agent’s model, assessing that an agent could be a better liar if he takes into account what others think about his beliefs when creates and maintains a lie.
4
Agent’s Model

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Story Representation</td>
<td>31</td>
</tr>
<tr>
<td>4.2 Interaction</td>
<td>35</td>
</tr>
<tr>
<td>4.3 Architecture</td>
<td>38</td>
</tr>
</tbody>
</table>
The virtual agent we developed is able to talk about events regarding past experiences but also adjust them to match the user’s knowledge and, at the same time, diverge from the truth. Considering the research done, we designed a story representation that contains real episodes describing the agent’s life but also accommodate alternative versions of the original story.

To talk about the story it wants to share, the agent’s interaction with the environment, specifically users, follows a question-answering format: the user asks a question, the agent perceives and then generates an answer. When it receives a new question, the agent’s model register the information sent by the user and adjusts its story based on the user’s knowledge and the original story.

Considering the interaction and story’s state, the agent’s model selects a proper strategy to handle the perceived question, change the story and filter the answer sent. This model endows the agent with the capability of identifying compromising sections of its story, create parallel events to hide the real ones and select what information will be shared with the users.

4.1 Story Representation

The agent has to remember events about past experiences and, to do so, it must represent them in its memory, but it should also be able to modify and adjust its content.

A narrative can be seen as a sequence of events [36] and we used this definition as a foundation to our agent’s memory main element: the story. Conceptually, it describes a period of the agent’s lifetime and represents it with association between entities and events.

The agent’s basic memory fragment is an entity. It is defined by the tuple \(\langle \text{Identifier}, \text{Type}, \text{Value} \rangle\),

where:

- **Identifier** - each entity has an unique identifier that is used by events to associate entities with their fields;

- **Type** - has a classification used to group similar entities. The entity can have any type and there is not any constraint between this element and the element **Value**, except for the type **Time Period** where a complex value with a \textit{begin} value and \textit{end} value is required;

- **Value** - this element represents the entity’s value and can be of any format: integer, float or string. It can be a combination of multiple values.

Figure 4.1 shows a representation of an entity with the value \textit{Revolver} of type \textit{Gun}, and another entity describing a **Time Period** between 14:00 and 14:30, March 4th of 2016.

Entities are added to the agent’s domain before the interaction starts and, after stored in its memory, they can not be deleted, however they can be associated and unlinked with events.
With the entities specified in its domain, the agent can create an event. The agent’s representation of an event uses a structure similar to frames used in FrameNet project [46]. In FrameNet’s database, each semantic frame is a detailed description of an event, relation or entity and its related elements. Instead of using a representation that requires specific entities for each context, as the ones used in FrameNet, we chose to create a flexible frame that represents an event as an association of multiple entities, without many constrains, and an action. With this decision, the author can describe a narrative with a certain degree of freedom but it also holds the semantics needed for the agent to create alternative stories.

An event is formally defined by the tuple:

\[
\langle \text{Identifier}, \text{Real}, \text{Incriminatory}, \text{Action}, \text{Time}, \text{Location}, \text{Agent}, \text{Theme}, \text{Reason}, \text{Manner}\rangle
\]

Each element of an event is described as follows:

- **Identifier** - this element is an unique value identifying each event;

- **Real** - a flag that states if the event really occurred or is an alternative version;

- **Incriminatory** - this value measures how compromising the event is. It is used by the agent to decide if the event’s content should be hidden from the user or not;

- **Action** - this element contains the action related to the event. It is represented as an infinite verb;

- **Time** - represents the event’s time and it is a mandatory field. This element must have a reference to an entity of type Time Period or Time Instant;

- **Location** - this element is also mandatory and only contains a single reference to an entity. It describes the place where the event occurred;

- **Agent** - specifies all the people involved in the event. Can have multiple entities;

- **Theme** - similar to the homonym thematic role, it represents the entity that is the target of the Action or other entities related to the event. Can have multiple entities;
• **Reason** - this element describes the **Agent**'s motivation or purpose to perform this event's **Action**. Can have multiple values;

• **Manner** - describes any entities that could be used to carry out the event's **Action**, such as instruments, methods or ways. Can have multiple values.

The **Real** and **Incriminatory** elements diverge from the rest of the tuple because they don’t describe the event’s content but the event itself. If the events are created by the developer, these two fields are assigned by him/her when writing the story, but if they are generated by the agent to adjust the original story, the **Real** value will always be false and the **Incriminatory** value will be calculated in function of their entities' values.

The author can add events to the original story that are false in order to indirectly influence the agent's process to create an alternative event. This feature, similar to Game AI as Storytelling’s author goals [14], allows the author to guarantee that the parallel story follows a certain direction and has some entities.

The sentence “Peter stole the silver necklace from the Jewlery Shop between 10:00 and 10:15 of October 4th” is represented in a frame as follows:

\[[\text{Agent Peter}] \ST\ \text{silver necklace}] \[\text{Location Jewlery Shop}] \[\text{Time 04/10/2016 10 : 00}].\]

This sentence is represented using the agent’s event structure in Figure 4.2. In addition to the actual event's content, the event is real and has an incriminatory value of 100%.

![Figure 4.2: Event Example](image)

Despite the previous example representing the event as a collection of multiple entities, they actually do not contain any entity but references to them. The same entity can be referred by multiple events and, as a whole, the story is a collection of entities and events where the second associates instances of the first.
Figure 4.3 shows an example of a story with one event linking multiple entities available on the agent's domain.

![Story with one event example](image)

**Figure 4.3:** Story with one event example

The collection of events inside the agent's memory do not constitute its timeline: they do not have any order neither links connecting them sequentially. To retrieve the events' timeline, the agent can create a list ordered by each event's Time element, however some events may be in conflict since the agent holds two different stories in parallel, the original one and the alternative version created. Figure 4.4 represents a section of an agent's timeline with events conflicting because they happened in the same Time Period $k + 1$.

![Timeline with conflicting events](image)

**Figure 4.4:** Timeline with conflicting events

To avoid the conflicts present in Figure 4.4, the agent maintains two timelines. It can group the real story by selecting all the events with the element Real set to true but, to gather the alternative version, the agent retrieves a list maintained during the interaction with all the events that are consider as actives, this is to say the events shared with the user. Some of the agent's events
may be used by both stories, as Figure 4.5 shows.

Figure 4.5: Section of the stories’ timeline, with two shared events

The original story is specified before the interaction starts, its content will never change during the course of the conversation and its events are persistent, they will never be deleted. On the other hand, the parallel story will suffer several changes during the interaction: events will be created, modified and swapped.

Throughout the dialogue, the agent will gather information about the user’s knowledge. This beliefs are stored in a collection of tuples defined by: \((Entity, Event, Known)\), where:

- **Entity** - is the reference to the entity believed to be known by the user;
- **Event** - identifies the context related with the **Entity** shared during the interaction;
- **Known** - is a boolean value that states if this belief was identified by the agent in the user’s interaction.

This collection of beliefs enlarges the information the agent has about the interaction state and will be used when selecting the information it should avoid talking about.

To create a story, an author writes the real sequence of events, identifies the compromising information and then uploads it to the agent’s memory. During the dialogue, this version will not change, however the alternative story will be continuously modified and adjusted according to the user’s interaction.

### 4.2 Interaction

The agent interacts with users by questions and answers: it receives questions proposed by an interviewer and, after some reasoning, generates an answer. To develop a representation for
the questions and answers, we studied Virtual Suspect: Response Model [15] and selected some of the aspects proposed by their question-answering model. We also gathered from other virtual agents implementations with conversational characteristics the main information shared between the user and the agent [12].

4.2.1 Question

During the interaction, the user’s objective is to ask questions to the agent and, to achieve it, he/she must use a specific structure to represent a question, called query. This structure is defined by the tuple \((question \text{ type}, question \text{ focus}, question \text{ conditions})\), where:

- **Question Type** - is the category of the question proposed. It can be of type validation or information-gathering;
- **Question Focus** - this element identifies the event’s field to be retrieved. Similar to the functionality of interrogative adverbs ("Why?", "What?", "When?", "Where?");
- **Question Conditions** - is a list of constraints that restrict the events to be shared.

The **Question Type** describes the goal of the question proposed. If its type is validation, the user’s intention is to validate the question conditions included in the query, but if its type is information-gathering, the goal is to retrieve, within the events constrained by question conditions, the event’s field that match the **Question Focus**.

The **question conditions** element is a collection of conditions that must be met by each story’s event in order to consider as related to the question. Each condition is defined by the tuple \((Field, Operator, Value)\):

- **Field** - refers to the event’s element to be tested, such as Time or Agent;
- **Operator** is the comparator used to test the condition;
- **Value** is the value to be satisfied in the condition.

If the user questioning the agent wants to ask about events that occurred in Técnico, the condition would have “Location” in the element **Field**, the **Operator** would be “Equal” and the **Value** would contain “Técnico” (see Figure 4.6).

Each condition’s **Field** has a set of operators available. Currently, every **Field** has the operator “Equal” that is satisfied if the event’s value and the condition’s value are exactly the same. The **Field** Time has another operator called “Between” that requires two values, a **begin** and **end** value, and is satisfied if any instant of the event’s time period is between the two values specified in the condition.
For example, the question “When was Peter at home?” aims to get a specific time: using our story representation, this is the entity related with an event’s Time element. In addition to that, the question also restricts the agent, “Peter”, and the location, “Home”. The instance of this question of type information-gathering using our query definition is shown in Figure 4.7(a).

The other type of question, validation, does not contain a specific field to retrieve, so the query structure does not contain the Question Focus. If the user wants to confirm that Maria and John were in Lisbon, the question proposed to the agent would have the validation type and three conditions, two for the agents, Maria and John, and another to the Location, Lisbon (See Figure 4.7(b)).

(a) Query representing the question “When was Peter at home?”

(b) Query representing the question “Were Maria and John in Lisbon?”

Figure 4.7: Question Example

4.2.2 Answer

The answer generated by the agent to the question proposed is represented by a query result. This structure contains 3 elements and is defined by the tuple: \((query, result, extra)\), where:

- **Query** - is the structure used to generate the answer;

- **Result** - contains the answer to the question. This element’s format depends on the Query’s question type;
• **Extra** - is a list of additional details about the answer created during the answer generation.

The *query result* can be an answer to a *validation* or an *information-gathering* question. In the first case, *Result* only contains a single value, either negative or positive, and if it is of the second type, this element contains a list of multiple results, each one defined by the tuple \(\langle \text{value}, \text{cardinality} \rangle\), where the *value* contains the entities that satisfy the *query* requirements, and the *cardinality* describes the number of times the *Value* appears in the agent’s story according to question’s restrictions.

In addition to the original *query* and the *result*, the *query result* structure has the element **Extra**. It is a collection of pairs \(\langle \text{property}, \text{value} \rangle\) that contain additional details about the agent’s answer and will be shared with the user. These properties can be used by the system responsible to show the answer. For example, a Natural Language Generation system or an animated Avatar may use this information to express cues to deception during the interaction.

The questions on Figure 4.8 show the answers to the two types of questions presented in Section 4.2.1. To the *information-gathering* question “When was Peter at home?”, the generated *query result* has two Time Period values. On the other hand, since the question “Were Maria and John in Lisbon” is a *validation* one, it only contains a value on the Result.

(a) Answer’s representation to “When was Peter at home?”

(b) Answer’s representation to “Were Maria and John in Lisbon?”

**Figure 4.8:** Answers Example

### 4.3 Architecture

The virtual agent’s architecture is vertically layered with two-pass control [1]. It has three layers and a Query Engine that will process the question, gather the events satisfying its requirements and then generates an answer. In this type of architecture, the information flows up the hierarchy, is processed and modified by each layer, reaches the Query Engine that creates an answer, and then flows back down until it is shared with the user. Parallel to the components previously
referred, there is a Knowledge Base holding the agent’s story representation that interacts with the different layers as the information flows.

The abstraction of behaviour between layers creates a functionality separation that increases the system’s modularity. The virtual agent’s model proposed has three layers: one for the Theory of Mind, another to select a strategy and finally the Story Adjustment layer. However, to add more functionality to the user, the architecture can receive more layers and stack them between the ones already existing. Figure 4.9 show a representation of the agent's architecture.

![Figure 4.9: The virtual agent’s vertically layered two-pass architecture. By definition [1] the architecture is represented vertically, however, for space reasons, it is drawn horizontal with the bottom layer on the left.](image)

As already explained, each layer encapsulates its own functionality but they all receive the same information: on the first pass they handle the question and, during the second pass, they process the answer to be shared. The layers’ responsibilities are described as follows:

- **Theory of Mind** - this layer identifies the entities shared in the questions and answers, and then tags them as known in the Knowledge Base.

- **Strategy Selection** - it analyses the state of the interaction, the agents beliefs about the user’s knowledge, the original and parallel story, and the question asked, to select from the strategies available the one that best suits the situation.

- **Story Adjustment** - this layer is responsible to adjust the agent’s false story and filter the answers’ content if they reveal compromising information.

With exception of the Strategy Selection Layer, all the layers have one procedure for each pass. Algorithm 1 shows the procedure calls for all the layers, since the agent perceives a new question to the moment it shares the respective answer with the user.

Each of the calls that make the agent’s cycle has a specific behaviour, and, to better understand each one, the following sections describe a deeper analysis of the procedures.
Algorithm 1 Algorithm representing the agent’s cycle

1: procedure AGENTCYCLE(question)
2: TheoryOfMindLayer.Update(question)
3: strategy ← StrategySelectionLayer.SelectStrategy(question)
4: StoryAdjustmentLayer.AdjustStory(question, strategy)
5: answer ← QueryEngine.Process(query)
6: filteredAnswer ← StoryAdjustmentLayer.FilterAnswer(answer, strategy)
7: TheoryOfMind.Update(filteredAnswer)
8: return answer
9: end procedure

4.3.1 Update Theory Of Mind with Question

The procedure UPDATE(question) is invoked when the agent receives the question asked, represented in a query, as parameter, and will pass it through the next layer without modifying it. Although it does not change the question’s content, the entities referred will be used to update the agent’s beliefs about the user’s knowledge.

A question with many conditions is a very specific one, and when the agent receives these type of questions, the agent assumes the user has doubts about a particular section of its story and wants to retrieve more information about it. Following this assumption, the virtual agent can believe the user has information about the constraints inside the query’s conditions.

To register these beliefs, the agent updates its collection of entries regarding the information already shared between both parties of the interaction. Algorithm 2 describes the process used in this layer to handle the question and update the information the user shared. First it checks if the question’s conditions are enough to consider as user’s knowledge. Then it retrieves from the Knowledge Base the events that satisfy the question restrictions and all the entities shared in the question. The agent then iterates all the events extracted and checks if they contain any shared entity. If so, a new belief is created. In the end the procedure returns the question as it was received.

4.3.2 Select Strategy

To select the strategy to be used by the agent, the procedure SELECTSTRATEGY(question) takes into account different aspects, such as the beliefs about the user’s knowledge, how comprising the information to be shared is or the number of alternative versions to the current story. After evaluating the different alternative strategies, it returns the selected strategy and method, defined by the pair (strategy, name), where:

- **Strategy** - is the name of the category that groups the similar method. It can have the value of *Don’t Lie* or *Lie*. 


Algorithm 2 Procedure to update the Theory of Mind using the question asked

1: procedure THEORYOFMINDLayer.UPDATE(question)
2:   if CountCondition(question) > threshold
3:      events ← searchRelatedEvents(knowledgeBase, question)
4:      sharedEntities ← extractEntities(question)
5:      for each event ∈ events do
6:         for each sharedEntity ∈ sharedEntities do
7:            if sharedEntity ∈ event
8:               AddBelief(event, entity)
9:         end if
10:     end for
11: end for
12: end if
13: return question
14: end procedure

- Method - is the name of the method that will be applied to adjust the story.

The methods available are grouped in two strategies: Lie and Don’t Lie. The last one groups two methods: None and Hide, and the strategy Lie holds three methods: Duplicate Event, Adjust Event and Adjust Entity. Table 4.1 has the methods available grouped by strategies, the conditions required to select them and the effects each one has.

4.3.3 Adjust Story

The layer, from where the procedure ADJUSTSTORY(question, strategy) is invoked, is responsible for adjusting the story in the Knowledge Base but also its segments in the answer ready to be shared. This procedure’s objective is to apply the strategy selected on the story representation stored in the Knowledge Base, based on the agent’s beliefs and the story’s content.

4.3.4 Filter Answer

The answer received in the procedure FILTERANSWER(answer) as parameter is passed by the Query Engine, after querying the Knowledge Base with the question handled before. Similar to the method AdjustStory, also called from this layer, FilterAnswer applies the strategy already selected, however only one method an effect that directly modifies the answer generated: the Hide method for the Don’t Lie strategy. It removes compromising information from the answer before it is shared with the user, and, to do so, the agent searches its story to verify if the entities passed in the answer are compromising. If so, those entities are removed from the answer.
<table>
<thead>
<tr>
<th>Strategy</th>
<th>Method</th>
<th>Condition</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Don’t Lie</td>
<td>None</td>
<td>The events that satisfy the question's conditions do not contain compromising information</td>
<td>Do not modified the story or the answer</td>
</tr>
<tr>
<td></td>
<td>Hide</td>
<td>The agent can not create an alternative story but the information to be shared in the answer should be kept hidden</td>
<td>Do not modified the story but removes from the answer compromising information</td>
</tr>
<tr>
<td>Lie</td>
<td>Duplicate Event</td>
<td>The information to be shared is compromising, the agent can adjust the story and the alternative events for these section of the story were not created yet</td>
<td>Copies the incriminatory events and replaces its elements that are incriminatory and not believed to be known to the user by uncompromising entities.</td>
</tr>
<tr>
<td></td>
<td>Adjust Event</td>
<td>The information to be shared is compromising, an alternative version of the relative events is already created but previously hidden entities are now believed to be known by the user</td>
<td>Modifies in the relevant events that contain incriminatory entities, the elements that are in conflict with the agent’s beliefs about the user’s knowledge with uncompromising entities.</td>
</tr>
<tr>
<td></td>
<td>Adjust Entity</td>
<td>If the information to be shared is compromising, there is only one entity that satisfies the question's requirements and it occurs in multiple events of the story</td>
<td>Changes all the references to the incriminatory entity by an uncompromising similar entity.</td>
</tr>
</tbody>
</table>

Table 4.1: Available methods to adjust the story (grouped by strategies)

4.3.5 Update Theory Of Mind with Answer

Before sharing the answer already filtered to the user, the agents registers the information it contains as known by the user. The procedure $\text{UPDATE}(answer)$ behaves similarly to Algorithm 2 but instead of receiving a query as parameter, it is passed a query result representing the modified answer.

The answer received by the procedure is not modified in this layer and, since it is the last step of the agent’s cycle, it is sent as the agents output.

4.3.6 Retrieve Similar Entities

The procedure $\text{RetrieveSimilarEntities}(event, entity)$ is called by all the methods in the strategy Lie every time they need to replace an incriminatory entity with an uncompromising one.
This method must return an ordered list of uncompromising entities that are similar to the entity to be replaced. To rank the entities in the agent’s domain, three aspects are considered:

- **Entity’s Type** - an entity has a higher rank if its type is the same as the one from the entity to be replaced;
- **Incriminatory Value** - if an entity has a lower incriminatory value, it is ranked higher;
- **Entity’s Context** - an entity that is referred in events with the same elements that the entity to be replaced has a higher rank.

The method `RetrieveSimilarEntities` receives an `entityToReplace` and a `sourceEvent` to contextualize the first. To retrieve similar entities to the `entityToReplace`, the agent iterates all the entities of its domain, and, for each one, tests its type and calls the method `EvaluateContextSimilarity` with two parameters, the entity being currently iterated and the `sourceEvent`. This method returns the percentage of common elements each event that refers the entity currently iterated has with the `sourceEvent`. Half of the similarity factor takes into account if the types match and the other half the returned value from `EvaluateContextSimilarity`. If the similarity is greater than 0, the current Entity is added to the `similarEntities` list alongside the product of its similarity and the inverse of its incriminatory (See Algorithm 3).

Algorithm 3 Procedure to retrieve a list entities similar to another entity

```plaintext
1: procedure RETRIEVE SIMILAR ENTITIES(entityToReplace, sourceEvent)
2: for each entity ∈ KnowledgeBase.entities do
3:     if entity ≠ entityToReplace then
4:         sameType ← entity.type = entityToReplace.entity
5:         contextSimilarity ← EvaluateContextSimilarity(entity, sourceEvent)
6:         similarity ← (sameType * 0.5) + (contextSimilarity * 0.5)
7:         if similarity > 0 then
8:             similarEntities.Add((entity, similarity * (1 − entity.incriminatory)))
9:         end if
10:     end if
11: end for
12: return similarEntities.OrderBy(similarity)
13: end procedure
```

4.3.7 Process Query

The Query Engine receives the question used across the three layers in the first pass, queries the Knowledge Base and sends an answer with the corresponding results to the second pass.
QUERYENGINE.PROCESS(question) is the procedure responsible to search in the Knowledge Base the events that satisfy the Question Conditions and, if the Question Type is information-gathering, retrieves a collection of entities that match the event's field specified on query.focus. On the other hand, if the Question Type is validation, it returns a negative answer if no events were found and a positive answer otherwise (See Algorithm 4).

Algorithm 4 Procedure to process the Knowledge Base using the question’s query

```plaintext
1: procedure QUERYENGINE.PROCESS(question)
2:    matchedEvents ← searchRelatedEvents(knowledgeBase, question)
3:    if question.type = validation then
4:       answer ← matchedEvents.Count > 0
5:    else
6:       for each event ∈ matchedEvents do
7:          focusEntity ← extractField(event, question.focus)
8:          relatedEntities.add(focusEntity)
9:       end for
10:      answer ← relatedEntities
11:    end if
12:    return answer
13: end procedure
```
## Use Case: a Virtual Suspect

### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Police Protocols</td>
<td>47</td>
</tr>
<tr>
<td>5.2 Implementation</td>
<td>48</td>
</tr>
<tr>
<td>5.3 Natural Language Generation</td>
<td>53</td>
</tr>
<tr>
<td>5.4 Interaction’s Interface</td>
<td>53</td>
</tr>
<tr>
<td>5.5 Who is Peter Barker?</td>
<td>57</td>
</tr>
</tbody>
</table>
The agent’s model proposed can answer questions asked by an user and, during the interaction, it adapts its real story to the user’s knowledge while hiding compromising information about its past events. This functionality can be used to implement a virtual agent that wants to mislead an interviewer looking to discover the agent’s story, such as in a police interview. This interaction is an example of an uncooperative dialogue where the police intention is to discover information the suspect do not want to share.

To test our agent’s performance we used the environment described before, where the user plays the role of a police officer interviewing a virtual suspect that implements the agent’s model proposed. The interviewer goal is to validate the information he/she has about the suspect being questioned and the related case, but also to discover new facts regarding the crime investigated.

To create an interaction similar to a real police interview we researched how they are conducted, what police protocols are involved and the best practices used by the investigators to extract the most about the suspect’s actions, not only to collect facts regarding the case but also to gather information about the suspect’s stance about the crime being investigated. With the strategies studied, we could replicate some of these methods to occur during the interaction to create a more appealing interaction.

Section 5.2 describes the technology used to implement the virtual agent’s model and its development process. The following sections elaborate on the mechanism used to convert the agent’s answer representations to natural language and on the different iterations of the interface used by the user to interact with the virtual suspect.

To implement the Virtual Suspect, a crime narrative about a jewelry shop robbery was written, the suspect’s point of view was extracted and then translated to match the story representation. Section 5.5 describes the process used to create the agent’s story and its refinements.

5.1 Police Protocols

A police interview is a delicate environment when seen from both parties of the interaction. The suspect usually has compromising information he/she do not want to share and the policeman needs to control and guide the dialogue such that the suspect exposes sensible details about his past action but, at the same time, the interviewer must guarantee his/her cooperation. To master these skills, polices officers have to train their capabilities using simulated interviews. During their training, they are taught strategies and guidelines that, if followed, the outcome of an interview would benefit more the police than the suspect.

During their instruction, police trainees learn they should previously prepare the interview and take into the interrogation room notes with details about the case, such as information gathered
from physical evidences or the suspect's register. The interviewer is advised to also prepare the flow of the interview, defining the topics he/she must talk about. To create a similar experience for the user interacting with the Virtual Suspect as the real police interviews, we present to the interviewer a notebook with information about the case that he could access before, during and after the interview.

Since policeman, namely Portuguese Police investigators, can not say false information or evidence to the suspect, the Virtual Suspect can assure that all the questions proposed and the information they implicitly contain are true. This assumption allows the agent’s implementation to consider all the information the questions contain as known by the user.

When implementing the Virtual Suspect, the method \textit{Hide} for the strategy \textit{Don't Lie} was removed based on two motives. If the intelligent agent's implementation decided to remove from its answers the information that was compromising, from the point of view of the user the interaction would be frustrating since he/she would never gather new real information. The other reason relates to the police interview approach chosen by the suspects: they are advised to cooperate and choosing to stop answering does not benefit them.

The interviewer is taught to ask open and generic questions in the begin of the conversation, also called information gathering phase, and only later confront the suspect with information he/she already had. By doing this two separate phases, the officer can expose the lies and pressure the suspect. The interaction environment created took this information in consideration: the questions the user could make were both of validation and information-gathering type.

### 5.2 Implementation

To create the Virtual Suspect and the environment to test its performance when questioned by an user, we divided the work to be done in two phases: the first one to implement the Virtual Suspect and the second to develop the Interface for the interaction.

The Virtual Suspect was implemented using C# and the messages exchanged with it, answers or questions, were structured in XML. The component that stores most of the agent's memory is the Knowledge Base. It holds the information about the agent's story, the alternative version of the events and its beliefs.

#### 5.2.1 Knowledge Base

The first step the agent does when starting is loading its story from a XML file into the Knowledge Base. The story’s file contains a list of entities with the multiple events that associate these entities and actions to form the agent's story (See Listing A.1).
Internally, each entity is implemented by an `EntityNode` and an event by an `EventNode`. Although these Nodes contain almost the same information for entities and events as the ones described in Story Representation (Section 4.1), the `EventNode` instead of having an action, it has a reference to an `ActionNode`. This modification, helps retrieving all the events associated with an action. The Virtual Suspect's story is represented as a graph with several instances of the Nodes described earlier with connections between them. In the agent's graph there are four different groups, one for each type of nodes, and a fourth to hold the events in the story shared with the user (See Figure 5.1).

Figure 5.1: Agent’s graph with multiple nodes associated in events

This graph does not explicitly group the agent’s original story, however it can be retrieved by selecting the events that are marked as real. Noteworthy that the events inside the shared story can belong to the original story too.

The story’s author can change the agent’s original story by modifying or replacing its content without using any code since it is stored in a file and loaded into the Knowledge Base before the interaction begins. Nonetheless, despite the original story being the same throughout the whole interaction, the agent also has a parallel story that will be continuously updated by the agent’s different layers.

The Knowledge Base has an API used by the layers in the agent’s vertically layered architecture to add, modify, remove and search its content. The interface offers the following methods:

- **Create new Entity** - This method receives as parameter an `EntityDto` that contains the entity’s type and value. It creates a new `EntityNode` with elements passed and the next available identifier.

- **Create new Action** - This method receives as parameter an `ActionDto` that contains the
action’s name. It creates a new ActionNode with action’s name and the next available identifier.

- **Create new Event** - This method receives as parameter an EventDto. This structure has the information about the event to be created: if it is real or not, its incriminatory value, the action associated, and a list of pairs that represent associations between the event’s fields and available entities. This list must have one association for the Time field and another for the Location Field. A new EventNode with the event’s details is created but it is not included with the shared story.

- **Replace Event** - This method receives two EventNode. One must be in the shared story and the other should be free. This method’s effect unlinks one event from the shared story and associates the other.

- **Filter Events** - This method receives a list of Filters (See Section 5.2.2). It returns a list with the events of the shared story that are satisfied by all the Filters passed.

- **Retrieve Original Story** - This method returns a list with all the events from the agent’s original story. The list is ordered by time.

- **Retrieve Shared Story** - This method returns a list with all the events from the agent’s shared story. The list is ordered by time.

### 5.2.2 Query Engine

The component that processes the user’s question and generates an answer from it is the Query Engine. As the name states, its functionality is to retrieve from the agent’s story the events and entities that satisfy the query restrictions. To do so, it interacts with the Knowledge Base, mainly with the objective of reading and searching its content.

To search the Virtual Suspect’s Knowledge Base, the Query Engine parses the elements of the query representing the question proposed and selects the proper filters to retrieve the corresponding entities and events. Each question condition in the query represents a comparison that must be satisfied to consider the event tested a match. This behaviour is encapsulated in Filters.

For instance, a question condition defined by the tuple \((location, equal, Lisbon)\) is then parsed into a Filter that is only satisfied when the location of the event being tested is equal to Lisbon. The Code Listing A.2 has the LocationEqualConditionFilter class implementation, one of the many Filters available. Each Filter implements the interface IConditionFilter and must define a constructor that receives the condition’s value and stores to later be tested when the predicate returned from the method CreatePredicate is tested.
As the Filter implements a question condition, the FocusPredicate implements the question focus. Similar to the Filter class, there is a FocusPredicate for each one of the fields that can be retrieved. They all implement the interface IFocusPredicate and Code Listing A.3 has the implementation of one of those classes, the TimeFocusPredicates that is used to extract this Field Time from the event tested.

Following a modular approach, as the one used on the rest of the Virtual Suspect, these abstractions of behaviour for the implementation of the question focus and question conditions create an easy and flexible way of increasing the range of possible operators to be used in the questions asked by the user.

The Filters and FocusPredicates are created as a result of parsing the XML question asked by the user. The comparisons they encapsulate are used by the Query Engine when it processes the question received.

The method that generates an answer based on a question is called ProcessQuery. For querying the Knowledge Base in order to retrieve the entities and events that satisfy the question specification, the method ProcessQuery calls the method FilterEvents and passes as parameter the Filters associated with the question conditions. This method returns a list of the events that satisfy the conditions sent and then, according to the question type, it generates the respective answers. When the type is information-gathering, the FocusPredicates are used to extract the fields referred in the question focus and, after that, they are counted and their cardinality is stored in the query result structure representing the answer to the question proposed (See Code Listing A.4).

5.2.3 Layers

The agent has a two-pass vertically layered architecture which means each layer will handle the question after it was perceived and the answer before been outputted. The approach chosen to implement these layers also follows a modular design: each layer is implemented as an independent module that must define one method for the first pass and another for the second pass. Each layer has a constructor that receives a reference for the Virtual Suspect where it is used, in order to interact with the other components. Also, it must implement the interface ILayer that has the following methods:

- **HandleFirstPass** - This method receives a query representing a question from the previous layer or directly sent by the user. It returns a question to be passed to next layer or the Query Engine.

- **HandleSecondPass** - This method receives as parameter a query result representing an
answer sent by the previous layer or the Query Engine. It returns an answer to be passed on the next layer or outputted to the user.

The method invoked first receives a query representing a question asked by the user and returns a query too. Depending on the layers’ functionality the question returned might be or not the same as the one passed as argument. For instance, the Layer that selects a strategy, StrategySelectionLayer, must gather information from the Knowledge Base and the question proposed, then present a proper strategy. Although it uses the query received, it does not change it, so it is returned as it was received.

After the answer was generated by the Query Engine, the query result goes through all the layers again but, in this pass, does it in the reverse order. Similarly with the method for the first pass, this method’s return is the same as the type of the parameter received, but does not need to be the same answer. For example, the Layer responsible for changing the story shared with the user, StoryAdjustmentLayer, must filter the answer if the method selected is Hide. To do so, it modifies the answer received and returns a new query result with those changes.

As previously described, the layers are independent of each other and can be added or removed as intended: new layers can be added to enhance the agent’s functionality and others removed to test specific behaviours.

### 5.2.4 Assembly the Virtual Suspect

The components presented were developed individually, however they interact with each other to endow the virtual agent with its story adjustment capabilities. Although most of the question-answering mechanism is done using the Query Engine and the Knowledge Base, the deceptive behaviour of the Virtual Suspect is mostly contained within its layers.

The overall Virtual Suspect’s functionality does not only depends on the existent layers but also in its order. For instance, if the layer responsible to update the agent’s beliefs about the user’s knowledge, the TheoryOfMindLayer, was placed after the StrategySelectionLayer, when the agent was selecting the best strategy it would not take into account the most recent beliefs.

Following the Agent’s Model Architecture proposed in Section 4.3 the layers are ordered as follows: TheoryOfMindLayer, StrategySelectionLayer and StoryAdjustmentLayer. The Listing A.5 contains the code needed to initialize the agent with the three layers stacked as referred before.

The main procedure of the Virtual Suspect, called ProcessQuestion, receives a query and returns a query result. First, the layers are called in the order specified in the initialization and the resulted question is passed as argument on the Query Engine to query the Knowledge Base.
After that, a query result is retrieved and then passes through the layers in the reverse order. Finally, the last layer’s answer is returned in the procedure ProcessQuestion (See Listing A.6)

5.3 Natural Language Generation

If the answer generated was shown to the user as a XML message, its content would be hard to comprehend. To present the answer in a format the user would understand, a template based natural language generation system was developed to translate the answer representation in a structured message into a sentence. To create the response in natural language, the system uses the entities shared in the answer and the question that generated it.

For the questions of type validation, their answers only have two possible outcomes: “Yes” for a positive answer and “No” otherwise.

The natural language generation for answers of information-gathering questions demands a wider range of possible answers. The system implemented generates the answer in three phases: one for the subject, another for the verb and one for listing the entities in the answer.

The first segment of the answer contains the subject of the sentence, it depends on the number of agents the question had and, since the question always has the Virtual Suspect as agent, the number of possible combinations are reduced. The only two possible subjects are the personal pronouns “I”, for a single agent, and “We”, for multiple agents.

The second segment represents the action associated with the question asked. If the question associated with the answer does not have an action, the system uses the verb “to be”. However if the question has one, it uses the simple past tense of the verb in the action field.

The other segment contains a list with all the entities related with the answer and, because they are all of the same field (question focus), they can be presented sequentially. To concatenate the entities with the rest of the sentence, the system uses a library of prepositions associated with each action.

For example, to the question “Where were you on March 4th between 12:00 and 18:00?” and, assuming the question generated contained the entities Home with cardinality two and Mall with cardinality one, the answer would be represented by: I + was + Home twice + Mall once and then it would be translated to “I was at Home twice and at the Mall once.”

5.4 Interaction’s Interface

The interaction’s interface between the user and the Virtual Suspect had two prototypes made. The first prototype allowed the user to ask any question to the Virtual Suspect but required him/her
to learn how to interact with the interface. The second interface restricted the questions the user could ask but offered a straightforward interaction. The technology used to implement the interfaces was Microsoft’s Windows Presentation Foundation.

5.4.1 Interaction’s Flow

Since the Virtual Suspect interacts with the user by questions and answers, the interface needed a mechanism to select questions, ask them to the agent and then present the answer generated. Figure 5.2 shows the flow of information for the user interaction with the Virtual Suspect.

![Interaction’s Flowchart](image)

The user may ask the same question to the Virtual Suspect the number of times he/she wants and, to help him/her know what questions were already asked, a log with all the previous interactions is available for the user to consult.

During the interview, the user can see a notebook with information about the case. It contains a list of notes where each one shows a sentence describing its content and the source of the information. The notebook is accessible throughout the whole interaction.

5.4.2 First Prototype

To develop an interface that allowed the flow information previously described, the first prototype designed had a complex interface that offered the user the opportunity to ask any question he/she wanted, following the question’s representation presented, the query. Figure 5.3 shows the interface of this prototype’s implementation.

The interface was composed by two areas: one for the answer and another for the question. The question is made in the Question Wizard where the user selects the fields to constrain in the left and then specifies their values to create the question conditions. When the question was
completed, the user clicks on the button "Ask Question" and the Virtual Suspect would process the question and generate an answer to be presented in the Answer area.

This prototype was tested by a focus group that had difficulties understanding how the questions were made without previous training. We also verified that the time the users needed to learn how to interact with the interface was longer than the pretended. From the feedback gathered, we reiterated the Interaction’s Interface and designed a simpler prototype.

5.4.3 Second Prototype

This prototype used the suggestions and concerns shared by the focus group to improve the user’s interaction with the Virtual Suspect. Using the previous iteration, the users did not know what questions to ask and they found where to start, it was difficult to ask the intended question. So we reconsidered the mechanism to create questions and discarded the approach used before.

In this new interface the user was not able to create new questions but could ask one from a list containing multiple options. This alternative method decreases the user’s freedom but creates an easier interaction. Figure 5.4 contains an initial sketch of the interface proposed for the second Prototype. This version has a list of questions, from which the user selects one to ask the Virtual Suspect, and the answer appears on the answer area. The interface also has some information about the suspect such as his/her name and age, a brief description of the case and its connection to the victim.
Figure 5.4: Second Prototype's interaction window sketch

Figure 5.5 shows this prototyped implement in WPF with the notebook windows and the logs of the question proposed by the user and the Virtual Suspect respective answers.

This version’s tests were performed with the same focus group and they revealed each interaction started sooner, the users knew were to start, and was faster, to ask a question it was just necessary to click the question button. However, as expected, in the end of the tests the users wished to ask more question that were not available.
5.4.4 Prototypes Remarks

The first prototype offered a higher range of question to ask since they are entirely created by the user but he/she needed to understand the question's representation and the time need to teach that structure was not valid. On the other hand, the second prototype restricts the number of questions the user can make but eases the interaction and reduces the overall time to make each question.

Despite renewing the first prototype and developing a second interface, the first iteration was not discarded. The simpler approach was best suited for the User Studies but the first one was used to develop and debug the Virtual Suspect's implementation since we had the freedom to analyse its response to any question we wanted to test.

5.5 Who is Peter Barker?

The Virtual Suspect is an implementation of the virtual agent's model proposed but without a crime to talk about, it would not be able to share its events about its past story, or the created parallel story.

To show the agent's capabilities and test its performance, we created two different plots involving different number of suspects and different complexities:

Case 1 Peter Barker is the only suspect of a jewelry shop robbery. A silver necklace was stolen and the police has several connections between him and the man trying to sell the stolen good.

Case 2 Lewis Murch is a 80 years old man near his death. He is currently involved in a relationship with a 56 years old woman. Mr. Murch sons have a rough past with their father and they only want his heritage, but with this new woman nearby, they might lost everything. Each one plans to kill their father without the other's knowledge. They both try to kill Mr. Murch however, only one is successful. They are brought to the police to be questioned about their father's death.

After conducting tests with the same focus group as before, we found that the story about Lewis Murch's Death, more complex than the first one, was too difficult for an inexperienced user. Case 1 offered a good challenge for the users and, at the same time, allowed us to test the agent's deceiving capabilities.

Based on the first case narrative, we created the main suspect's perspective of the events. We recreated Peter Baker's life around the actual robbery to contextualize it, one day before and another day after. With the descriptions of the events related with these three days of the agent's
life, we extracted the relevant entities and assigned them types as the story’s representations specifies. Then we associated the entities into events to create the real sequence of events of the Virtual Suspect’s story. In addition to factual information about the suspect’s story we needed to tag each one of the events with an incriminatory value between 0 and 100 to describe how compromising the event was.

Before the interaction starts, the XML file containing Peter Barker’s story was parsed by the Virtual Suspect Knowledge Base, then we tested the Virtual Suspect’s answer to the questions asked. After that we modified the suspect’s story by adding one event with an alternative version to a section of its life. This addition indirectly influenced the agent when creating a new event to replace that section of the Virtual Suspect’s story.

Peter Barker’s story about the three days around the robbery contains a total of 50 entities with 15 different types. The real agent’s story has 23 events with 14 different actions associated. The Virtual Suspect playing the role of Peter Barker on a police interview environment offered an interaction with the desired duration and complexity.
Demonstration
To better understand what the agent’s model does during the interaction and to verify that the Virtual Suspect’s implements it properly, the following section demonstrates its behaviour as well as interactions between its components from the moment a new question is asked until an answer is outputted. For the question proposed, we will cover the following aspects of the interaction:

- The query representation of the question;
- How the Theory of Mind Layer processes the question and updates its beliefs about the user’s knowledge;
- How the Strategy Selection Layer determines what is the best strategy to select;
- How the Story Adjustment Layer applies the strategy select by creating and changing events on the shared story;
- The Process Query with the Knowledge Base to retrieve the relevant entities;
- The query result representing the agent’s answer.

For example, consider the question “Did you go to the jewelry store on March 3rd or 4th?”. Its intention is to validate that Peter Barker was on the jewelry store between the begin of March 3rd and the end of March 4th. When formalized as a query, this question representation is shown in Figure 6.1.

![Figure 6.1: Query representation for the question “Did you go to the jewelry store on March 3rd or 4th?”](image)

When the Theory of Mind updates the Virtual Suspect’s beliefs about the user’s knowledge based on the question asked, it first verifies if the question conditions are enough to consider the event as known by the user. In this particular case, the query has three conditions, which is not enough to believe the user knows what happened in that particular moment. So the Theory of Mind Layer does not update the virtual agent’s beliefs. However, if another condition was present, such as the Action field, the agent would consider it as specific enough to believe the user knows something about what happened.
The next procedure to be invoked is the first pass method of the Strategy Selection Layer. First, it gets the events that satisfy the *question conditions* and, for each one of the three retrieved, verifies if they are compromising. Because the events retrieved are incriminatory, it selects the strategy *Lie* and, since there is no alternative events created already, the method to apply is the *Duplicate Event*.

With the strategy selected, the Story Adjustment Layer receives the question and must apply the method previously determined. The method is applied to all the events that satisfy the *query condition*. The events on the original story before the adjustment is performed are represented in figure 6.2.

**Figure 6.2:** Story section with events satisfied by the question “Did you go to the jewelry store on March 3rd or 4th?”

All the events retrieved are incriminatory and the strategy selected is Duplicate Event so it creates a copy for all the events and replaces the incriminatory entities with uncompromising ones. For instance, in the event with the identifier 5, the three fields that need to be replaced are the *Action*, *Location* and *Reason*. The Time field does not change because it determines the event’s position in the story and the Agent field is kept the same since the Virtual Suspect is the only participant. To replace each field, the Virtual Suspect invokes the method *RetrieveSimilarEntities* and finds the best alternative entity, and for this particular event the *Action* is kept the same but the *Location* is changed to “Peter’s Home” and the *Reason* to “Look for a watch to buy” (Figure 6.3).

**Figure 6.3:** Duplicated events that replace the compromising section of the Real Story
After going through the three layers, the question reaches the Query Engine and an answer is finally produced. The procedure \textit{Process} that receives the question to query the Knowledge Base is invoked but, now, instead of having three events that match the \textit{question conditions}, there is not any event that satisfies these restrictions, since the shared story was adjusted and the compromising events replaced. The \textit{query result} representing the answer generated contains the value \textit{false} (See Figure 6.4).

![Figure 6.4: Query result representing the answer](image)

Before the question was asked, the Virtual Suspect's original story and shared story had the same events and were coincident. But after the modifications, new events were created and the shared story was adjusted. Figure 6.5 represent both timelines before and after the question was asked.

![Figure 6.5: Virtual Suspect's stories timelines before and after the question was proposed](image)

The example presented before describes each step of the virtual agent's process to answer a question and, at the same time, create alternative events to hide compromising sections of its real story. Despite showing that Virtual Suspect behaves as expected we conducted experiments with users to test the agent's behaviour to different sequences of question.
Evaluation

Contents

7.1 Sample ......................................................... 67
7.2 Measurements ................................................. 67
7.3 Methodology and Procedures ............................... 69
7.4 Results ......................................................... 70
7.5 Empirical Results ............................................. 72
7.6 Discussion ...................................................... 73

65
In order to analyse how the Virtual Suspect behaves with different interviewers, we designed an experiment where the users would interact with the virtual agent implemented in a scenario that simulates a police interview. The user's studies goal was to validate our hypothesis by showing that the Virtual Suspect was capable of creating and maintaining a parallel story that diverged from the real events when asked different sequences of questions and, consequently, misleading the interviewer.

In addition to verify if the participant was fooled by the Virtual Suspect, we also evaluated the user's sequence of questions, using the rules and guidelines specified in police protocols, and the agent's performance, using metrics related to number and lie characteristics, to verify if a sequence of questions with a lower classification, related with bad interviewing skills, would allow the agent to lie more successfully and if question's sequences with a higher classification decreased the Virtual Suspect's efficiency.

The duration of the interviews was intended to be between 8 and 12 minutes and the tests initially done with the focus group revealed that the Second Prototype implemented offered the best interface and mechanism to match the time requirements, however, the duration of a real police interview can take several minutes or even hours. This difference could create a gap between the simulated interaction and the real interview but, to create an appealing and interesting experiment for the users, we had to reduce the dialogue duration.

7.1 Sample

In our user's studies, a total of 31 participated, where 24 were male and 7 female, their ages ranging from 18 to 30 with a Mean of 22.74 and a Standard Deviation equal to 1.932. When asked "How familiarized with police interviewing techniques are you?" using a 5 points Likert scale, where 1 was "Not Familiarized" and 5 was "Very Familiarized", the answer's mean value was 2.36 with a standard deviation of 1.02, however none of the participants in the story had training related to police interrogations or cognitive interviews.

Since no participant had any instruction on police interviews, there were no outliers in the sample, so all the user's interaction results were considered valid for analysing. All participants signed a consent to allow the interaction's content to be used in this study.

7.2 Measurements

To retrieve the data necessary from each interaction, we used three measurements:
• A questionnaire to collect data about the participant’s experience with this type of environment and their perceptions of the Virtual Suspect’s behaviour regarding its past actions and their decisions.

• A log containing the user’s interaction with the Virtual Suspect, containing the questions and answers exchanged, the time between the interaction begin and the first question, and the whole interaction’s time.

• A log describing the Virtual Agent’s story adjustments done during the interaction.

**Questionnaire**

The questionnaire presented to the users is entirely available in Appendix B. This questionnaire is divided in two sections, one to be filled before the interaction and another when concluded.

The first set of questions focused on gathering information about the participant's experience with police interviews and their perspective about them. The questions proposed in this section of the questionnaire were:

• Q1.1 “How knowledgeable about police interviewing techniques are you?”

• Q1.2 “Did you ever had training on police interrogations or cognitive interviews?”

• Q1.3 “How important do you think the emotional aspect is during a police interview?”

• Q1.4 “How important do you think the reasoning side is during a police interview?”

Q1.1, Q1.3 and Q1.4 used 5 point Likert scales to collect the user's answers.

The second part of the questionnaire was presented after the interaction with the Virtual Suspect was completed. This section had questions about the crime investigation and the participant’s perception of the Virtual Suspect’s behaviour.

• Q2.1 “Did Peter Barker, the suspect, sell the necklace to John Frey?”

• Q2.2 “How confident are you of your previous answer?”

• Q2.3 “Did the suspect rob the jewelery store?”

• Q2.4 “How confident are you of your previous answer?”

• Q2.5 “Who was an accomplice of Peter Barker?”

• Q2.6 “How confident are you of your previous answer?”

• Q2.7 “How much did you trust the suspect during the interview?”

• Q2.8 “Do you think the suspect lied during the interview?”
• Q2.9 "How efficient do you think your question's choice were?"
• Q2.10 "Regarding the jewelry robbery, how do you classify the suspect?"
• Q2.11 "How confident are you of your decision?"

**Interaction's Log**
The interaction's log contains the start and end time of each interaction. It also records the questions asked by the users and the respective answers of the Virtual Suspect.

**Virtual Suspect’s Log**
During the interactions, the Virtual Suspect's memory changed and this log records those modifications. The information saved in these records contains the events created throughout the story and the strategies selected for each answer.

### 7.3 Methodology and Procedures

Before the start of each study, the loggers were launched and the environment arranged to prepare the user's interaction.

When the participant arrived, he/she was invited to sit down, facing the monitor with the interface to be used for the interaction. Before starting the interview with the Virtual Suspect, the user was asked to fill the first section of the questionnaire and, when finished, to call the researcher.

With the first part of the questionnaire completed, we told the user what was the context of the interaction, a police interview, and what was his/her role, to be a police officer interviewing a suspect of a crime. Then, we described the interface's components and the mechanisms the user should use to interact with the Virtual Suspect. After that, we asked if the participant had any doubts about interface or the overall interaction, and answered them when necessary. Right before the participant started the interaction, we gave him a paper and a pen to write notes if needed and advised him to read carefully the notes about the case available on the screen and to plan its questions before making them.

While the interaction was taking place, the participant was alone in the room. When the interview's duration reached eight minutes we entered the room and checked the user's progress. The participants that had already concluded the interview were asked to fill the rest of the questionnaire. The other users that intended to ask more questions to the Virtual Suspect were left unsupervised and asked to call the researcher when finished.

After the study ended, the participants were encouraged to ask questions about the study, propose suggestions and share the overall feedback of the interaction. Most of the users at the
end of the interaction, politely demanded to know if the suspect had committed the crimes it was being accused.

In the end of the interaction, the participant were thanked for the collaboration in our study and gifted with a cinema coupon to be exchanged for a free ticket.

After the session ended and the participant left the room, we stored all the logs recorded and saved the questionnaire results in a digital format.

7.4 Results

With the first part of the questionnaire, we were able to retrieve information about our sample's experience with the environment they were going to face. All the participants of the study did not had any training in police interrogations or cognitive interviews. On the answer to the questions Q1.3 and Q1.4, the participants revealed to slightly value more the logical side of a police interview over its emotional aspect as presented in Table 7.1.

<table>
<thead>
<tr>
<th></th>
<th>Emotional Aspect</th>
<th>Logical Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>4.19</td>
<td>4.39</td>
</tr>
<tr>
<td>Std. Deviation</td>
<td>0.703</td>
<td>0.558</td>
</tr>
</tbody>
</table>

Table 7.1: Descriptive Statistics for Questions Q1.3 and Q1.4

The second part of the questionnaire’s purpose was to discover if the user thought the Virtual Suspect had committed the it was being accused of. The first four questions were used to evaluate if the participant tought the user was responsible for the crimes he was being accused. As seen on Table 7.2, regarding the jewelry robbery, 71% of the participants considered the Virtual Suspect guilty, while only 58% believed he sold the stolen necklace. According to questions Q2.2 and Q2.4, both answers had a confidence level with a mean value of 3.30 but the first had a Std. Deviation of 1.013 and of 1.279.

<table>
<thead>
<tr>
<th></th>
<th>Jewelery Robbery</th>
<th>Selling the Stolen Necklace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Percentage</td>
<td>29%</td>
<td>71%</td>
</tr>
<tr>
<td></td>
<td>42%</td>
<td>58%</td>
</tr>
</tbody>
</table>

Table 7.2: Answers to the questions Q2.1 and Q2.3

The next questions focused on discovering if the participant tought the Virtual Suspect had any accomplices on the crimes he was being accused. Figure 7.2 shows the percentage of participants that considered the other people in the story as accomplices of the Virtual Suspect.
Figure 7.1: Graph presenting the ratio of Yes and No answers for each accomplices

The answers to Q2.7 had a mean value of 2.03 and a Std. Deviation of 0.836 with more than half of the participants, 51%, choosing the answer 2 from a Likert Scale that ranged from 1 - “Do not trust” to 5 - “Fully trusted”. Question Q2.8’s objective was to survey if the user’s believed the agent lied during the interview. With three possible answers, 12.9% believed the Virtual Suspect lied always during the interaction, whereas 87.1% thought the virtual agent lied sometimes but also told the truth.

Figure 7.2: Graph presenting the percentages of each answer to question Q2.8

To measure the participant’s evaluation of their own sequence of questions, we asked question Q2.9. The answers had a mean value of 2.81 with a standard deviation of 0.910.

The last two questions, Q2.10 and Q2.11, were proposed to determined if the participant, with all the information gathered from the interview, was able to consider the Virtual Suspect guilty of the crimes accused. 87.1% of the users believed Peter Barker was guilty and their decision had a mean confidence value of 3.45 with a Std. Deviation of 1.028.
7.5 Empirical Results

In addition to the questionnaire results, we also analysed the logs retrieved from the interaction and the Virtual Suspects state.

The questions registered in the logs were evaluated based on their order according to a rule taught during police interviewing skills training: the interviewer should create a situation where the suspect would be forced to lie, such that, when confronted with information that contradicts its previous statements, the lies would be exposed.

With the questions available, during the interaction the participant could create three situations where it was possible to expose a lie. First the question asked should allow the Virtual Suspect to share a false state and then, the participant could expose its lie by confronting it with real evidences.

Based on the number of lie exposing moments, we divided the logs into two groups, one for logs with sequence of questions capable of revealing the Virtual Suspects lies, \( G_0 \), and the other for sequence of questions that never satisfied the rule previously presented, \( G_1 \). 16,1% of the questions’ sequence had no chance of exposing any lies while the remaining 83,9% created opportunities to expose the virtual agents lies.

On the other end, with the Virtual Suspect’s logs we retrieved the number of false events created and adjusted, as well as the number of real events shared during the interaction. Based on the two clusters identified for the Interaction’s log, we divided the records about the agent’s story in the same two groups and analysed each of the logs’ values with the results shown in Table 7.3.

<table>
<thead>
<tr>
<th>Percentage of Logs</th>
<th>Number of False Events Created</th>
<th>Number of False Events Adjusted</th>
<th>Number of Real Events Shared</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std. Deviation</td>
<td>Mean</td>
</tr>
<tr>
<td>( G_0 )</td>
<td>83,9%</td>
<td>6,35</td>
<td>0,689</td>
</tr>
<tr>
<td>( G_1 )</td>
<td>16,1%</td>
<td>5,80</td>
<td>0,837</td>
</tr>
</tbody>
</table>

Table 7.3: Results of the Virtual Suspects Logs divided in the two groups identified

The number of false events created by the Virtual Suspect when questioned with the groups of questions’ sequence that exposed its lies was 8,7% higher than the other set of logs, however, regarding the number of false events adjusted, for the questions of the logs in \( G_0 \), the Virtual Suspect adjusted 111,7% times more the number of adjustment when asked with the questions’ sequence of the logs of \( G_1 \). For the questions in \( G_1 \), the number of times the Virtual Suspect had to reveal real events was only 5,2% the total number of events shared when questioned with the
contents of the logs in $G_0$.

### 7.6 Discussion

Since all the participants had no training in police interviews, we could consider the sample had the same level of expertise regarding the interaction context. Despite only having a small difference, the participants believed the police interviews relied less in emotions than in its logical side, revealing an interesting perception by the users in the study.

The remaining questionnaire focused on the aspects related with the crime itself. After the interaction, most of the participants, 71%, considered the Virtual Suspect responsible for the jewelry store robbery, however only 58% believed that it sold the stolen necklace. It is also relevant to compare these results with Q2.10 - “Regarding the jewelry robbery, how do you classify the suspect?” - where most of the participants choose with a high degree of confidence, over 3,45, that Peter Barker was guilty of the crime accused. After the observation and comparing the results of the three questions proposed, a percentage of the users accused the Virtual Suspect without believing it was the author of either selling the stolen necklace or the jewelry robbery. One possibility to this observation could be that, because the participants are interacting in a police interview scenario, had a certain tendency towards considering the suspect guilty without evidences.

The overall results of the questionnaire showed that the participants were not fully confident on the sequence of questions asked but they all eventually caught the Virtual Suspect lying during the interaction. In addition to that, the users were not fully sure about their statements regarding the Virtual Suspect’s actions neither its accomplices.

From the empirical results, we could verify that the best sequence of questions made the Virtual Suspect readjust its shared story multiple times and, at some points of the interaction, the virtual suspect was forced to share the real events. While a sequence of questions that did not follow the police protocols rules not only could not expose as many lies but also made the Virtual Suspect adjust its story less times when compared to the other set of questions’ sequence.

This results show that the Virtual Suspect was able to mislead the user during the interaction until a certain point where he could not create more alternative events to adjust its false story to the user’s knowledge with the domain available. We could also verify that a interviewer that makes the best questions’ sequence will expose the agent’s lies faster (with less questions), whereas an user that proposes a worst sequence may never discover the truth.
8 Conclusion

Contents

8.1 Future Work ....................................................... 78
With this dissertation we intended to contribute to the intelligent virtual agent's research by presenting a model that describes the architecture for a virtual agent able to engage on an interactive dialogue and deceive, in real-time, about its story, without the need of previously scripted lies.

We started by investigating and presenting several theoretical concepts related with the work to be developed. We covered how Deception was defined and proposed theories about the cognitive process used by humans to create a lie in their daily interactions. We described how humans model what others are thinking and then provided a brief study about police interviews, their protocols and the suspects' behaviour during the interrogation.

With the background defined for the theoretical concepts used throughout the virtual agent's model, we reviewed different researches and agent's implementations concerning our problem. To study the work related with this dissertation's content, we divided this analyses in different topics. First, we covered a solution to a problem with the same motivation as ours. After that, we surveyed different interactive storytelling systems to discover the techniques used to adjust narratives based on the user's decisions. Finally we reviewed some memory architectures for agents and agents with deceptive capabilities used in different contexts.

Inspired and guided by the works studied before, we designed a virtual agent's model that receives questions about the agent's past actions and generates answers with the intention of misleading the asker. During the dialogue the agent creates and adjusts a parallel story to be shared while the compromising events in the original story are kept hidden. The model proposes a two-pass control vertically layered architecture that updates the agent's beliefs about the user's knowledge, creates events and adjusts its story to conceal incriminatory events.

In order to test the virtual agent's model's performance, we implemented the Virtual Suspect, a virtual agent playing the role of a suspect in a robbery. It would interact in a question-answering environment simulating a police interview where the user would ask questions to the suspect in order to discover more information about the crime being investigated. In addition to the implementation of the virtual agent's model, we developed two interfaces for the interaction between the Virtual Suspect and the users.

To test the agent's model with different questions' sequences, we designed an experiment where the participant had to interview the Virtual Suspect playing the role of Peter Barker, the main suspect of a robbery, aiming to discover more information about the case. With this study, we have collected not only the participants' perception of the Virtual Suspect lies, but also multiple questions' sequence asked by the user and the agent's behaviour during that interaction.

The results of these experiments helped us prove our hypothesis since the Virtual Suspect was able to mislead most of the participants and shared a false story during the interaction that
hid the compromising events of its real story. With these users’ studies we also observed an interesting remark: the agent was able to lie more successfully when asked a bad sequence of questions while the interviews that respected the police protocols made the agent recalculate the shared story more times.

8.1 Future Work

Although the results achieved were promising, The Virtual Suspect has room for improvements that can make the virtual agent capable of interacting during a long period in a police interview simulation. If such limitations were addressed, the agent's model designed could be used to teach police trainees the skills they need to successfully interview real suspects. We believe the following improvements could make the agent performance better:

- The Virtual Suspect implemented had a story with a duration of three complete days. Since the procedure used to retrieve similar entities needs to perform a search on the agent's Knowledge Base to look for suitable replacements, if the agent's domain does not contain enough alternatives for all the incriminatory content, the selected entity would be the best based on the all the possibilities but not good enough to mislead the user. We believe if a story with a longer duration was created, for instance describing an entire month of the Virtual Suspect's life, the agent would have more entities to chose from when creating and adjusting its story and, consequently, improving its lies believability.

- The representation used for the agent's story was designed to be flexible without many constraints, however the virtual agent's model would benefit if a representation with more semantics was used. In future versions of the Virtual Suspect, the story representation could find on Knowledge Representation's research alternative approaches from the one used.

- A limitation that directly influenced the users interaction with the Virtual Suspect is the interface used. Despite developing two different prototypes none of them were ideal, yet the second approach offered the best option based on preliminary test with a focus group. Some of the participants wished they could ask more questions about the agent’s past actions than the ones available. We believe if an approach that uses Natural Language Processing techniques is developed, the interaction between the user and the Virtual Suspect would highly improve.

- It would be interesting to test how the Virtual Suspect would handle a series of randomly generated questions as a brute force approach. This alternative experiment would require the development of another virtual agent to generate random questions to be proposed. The
interaction between both agents needed to be supervised to evaluate the Virtual Suspect’s answers.
Bibliography


Project Source Code Samples

A.1 Knowledge Base Code

This section contains the code and file listings related with the Virtual Suspect's Knowledge Base.

```xml
<story name="Agent's Name">
  <entity>
    <id>1</id>
    <type>...</type>
    <value>...</value>
  </entity>
  ...
  <entity>
    <id>n</id>
    <type>...</type>
  </entity>
</story>
```
Listing A.1: XML file structure representing the Agent’s Story.

A.2 Query Engine Code

This section contains the code listings related with the Virtual Suspect’s Query Engine component.

```csharp
public class LocationEqualConditionFilter : IConditionFilter{
    private string location;

    public LocationEqualConditionFilter(string location){
        this.location = location;
    }

    public Predicate<EventNode> CreatePredicate() {
        return delegate (EventNode node) {
```

86
Listing A.2: "Location Filter implementation"

```
public class TimeFocusPredicate : IFocusPredicate {
    public Func<EventNode, QueryResult.Result> CreateFunction() {
        return delegate (EventNode node) {
            return new QueryResult.Result(node.Time, node.Time.Count , KnowledgeBaseManager.DimentionsEnum.Time);
        };
    }
}
```

Listing A.3: Time Focus Predicate implementation

```
QueryResult result = new QueryResult(query);
List<EventNode> queryEvents = FilterEvents(query.QueryConditions);
if (query.QueryType == QueryDto.QueryTypeEnum.Validation) {
    result.AddBooleanResult(queryEvents.Count != 0);
} else if (query.QueryType == QueryDto.QueryTypeEnum.InformationGathering) {
    foreach (IFocusPredicate focus in query.QueryFocus) {
        result.AddResults(queryEvents.Select(focus.CreateFunction()));
    }
} result.CountResult();
```

Listing A.4: Code to query the Knowledge Base with the question asked

A.3 Virtual Suspect Code

This section contains the code listings related with the Virtual Suspect initialization and main methods.

```
SortedDictionary<int, ILayer> layers = new SortedDictionary<int, ILayer>();
```
layers.Add(1, new TheoryOfMindLayer(this));
layers.Add(2, new StrategySelectionLayer(this));
layers.Add(3, new StoryAdjustmentLayer(this));

Listing A.5: "Agent's initialization code"

QueryResult ProcessQuestion(Query question) {
    foreach (ILayer layer in layers.Values) {
        question = layer.HandleFirstPass(question);
    }
    QueryResult result = QueryEngine.ProcessQuery(question);
    foreach (ILayer layer in layers.Reverse().Values) {
        answer = layer.HandleSecondPass(answer);
    }
    return answer;
}

Listing A.6: "ProcessQuestion's body"
User Studies

These Appendix contains the questionnaires and results of the User’ Studies
Muito obrigada por colaborares neste estudo! Pedimos-te que agora leias as seguintes perguntas e respondas o mais honestamente possível.
Assinala as tuas respostas com uma cruz no número ou caixa que melhor representa a tua opinião.

1. Quão familiarizado estás com as técnicas de interrogatório usadas pela polícia?
   Nada familiarizado  1  2  3  4  5  Muito familiarizado

2. Tiveste algum treino na área de interrogatórios policiais ou entrevistas cognitivas?
   □ Sim   □ Não

3. Quão importante achas que é a vertente emocional num interrogatório?
   Nada importante  1  2  3  4  5  Muito importante

4. Quão importante achas que é a vertente lógica num interrogatório?
   Nada importante  1  2  3  4  5  Muito importante

Não vires a página!
Por favor, chama o investigador!

Sessão  :  

Página 1
Por favor responde agora às seguintes perguntas sobre o suspeito interrogado. Caso não tenhas a certeza sobre alguma pergunta, escolhe a resposta que acho mais adequada. Indica também o quão confiante estiveste da tua resposta.

1. **O suspeito, Peter Barker, vendeu o colar ao John Frey?**
   - [ ] Sim
   - [ ] Não

2. **Quão confiante estás da resposta anterior?**
   Nada confiante 1 2 3 4 5 Muito confiante

3. **O suspeito assaltou a joalheria?**
   - [ ] Sim
   - [ ] Não

4. **Quão confiante estás da resposta anterior?**
   Nada confiante 1 2 3 4 5 Muito confiante

5. **Quem era cúmplice do Peter Barker?**
   - [ ] John Frey
   - [ ] Delivery Boy
   - [ ] Jaime
   - [ ] Shop Owners

6. **Quão confiante estás da resposta anterior?**
   Nada confiante 1 2 3 4 5 Muito confiante

7. **Quanto confiaste no suspeito durante a entrevista**
   Não confiei nada 1 2 3 4 5 Confiei plenamente

---

*Figure B.2: Questionnaire Page 2*
8. **Achas que o suspeito mentiu durante a entrevista?**
   - [ ] Sim, ele mentiu sempre.
   - [ ] Não, ele nunca mentiu.
   - [ ] Mentiu às vezes, mas também disse a verdade.

9. **Quão eficaz achas que foi a tua escolha de perguntas feitas ao suspeito?**
   
   Nada eficaz 1 2 3 4 5 Muito eficaz

10. **Relativamente ao assalto da joalheria, como consideras o suspeito?**
    
    - [ ] Culpado
    - [ ] Inocente

11. **Quão confiante te sentes da tua decisão?**
    
    Nada confiante 1 2 3 4 5 Muito confiante

**Obrigado pela participação!**