Explaining Agent Behavior in Virtual Training

SIKS Dissertation Series No. 2011-35.

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

The research reported in this thesis was partly conducted at TNO Human Factors, Soesterberg, The Netherlands.

This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO).

Copyright © 2011 by Maaike Harbers

ISBN 978-90-393-5657-9

Explaining Agent Behavior in Virtual Training

Het verklaren van agent gedrag in virtuele training

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof.dr. G.J. van der Zwaan, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 2 november 2011 des middags te 2.30 uur

door

Maaike Harbers

geboren op 2 april 1982 te Grolloo

Promotor:	Prof.dr. J.-J. Ch. Meyer
Co-promotor:	Dr. K. van den Bosch

Contents

Chapter 1

Introduction

Computer games are increasingly often used for training purposes. Such ‘serious games’ are exploited to train competences such as leadership, negotiation or social skills. Sometimes, behaviors of the virtual characters in a training game are not clear. Consider for instance the following fragment of a training scenario, and imagine that you practice the tasks of a leading firefighter.

> *When you are called for a fire in a house, you and your four team members get into a fire engine and drive to the location of the incident as quickly as possible. Once arrived, you see smoke coming out of a house, people are standing dangerously close by and from the distance a siren is approaching. You have to assess the situation quickly, make an attack plan and instruct your team. Subsequently, while discussing with a policeman where to block the road, you see that your first two team members, against your instructions, enter the house through the back instead of the front door. The next moment, a woman tells you in panic that her dog is still inside the house...*

Somewhat later, you finished the virtual training successfully. But though the fire has been extinguished, the dog has been saved and the roads are open again, you still do not know why the two virtual team members did not follow your orders. You would like to actually ask them: *“Why did you enter the house through the back door?”* The virtual team members may explain that they misunderstood your instructions. From this explanation, you learn that you will have to provide clearer instructions in similar situations in the future. The team members may also explain that they found explosive material at the back door, which they had to remove before entering the house. In that case, you know that your instructions were clear, but that you should have checked for the presence of explosive material. Knowing the reasons behind the actions of virtual characters thus gives more insight into the training situation and your own performance.

This thesis is about automatically generating explanations of the behavior of virtual characters in training games. The behavior of these characters is also computer-generated, that is, by intelligent software agents. When trainees can ask their virtual team members, colleagues or opponents to explain the reasons behind their actions, they are given the opportunity to understand played scenarios better. Thus, the aim of explaining the behavior of such agents is to support trainees' learning from virtual training.

In this chapter we will provide a motivation for the research in this thesis, describe our research plan and give an outline of the thesis. In the last section we will also list on which publications each of the chapters in this thesis is based.

1.1 Motivation

In this section we motivate the goal of our research, developing explainable agents for virtual training. We will first discuss the notions of virtual training and intelligent software agents. Subsequently, we provide an overview of existing explanation components that explain agent behavior in virtual training. Finally, we present the approach for explaining agent behavior that we will take in this thesis.

1.1.1 Virtual training

Virtual training systems typically visualize a virtual environment in which trainees have to accomplish a given task or mission. By that, virtual training systems offer the possibility to practice tasks while the actions of trainees have no effects in the real world. This makes virtual training particularly appropriate to train tasks or skills that are dangerous or expensive to practice in reality, or that involve actions with important consequences in the real world. For example, the negotiation skills of a military commander may determine the safety of many soldiers and civilians, and the capacities of a leading firefighter have a big influence on the outcome of a fire attack. Scenario-based virtual training has demonstrated considerable potential for learning tasks in complex and dynamic environments (Oser, 1999).

For effective training, it is important that the skills learned in the virtual world are applicable in the real world, i.e., there should be a good transfer of training (Baldwin and Ford, 1988). To achieve transfer of training, those parts of the real world that are relevant to the execution of the training task must be are offered to trainees in a realistic way. This requires realistic computer simulations or human experts controlling the virtual world, or a combination of both.

To ensure realistic virtual character behavior, subject matter experts (usually instructors) often play the roles of key players in virtual training. Houtkamp and Bos (2007), for instance, describe a virtual training system for leading firefighters in which instructors manage the changes in the environment such as the size of a fire, and impersonate other players in the scenario, e.g., a police officer or bystander. Also in the military domain, it is common practice that more than one instructor is needed to train a single trainee (Van den Bosch and Riemersma, 2004). A disadvantage of this approach is that instructors are generally scarcely available, and the need for multiple instructors elevates costs of training. As a result, there are often only a few opportunities to receive this type of training, whereas trainees need a large number of training opportunities. In fact, acquiring expertise for complex tasks requires intensive, deliberate and reflective practice over time (Ericsson et al., 1993).

To create more training opportunities, artificial intelligence can be used to take over (a part of) the instructor's tasks. Intelligent software agents, for example, can be used to play the roles of the virtual characters in the training scenario autonomously. Recently, several virtual training systems with intelligent agents have been introduced, e.g., to train tactical command and control (Van Doesburg and Van den Bosch, 2005), leadership skills (Riedl and Stern, 2006), negotiation skills (Core et al., 2006b), and deal with bully behavior (Hall et al., 2006). When intelligent agents produce realistic behavior, one instructor can monitor a group of trainees and provide help and answer questions if necessary. Trainees could also use virtual training systems without the presence of an instructor, and save their questions for later. Virtual training that can be used (more) independently is cost-efficient, and gives trainees the flexibility to train wherever and whenever they want.

1.1.2 Intelligent software agents

The concept of an agent is used in computer science to denote an entity that perceives and performs tasks in an environment more or less independently. Software agents can be opposed to agents that have a physical body like hardware agents (robots), and biological agents (animals or humans) (Franklin and Graesser, 1997). Applications of software agents are for instance online buying and shopping, providing personal assistance, and representing human behavior in (training) games (Jennings and Wooldridge, 1998).

There has been much discussion about what exactly constitutes an agent (Castelfranchi, 1997), and many definitions of an agent have been proposed (Franklin and Graesser, 1997). In general, definitions that are acceptable for many researchers are often considered too broad, but more specific definitions are usually only accepted by a small group of people. In this thesis, we will follow one of the most common definitions of agents (Wooldridge and Jennings, 1995; Wooldridge, 2002), which says that: "an agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives".

Shoham introduced the agent-based programming paradigm (Shoham, 1993). Agent-based programming uses (proactive) agents as the main components of a program, in contrast to object-oriented programming, where (reactive) objects form the building blocks of a computer program. There are a number of programming languages that support the development of agents and multi-agent systems (Bordini et al., 2005, 2009).

Agents can be used to represent human behavior in order to generate believable behavior of virtual game characters (Gratch et al., 2002; Norling, 2003; Patel and Hexmoor, 2009). However, as shown in the example at the beginning of this chapter, it is not always clear why agents act the way they do, and without knowing the motivation of an agent's actions, it may be difficult for a trainee to understand the situation and learn from it. Therefore, several systems that explain agent behavior in virtual training have been proposed.

1.1.3 Existing approaches to explainable agents

In this section we will give a brief overview of existing approaches to explaining agent behavior, in order to motivate our research. In Subsection 2.1.3 we will provide more detailed descriptions of the approaches.

The first approach in which agent behavior is explained in virtual training is **Debrief** (Johnson, 1994). Debrief is an explanation component that is implemented as part of a fighter pilot simulation in which a trainee pilot interacts with a pilot played by an agent. Debrief constinously logs the agent's state during a training session. Afterwards, the trainee can ask explanations about any of the artificial fighter pilot's actions. Based on the logs of the agent's state, Debrief determines what must have been the factors responsible for the agent's decisions, e.g., the position of the aircraft in relation to its environment, and provides them as an explanation to the trainee.

A second approach to explainable agents, **XAI 1** (Van Lent et al., 2004), forms part of a simulation-based training for commanding a light infantry company. After a training session, the trainee can select a time and an agent, and ask questions about the agent's physical state at that point, such as its location or health.

A third approach is **VISTA 1**, the Visualization Toolkit for Agents (Taylor et al., 2002). VISTA 1 is applicable to different intelligent agent architectures, knowledge representations and task domains, and aims to visualize the internal reasoning processes of intelligent agents. The Situation Awareness Panel (SAP) is a particular instantiation of VISTA that visualizes the knowledge and reasoning of Soar agents operating in the tactical air combat domain.

The three approaches discussed so far (Debrief, XAI 1 and VISTA 1) continuously store the agent's state during training, and use these so-called behavior traces to generate explanations. The advantage of this method is that it is efficient to develop. Agents already perform (reasoning) steps to generate actions, and these steps are reused in order to generate explanations. A disadvantage of this method, however, is that behavior traces do not always yield useful explanations. The behavior traces logged in XAI 1, for instance, only contain information about the agent's physical state and not about its mental state. Therefore, XAI 1 fails to provide explanations that give insight into the agent's mental state, e.g., its reasons for performing an action. And as VISTA 1 is applicable to different training systems, the usefulness of its explanations will strongly depend on the agents' behavior representations used in that system. For example, if an agent's behavior is generated by a neural network, the visualization of behavior traces will not give much insight to a trainee.

In order to provide more useful explanations, several alternative approaches to explaining agent behavior were proposed. The fourth approach is a more recent version of the XAI 1 approach. This new approach, **XAI 2** (Gomboc et al., 2005; Core et al., 2006b), claims to provide explanations about the motivations behind an agent's actions, and to be domain-independent, i.e., applicable to different virtual training systems. XAI 2 still imports behavior traces from virtual training systems, but when these do not include an agent's goals, the developers manually build a XAI representation of the agent's behaviors including its goals.

The fifth approach is **VISTA 2** (Taylor et al., 2006), which is a follow-up of VISTA

1. VISTA 2 was also extended to be able to explain the rationale behind agent behavior, e.g., why a particular goal was selected. This information cannot automatically be imported from training simulations, but has to be added by the developers of the explanation approach.

The sixth approach, **TRACE** (Young and Harper, 2005), can also be applied to different training systems, and allows users to investigate an agent's beliefs, goals, and percepts to answer why a certain decision was made. TRACE uses a domain-independent ontology and developers extend it to match the target domain and simulation.

The last three approaches (XAI 2, VISTA 2 and TRACE) all provide explanations that include the motivations, reasons, or goals behind an agent's actions. By that, they give trainees useful insights into the agent's behavior. Behavior traces alone, however, are often not sufficient to generate such explanations, and developers need to add extra information to an explanation component in order to generate useful explanations. The developers of XAI 2, for instance, mentioned that training simulations differ in their *explanation-friendliness* (Core et al., 2006a). At best, agent behavior is represented by goals, and the preconditions and effects of actions. In such a case, XAI 2 can automatically import behaviors, and the training simulation is considered explanation-friendly. At worst, behavior is represented by procedural rules. Then, a manually built XAI representation of the behaviors has to be made, and the training simulation is not considered explanation-friendly. Thus, though these more recent explanation approaches give trainees more insight into the agents' motivations, their development requires more effort than the development of earlier approaches.

Existing research on explaining agent behavior in virtual training focuses on the development of explanation approaches. There is not much literature describing studies that evaluate the effectiveness of the proposed approaches. An exception forms the work of Haynes et al. (2009), who performed several studies on what explanations users of virtual training systems require. Based on the results of these experiments, Haynes et al. proposed a framework for explaining intelligent agents in simulations and decision support systems.

1.1.4 Our approach to explainable agents

In this thesis, we aim to develop an approach that combines two strengths in existing approaches: providing useful explanations for actions (goals, motivations) in an efficient way (using behavior traces). The existing approaches that provide goals and motivations as explanation for actions are all application-independent explanation components (Gomboc et al., 2005; Core et al., 2006b; Taylor et al., 2006; Young and Harper, 2005). Though it is an advantage that these components can be applied to many training simulations, application independence also has a major drawback. The developers of the explanation components do not control the way in which agent behavior is represented in training simulations, and often have to represent most of the information needed for explanations manually. To overcome this extra work, we adopt an approach that is not application-independent, but instead, integrates the development of agents and their explanation capabilities. By exerting control over the development of agents in virtual training, we can ensure that the agents' behavior is represented in an explanation-friendly way, and use behavior traces for the generation of explanations.

Our approach requires that an agent has explicit representations of the concepts and processes by which its behavior should be explained. We already mentioned that agent behavior should not be explained by physical properties only, but also by an agent's underlying goals or motivations. This is supported by psychological research showing that people usually explain and understand human (or human-like) behavior in terms of mental concepts such as beliefs, goals, and intentions (Malle, 1999; Keil, 2006). In Dennett's words, people adopt the *intentional stance* to understand and explain their own and others' behavior (Dennett, 1987). We aim to develop agents that display *human* behavior, and therefore we believe that people will understand the agents' behavior best when it is explained in a similar fashion as human behavior. Thus, in our approach agents' actions are explained by their underlying goals, plans, and beliefs.

BDI-based (Belief Desire Intention) programming languages allow for the representation of agent behavior by beliefs, goals, plans, and intentions, and a BDI agent determines its actions by a reasoning process on its mental concepts (Rao and Georgeff, 1991, 1995). We expect that by modeling explainable agents as BDI agents, the mental concepts that are responsible for the generation of an action, can also be used to explain that action. Therefore, we adopt a BDI-based approach to developing explainable agents.

To summarize, connecting the generation and explanation of agent behavior will not result in application-independent explanation components such as in most of the existing approaches. However, it avoids that developers have to represent all behaviors twice, that is, first in the agent itself and then again in the explanation component.

Besides that we adopt a new approach to the development of explainable agents for virtual training, we will also pay more attention to the empirical evaluation of explainable agents than is done in existing work.

Our approach to explainable agents can be useful in other application domains than virtual training. For instance, in tutor and pedagogical systems, natural dialog between the user and system has been shown to increase the training effect of such systems (Graesser et al., 2005). Debugging tools for BDI agent programs might benefit from a natural way of interaction involving asking why agents perform certain actions instead of looking at execution traces and internal mental states (Broekens and De Groot, 2006). In gaming and interactive storytelling (Cavazza et al., 2002; Theune et al., 2003), having automatic mechanisms to generate explanations of agent actions could enhance the flexibility and appeal of the storyline.

1.2 Research plan

In this section we present the research aim and questions that will be addressed in this thesis. Subsequently, we present the used research methodology, and finally, we provide an overview of the context in which this research was conducted.

1.2.1 Research aim and questions

The aim of the research presented in this thesis is as follows.

> **Research aim:** *To help trainees to learn from virtual training by using BDI agents that explain their own behavior.*

In this thesis, we focus on two questions in particular. The first question concerns the learning process of trainees.

> **Question 1:** *What explanations about agent behavior can help trainees to learn from virtual training?*

As we will show in Chapter 2, there are many different ways to explain one single event, action or phenomenon. We argued that human-like agent behavior is probably best understood when it is explained in terms of mental concepts such as beliefs and goals. But even with this constraint, there are still many ways to explain an action. Therefore, we will investigate which types of explanations are considered most useful, i.e., can contribute most to learning of trainees.

When it is known which types of explanations about agent behavior are required, BDI agents that provide such explanations can be developed. The process of developing agents that can explain their behavior is addressed in the second research question.

> **Question 2:** *How can we develop explainable BDI agents that help trainees to learn from virtual training?*

This question addresses a more technical part of the research. To answer this question, we will investigate how we can represent agent behavior in a BDI model, and how we can use such behavior representations to generate explanations in an efficient way.

1.2.2 Methodology

Now that we have presented the research questions, we will describe the methods used to answer them. For that, we use the distinction between design science and natural science, where we follow March and Smith's (1995) use of the term natural science, which includes, besides research in physical and biological domains, also research in social, and behavioral domains.

The notion of design science, also called the science of the artificial, was introduced by Simon (1969) and denotes sciences in which, in contrast to the natural sciences, the objects of research are artificial. Simon claims that this type of research requires an alternative research methodology from natural science. The goal of natural science is forming theory about phenomena in the world. A typical research cycle involves the collection of observations, theory formation, theory testing in empirical experiments, and possibly adapting the theory. The goal of design science, in contrast, is to accomplish the goals or purposes of a designer, e.g., to create a certain functionality or make a process more efficient. A typical research cycle involves the design of an artifact and its evaluation. The distinction between natural and design science is generally accepted. March and Smith (1995) criticized Simon's view, however, by arguing that developing theory is not restricted to natural phenomena, but can also apply to artifacts.

The research reported in this thesis clearly tries to achieve a design goal, that is, developing explainable agents for virtual training. The design part of the research is addressed by the second research question, and conform design science, a major part of the activities performed in this research are design and evaluation. However, like March and Smith, we believe that theory development can also apply to artifacts, and in our view, the research addressed by the first research question should be classified as natural science.

The following research activities are performed in each chapter. Chapter 3 presents the initial design for explainable agents, which is based on a literature research described in Chapter 2. Chapter 4 contains a natural science research cycle, investigating which explanation types people prefer. These empirical experiments are explorative in nature. Rather than collecting data from as many subjects as possible, they aim to yield new ideas to improve the design and application of explainable agents. In Chapter 5, the (improved) design of Chapter 3 is evaluated. Chapter 6 describes new design activities and presents an extension to the initial design. Finally, in Chapter 7, several steps towards a framework for explaining agent behavior are made. In summary, Chapter 3, 5 and 6 mostly describe design science research, and Chapter 4 and 7 mainly describe activities that can be classified as natural science.

The most important criterion to evaluate explainable agents for virtual training is to test whether they contribute to learning of trainees. In general, most models that generate behavior are evaluated by their intended use, that is, from the perspective of the end-user (Chandrasekaran and Josephson, 1999), the trainee in our case. For this type of evaluation we use experimental techniques from psychology and human-computer interaction.

Van Doesburg (2007) states that, besides the end-user perspective, human behavior representation models can also be evaluated from a psychological and developer's perspective. The psychological perspective considers how truthful the generation of human behavior is represented, or in other words, how well the cognitive processes underlying the observable behavior are simulated. We do not adopt this perspective in this thesis. As mentioned in Subsection 1.1.2, our aim is not to simulate human cognition, but to develop models that generate realistic human behavior and that have the ability to explain it.

The developer's perspective concerns the effectiveness and efficiency of model creation. We do pay attention to this perspective. As discussed in Subsection 1.1.3, some ways of representing agent behavior are more explanation-friendly than others, and we aim to reduce the amount of work for a programmer when developing explainable agents. The evaluation from the perspective of the developer is mostly based on our own experiences because, though there are standard works for the assessment of software quality, e.g., the IEEE Standard 1061 IEEE98, these are not available for representations of human behavior in particular (Harmon et al., 2002).

1.2.3 Research context

The idea for this research project originated at TNO Human Factors, when at several points in the development of virtual training systems with intelligent agents the reasons behind the agents' behavior were unclear, even for the developers of the system (e.g., the TACOP training system (Van Doesburg et al., 2005)). Of course, the developers could look into the programming code to figure out why the agents were acting the way they did. But trainees cannot do that, and moreover, programming code is not always easy to interpret. Therefore, it was considered useful to make the agents explainable and give trainees the opportunity to ask the agents for the motivations of their behavior. Eventually, that idea has led to this PhD thesis.

The research has been performed in a collaboration between TNO Human Factors and Utrecht University. Through TNO, there was access to virtual training systems and its users. Through Utrecht University, we had access to expertise in the field of intelligent software agents, and the agent-based programming language 2APL described in Subsection 3.1.2. We also made use of the knowledge and facilities at other places. The research described in Section 5.3 was performed at the Institute for Human and Machine Cognition (IHMC) in Pensacola in Florida. The research described in Section 4.3 and 5.2 has been performed in collaboration with the Human-Machine Interaction group at the Technical University Delft.

The research was funded by GATE, Game Research for Training and Entertainment (GATE, 2011). The goal of GATE research is to advance the state-of-the-art in gaming, simulation and virtual reality substantially, in order to create highly effective entertainment products and learning systems. The research presented here is part of the Virtual Characters theme, which deals with the creation of realistic behavior for the virtual characters that inhabit the virtual worlds and games. Besides academic research into games and game-technology, GATE also aims to develop this knowledge further into practical solutions. That happens through projects in which small and medium size enterprises collaborate with research partners, in which companies provide knowledge questions and intended applications, and the research partners provide new technology.

1.3 Thesis outline

In **Chapter 1** the topic of this thesis is introduced. The introduction includes a motivation for the research, two research questions, our research methodology and an overview of the research context. The rest of this thesis is organized as follows.

Chapter 2 is a background chapter that discusses related work. The first section provides an overview of explanation research in different fields, and is an extended version of the overview provided in a publication at the conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH) (Harbers et al., 2011c). The second section gives an overview of BDI-based agent programming. The third section of this chapter discusses related work on providing feedback in virtual training, and partly incorporates work discussed in a publication at the workshop on Human Aspects in Ambient Intelligence (Mioch et al., 2008). The fourth section provides a broader perspective on the research topic, and discusses two problems that are related to agent-based virtual training. The first part of this section discusses difficulties of connecting intelligent agents to game engines (Subsection 2.4.1), and is based on parts of a publication in the International Journal of Computer Games Technology (Dignum et al., 2009). The second part discusses the issue of balancing player freedom and scenario direction in virtual training (Subsection 2.4.2). This topic was first introduced in a publication at the Digital Human Modeling Conference (Van den Bosch et al., 2009), and further elaborated in a paper presented at the workshop on Human Dimensions in Embedded Virtual Simulation (Heuvelink et al., 2009).

Chapter 3 introduces an approach for developing explainable agents. The chapter involves the behavior representation model of explainable agents, which was first introduced in a paper at the workshop on Explanation-aware Computing (ExaCt) (Harbers et al., 2008), and further elaborated in an article at the Journal of Artificial Societies and Social Simulation (JASSS) (Harbers et al., 2010c). Next, the implementation of this model in 2APL is discussed, based on a paper that appeared as an extended abstract at the conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (Harbers et al., 2009b), and as a full paper at the workshop on LAnguages, methodologies and Development tool for multi-agent systemS (LADS) (Harbers et al., 2009c). Finally, the design and implementation of an explanation module is discussed, which appeared in a publication at the conference on Intelligent Agent Technologies (IAT) (Harbers et al., 2010b). The approach introduced in this chapter yields agents that are able to provide different types of explanations.

Chapter 4 presents three user studies that evaluate which explanation types are preferred by novices, experts, and instructors. The first study was conducted in the domain of onboard firefighting and has been published at the conference on Intelligent Virtual Agents (IVA) (Harbers et al., 2009e), the second study was performed in the domain of firefighting and has been published at the conference on Intelligent Agent Technologies (IAT) (Harbers et al., 2010b), and the third study was performed in the domain of cooking and has been published at the conference on Multi-agent System Technologies (MATES) (Broekens et al., 2010a). The last section of Chapter 3 presents guidelines for the design and modeling of explainable agents, based on the three studies. The guidelines have been published at the International Conference on Cognitive Modeling (ICCM) (Harbers et al., 2010a).

Chapter 5 presents three studies that investigate the effects of explanations generated according to our approach on performance. The first two studies investigate the effects of explanations on learning from virtual training. The first study was conducted in the domain of onboard firefighting and has not been published. The second study was performed in the domain of negotiation and is accepted as a poster at the conference on Intelligent Virtual Agents (IVA) (Broekens et al., 2011). The third study investigates the effects of our explanations on coordination in human-agent teams, and has been published at the workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN) (Harbers et al., 2011a). The final section of this chapter provides an overall discussion.

Chapter 6 discusses the development of explainable agents with a theory of mind, and their application in virtual training. The chapter starts with an example training scenario involving agents with a theory of mind, and an overview of theory of mind literature. Based on the literature, two executable models of agents with a theory of mind are proposed (Section 6.3), which were first described in a paper published at the workshop on Educational Uses of Multi-Agent Systems (EduMAS) (Harbers et al., 2009a). Subsequently, a simulation study comparing the two models is described (Section 6.4), which has been published at the conference on Intelligent Agent Technology (IAT) (Harbers et al., 2009d). In the next section, extensions to the programming language 2APL are proposed in order to implement agents with a theory of mind (Section 6.5). This section is based on a book chapter that appeared in "'Multi-Agent Systems for Education and Interactive Entertainment: Design, Use and Experience"' (Harbers et al., 2011b). An article with an overview of all the work presented in this chapter will appear in the Journal of Web Intelligence and Agent Systems (WIAS) (Harbers et al., 2012).

Chapter 7 provides a discussion of the work presented in this thesis. First, the advantages and disadvantages of our approach are discussed. Then, the usefulness of our approach in different application domains is discussed. The section about explainable agents in social simulations is based on a publication in the Journal of Artificial Societies and Social Simulation (JASSS) (Harbers et al., 2010c). Finally, a more general perspective on explanations about agent behavior is provided, based on a publication at the conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH) (Harbers et al., 2011c).

Chapter 8 concludes the thesis. In this chapter, the main contributions of the thesis are discussed, and suggestions for future research are made.

In this thesis, male pronouns ('he', 'him' and 'his') are used when referring to a person of unspecified gender.

Chapter 2

Background

This chapter discusses literature from several fields that is relevant to the research presented in this thesis. We start with an overview of research on explanation, in which we particularly focus on the content of an explanation. In the second section, we give an overview of BDI-based agent programming, the family of languages we will use to implement explainable agents. In the third section, we discuss work on how explanations and feedback are provided in virtual training. In the fourth section, we provide a broader perspective on the research of this thesis by discussing advantages and difficulties of agent-based virtual training in general. We use the term agent-based virtual training to refer to virtual training in which the behavior of virtual characters is generated by intelligent agents.

2.1 An overview of explanation research

Most events, processes or phenomena can be explained in different ways. One explanation is not by definition better than another; the desired explanation depends on the receiver of the explanation and the context in which it is given. For example, possible explanations for why an apple fell are that someone dropped it, because the person holding the apple stumbled, or because someone pushed the person holding the apple. A whole other type of explanation is that the apple fell because of the gravitation force.

Explanation is a complex phenomenon that has been studied in different fields. In this section, we discuss psychological research about explaining human behavior, followed by work on argumentation in which the motivation for action is studied. Finally, we discuss literature in the field of artificial intelligence on the explanation of intelligent system behavior, and more specifically, of agent behavior.

2.1.1 Explaining human behavior

Keil (2006) provides an extensive overview of explanation in general, in which he categorizes explanations according to the causal patterns they employ, the explanatory stances they invoke, the domains of phenomena being explained, and whether they are value or emotion laden. He notes that any phenomenon or action has many possible explanations, but that people usually need remarkably little information for a satisfying explanation. Which piece of information best explains an action depends on the context of the action, and the person to whom the action is explained. Designing automatic explanation generation is not always easy, as people frequently prefer one explanation to another without explicitly knowing why (Kozhevnikov and Hegarty, 2001).

For the explanation of human behavior, Keil refers to Dennett's three explanatory stances: the mechanical, the design, and the intentional stance (Dennett, 1987). The mechanical stance considers simple physical objects and their interactions, the design stance considers entities as having purposes and functions, and the intentional stance considers entities as having beliefs, desires, and other mental contents that govern their behavior. Humans usually understand and explain their own and others' behavior by adopting the intentional stance. The intentional stance is closely related to the notion of folk psychology, which refers to the way people think that they think. Folk psychology determines the language people use to describe their reasoning about actions in everyday conversation (Norling, 2004).

Besides folk psychology, attribution theory is one of the most important theories about explanations of human behavior in psychology (Heider, 1958; Kelley, 1967). Attribution theory focuses on the various causes that people assign to events and behavior. External attribution assigns causality to factors outside of the person, e.g., the weather. Internal attribution assigns causality to factors within the person, e.g., own level of competence. Related to attribution theory is the concept of explanatory style, i.e., people's tendency to explain causes of events in particular ways (Buchanan and Seligman, 1995). People with a negative explanatory style believe that positive events are caused by things outside their control and that negative events are caused by them. People with a positive explanatory style, in contrast, believe that positive events happened because of them and that negative events were not their fault. Explanatory style is part of someone's personality.

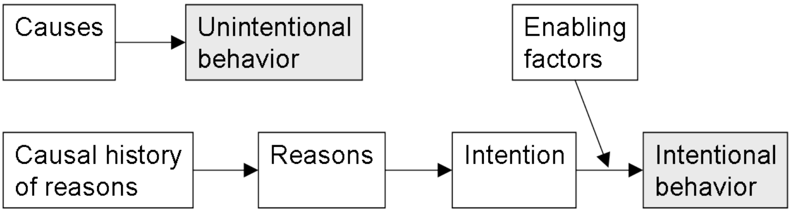


Figure 2.1: Malle's four modes of explaining behavior.

Malle (1999) criticized attribution theory because it does not make a distinction between the explanation of intentional and unintentional behavior. He proposed a framework about how people explain behavior that distinguishes four modes of explanation (see Figure 2.1). One mode considers explanations about unintentional behavior and the other three consider explanations about intentional behavior. The three explanation modes of intentional behavior are (1) reasons, (2) causal history of reasons, and (3) enabling factors. Reason explanations consist of beliefs and goals, causal history explanations explain the origin of beliefs and goals, and enabling factor explanations consider the capabilities of the actor. Malle states that people mostly give reason explanations for intentional behavior.

In psychological literature, Malle's proposal is one of the most elaborate frameworks on how people explain behavior. It gives a useful overview of different explanation types. However, the distinction between different types may not always be clear. Though beliefs are categorized as reason explanations, enabling factor and causal history explanations may also involve beliefs, i.e., beliefs about an enabling factor or causal history, respectively. For instance, 'I go to the supermarket because I believe I will be able to reach it before closing time'. Is the explanation of me going to the supermarket a reason or an enabling factor? The same confusion can occur between reason explanations with a goal and causal history explanations. For instance, I go to the supermarket because I want to buy food, I want to buy food because I want to prepare a meal, I want to prepare a meal because I am hungry, I am hungry because I just smelled food, etc. In this chain of explanations it is not clear which are goal explanations and which are causal history explanations.

2.1.2 Argumentation and explanation

In philosophical literature, explanations used to be viewed within the deductive-nomological model, in which explanations are seen as logic proofs in natural language (Hempel and Oppenheim, 1948). In this view, an explanation consists of a set of laws and the deductive consequences of those laws. Later, however, it was acknowledged that everyday explanations almost never have the structure of such logic proofs, and more practical approaches to explanation emerged (Salmon, 1989).

One of the fields studied in philosophy that is relevant for the explanation of agent behavior is argumentation, and argumentation for practical reasoning in particular. Argumentation is the study of how humans should, can, and do reach conclusions through logical reasoning. In argumentation for practical reasoning, argumentation about what is most sensible to do is studied. This is closely related to explanation of behavior, as arguments for a certain action can also be used to explain that action (Moulin et al., 2002). Atkinson et al. (2006) proposed an argumentation scheme for practical reasoning, which provides a motivation of an action.

In the circumstances R
we should perform action A
to achieve new circumstances S
which will realize some goal G
which will promote some value V.

When using the argumentation scheme for explanation, an action can be explained by the current circumstances, the new circumstances it will achieve, the goal it will realize, and the value that it promotes. For instance, I go to the supermarket (A) because I am at home (R) and after going there I will be at the supermarket (S), and I want to buy food (G) so that I can eat healthy (V).

The argumentation scheme has similarities with Malle's framework. Goals in this scheme could be mapped to reason explanations in the framework. Values have similarities with causal history explanations, though a causal history is broader than a value and can involve other aspects as well. A description of the current and new circumstances in this scheme could be seen as beliefs in Malle's framework, but this link is less clear. Note that in this argumentation scheme sometimes the new circumstances an action achieves are the same as the goal that it realizes. For instance, I go home (A) because I want to be home (G), and after going home I am home (S).

2.1.3 Explainable AI

Explainable AI stands for explainable artificial intelligence and refers to the explanation of intelligent system behavior. Most research in explainable AI has been done in the field of expert systems, where outcomes such as diagnoses or advices are often accompanied by an explanation (Swartout et al., 1991; Swartout and Moore, 1993; Wick and Tompson, 1992). Such explanations increase the user's acceptance, understanding and confidence in the decisions and recommendations of the system (Nakatsu, 2004; Herlocker, 1999; Dhaliwal and Benbasat, 1996; Ye and Johnson, 1995).

In the field of expert systems, the following four types of explanation are distinguished: (1) trace, (2) justification, (3) strategy, and (4) terminological explanations (Gregor and Benbasat, 1999). Trace explanations (1) are explanations in which the outcome of a system, such as a diagnosis or an advice, is explained by the steps that lead to it. For example, the outcomes of rule-based systems are explained by the rules that were applied (Buchanan and Shortliffe, 1984), and the outcomes of case-based reasoning systems are explained by cases (Srmo et al., 2005). A trace explanation of an expert system on medical diagnoses is for example: 'the patient has disease X because it shows symptoms Y and Z'. The first explainable expert systems provided trace explanations. Justification explanations (2) contain the reasons behind a rule, e.g., an explanation of why symptoms Y and Z indicate disease X. Justification explanations were added to trace explanations because researchers concluded that simple traces of behavior generation usually do not provide sufficient information to explain a system's actions in a satisfying way (Swartout and Moore, 1993). Users not only wanted to know *how* an outcome was reached, but also *why* this trace of rules was applied. Strategy explanations (3) involve information about how the expert system uses its domain knowledge to accomplish a task. For instance, a system could explain how it calculates the probability that someone has a certain disease given a specific combination of symptoms. Terminological explanations (4) clarify the meaning of certain concepts. Currently, in expert systems usually a combination of different explanation types is provided.

Our aim is not to explain expert system behavior, but that of agents in virtual training. Haynes et al. (2009) propose a framework for explaining intelligent agents in simulations or decision support systems, based on several user studies. They distinguish the following four explanation types for explaining intelligent agents: (1) ontological explanations explain the properties of an agent, (2) mechanistic explanations tell how it works, (3) operational explanations show how to use it, and (4) design rationale explanations explain why it has been designed the way it is. Though related, Haynes et al.'s focus is still different from ours, as they study all kinds of explanations related to agents, whereas we study the explanation of agent behavior in particular.

In Subsection 1.1.3 we already shortly introduced six approaches to explaining agent behavior. Here, we will discuss them more extensively. The Debrief explanation component (Johnson, 1994) can explain actions of Soar agents in the TacAirSoar tactical air combat domain (Laird and Nielsen, 1994). Debrief makes use of Soar's infrastructure to continuously log an agent's state during a mission. After the mission, Debrief can recall the state of the agent at various points during the mission. It then modifies the state repeatedly and observes the effects of the modifications on the agent's decisions. By this process, Debrief can determine which factors must have been responsible for the agent's next decision. Debrief is not applicable to agents other than Soar agents.

The XAI 1 approach (Van Lent et al., 2004) forms part of a simulation-based training for commanding a light infantry company. Like Debrief, XAI 1 continually logs an agent's state during a training session. These logs contain physical information about the agent, like its location, ammunition and health. After the session, trainees can select a time and an agent, and investigate the agent's physical state at that time. XAI 1 is not applicable to other training simulations, and it is not possible to query agents about the reasons for their actions.

The third approach to explaining agent behavior that we mentioned in the introduction is VISTA 1 (Taylor et al., 2002), the Visualization Toolkit for Agents. VISTA 1 is applicable to different intelligent agent architectures, knowledge representations, and task domains, and aims to visualize the internal reasoning processes of intelligent agents. Explanations are generated from behavior traces. The Situation Awareness Panel (SAP) is a particular instantiation of VISTA 1 that visualizes the knowledge and reasoning of Soar agents operating in the tactical air combat domain.

XAI 2 is claimed to overcome shortcomings of XAI 1 (Gomboc et al., 2005; Core et al., 2006b). Namely, XAI 2 supports domain independence, i.e., it is applicable to different virtual training systems, and it has the ability to explain the motivations behind an agent's actions. To demonstrate its domain independence, the system has been applied to a tactical military simulator (Gomboc et al., 2005), and a virtual trainer for soft skills such as leadership, teamwork, negotiation and cultural awareness (Core et al., 2006b). As mentioned in Subsection 1.1.3, XAI 2 imports information that is made available by the simulation and uses it for the generation of explanations. When the available information is not sufficient to generate useful explanations, a manually-built XAI representation of the behaviors has to be made.

TRACE (Young and Harper, 2005) offers a graphical interface through which users can search for answers to the questions 'why' and 'why not' a certain action was taken. The interface shows an agent's beliefs, goals and percepts at different point in the training session. TRACE uses a domain-independent ontology and extends it to match the target domain and simulation. TRACE can be applied to different training systems.

VISTA 2 (Taylor et al., 2006) provides a general framework for building explanation systems across agents systems and application domains. Besides behavior trace knowledge, it several other knowledge sources to generate explanations, such as design knowledge, domain knowledge and display knowledge. The user interface involves both textual and graphical representations.

2.2 BDI-based programming

To implement agents that can generate explanations in terms of mental concepts, we will use BDI-based programming languages. The BDI-based agent programming paradigm is based on Bratman's theory of human practical reasoning, in which human reasoning is described with the notions of belief, desire and intention (Bratman, 1987). This theory is closely related to the notion of folk psychology mentioned in the previous section. Rao and Georgeff (1991) were the first to formalize Bratman's theory, and later, they developed a BDI-based software model (Rao and Georgeff, 1995). Note that BDI-based programming languages differ from agent-based programming languages that aim to simulate human cognition, such as Soar (Rosenbloom et al., 1993) and Act-R (Anderson and Lebiere, 1998). Instead of simulating the way people think, BDI languages simulate the way people think that they think.

BDI programming has been and is being developed at universities and is currently mostly used in scientific settings. Still, there are some examples of practical applications of BDI-based programming. For instance, BDI agents have been successfully applied to generate virtual player behavior in computer games (Norling, 2003) and in virtual training (Van Doesburg et al., 2005; Heuvelink et al., 2009).

In this section we will introduce a general BDI architecture. Subsequently, we will discuss the BDI-based programming languages 2APL and GOAL in more detail, as we will use these languages to illustrate our approach and to implement agents for the user studies in this thesis. For a more extensive discussion of the BDI approach we refer to Wooldridge (2000), who presents a mainstream view on BDI agents.

2.2.1 BDI architecture

These is no single BDI model. The BDI approach is represented by a family of BDI architectures, each implementing its own interpretation of BDI theory. BDI-based programming languages are for example PRS (Georgeff and Lansky, 1987), Jadex (Pokahr et al., 2003), Jack (Busetta et al., 1999), Jason/AgentSpeak (Bordini et al., 2007), ConGolog (De Giacomo et al., 2000), 3APL (Hindriks et al., 1999), 2APL (Dastani, 2008), and GOAL (Hindriks, 2009). These languages have in common that an agent's mental state is defined by its beliefs (representing the agent's knowledge), goals (desires) and intentions (goals to which the agent commits itself). Usually, BDI agents also have a plan library containing a set of plans, where a plan is a recipe for achieving a goal given particular preconditions. The plan library may contain multiple plans for the achievement of one goal. An intention is the commitment of an agent to execute the sequence of steps making up the plan. A step can be an executable action, or a subgoal for which a new plan should be selected from the plan library.

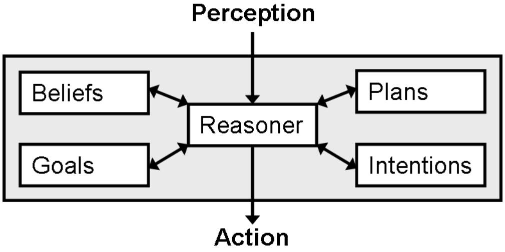


Figure 2.2: Overview of the BDI architecture.

Figure 2.2 displays a general BDI architecture shared by most BDI-based programming languages (adopted from Wooldridge, 2000). The mental state of a BDI agent (the gray box in the figure) is constituted by its beliefs, goals, plans, and intentions in its belief base, goal base, plan library, and intention stack, respectively. A BDI agent can perceive and act in its environment (perception and action in the figure). The behavior of a BDI agent is generated by a deliberation process on its mental state, performed by the reasoner. Deliberation cycles differ per agent architecture, but a typical BDI execution cycle contains the following steps: (i) perceive the world and update the agent's internal beliefs and goals accordingly, (ii) select applicable plans based on the current goals and beliefs, and add them to the intention stack, (iii) select an intention, and (iv) perform the intention if it is an atomic action, or select a new plan if it is a subgoal.

2.2.2 The 2APL language

2APL is a typical BDI-based agent programming language and allows for agent representations in terms of beliefs and goals. Moreover, 2APL is built in Java, which makes it suitable to extend the language with explanation facilities. In this section, we provide a short overview of 2APL. For a more complete and detailed overview of 2APL we refer to Dastani (2008).

The mental state of a 2APL agent is defined by its beliefs, goals, plans, and reasoning rules. When the agent is executed, a deliberation process on this mental state determines the agent's actions. 2APL agents can interact with environments, e.g., the blockworld environment, by performing actions in the environment and receiving external events from the environment.

A 2APL agent can execute different types of *actions*: actions that add and remove beliefs to and from the belief base, actions that pass a message to another agent, actions to interact with the environment, abstract actions encapsulating a plan by a single action, actions to test the belief and goal bases, and actions to add and remove goals to and from the goal base. In 2APL, an agent's *beliefs* are Prolog facts and rules, and the belief base of a 2APL agents is a Prolog program. Thus, from the beliefs *x* and *y :- x*, the belief *y* can be derived. The *goals* of a 2APL agent are declarative, that is, they what

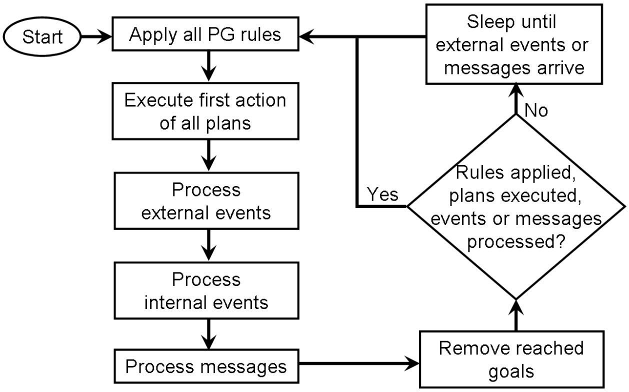

Start
Apply all PG rules
Execute first action of all plans
Process external events
Process internal events
Process messages
Remove reached goals
Rules applied, plans executed, events or messages processed?
Yes
No
Sleep until external events or messages arrive

Figure 2.3: 2APL deliberation cycle.

an agent wants to achieve, not how to achieve it. To reason with goals in the agent's goal base, so called PG-rules (Planning Goal rules) are used, which are of the form Goal <− Belief | Plan. Informally this means that if the agent believes Belief, then to achieve the Goal it should execute Plan. An agent can adopt a new goal by executing the action *adopt(Subgoal)*, which means that the Subgoal is added to the agent's goal base. A *plan* is a sequence of actions or subplans. A 2APL agent can reason with plans through PC-rules (Procedure Call rules), which are of the form Plan <− Belief | Plan'. If an agent has Plan in its plan base and Belief in its belief base, it can execute Plan by executing Plan'. In the agent's plan base, Plan is replaced by Plan', which usually contains a set of actions and/or subplans.

Figure 2.3 shows the deliberation cycle of a 2APL agent. The cycle starts with trying to apply all PG-rules. Subsequently, the first action of all plans are executed, and external events, internal events and messages are processed. Then reached goals are removed. Next, it is checked whether any rules were applied, plans were executed, or events or messages were processed in the current deliberation cycle. If yes, a new cycle starts, and if no, the process sleeps until an external event or message arrives.

2.2.3 The GOAL language

Like in 2APL, GOAL agents are specified by mental concepts, and their actions are determined through a reasoning cycle on these mental concepts. A GOAL agent program consists of the six different sections, specifying the agent's knowledge, beliefs, goals, action rules, action specifications, and percept rules. In the remainder of this section we only discuss two important differences between GOAL and 2APL. For a more extensive overview of GOAL we refer to Hindriks (2009).

The first difference between GOAL and 2APL is that in 2APL all beliefs are part of one belief base, whereas in GOAL the distinction between knowledge and beliefs is made. An agent's knowledge section contains general knowledge or rules about the world, e.g., a fire can be extinguished with water, and an agent's belief section contains observations about the current state of the environment, e.g., there is a fire at location X.

The second difference between GOAL and 2APL is the usage of plans. 2APL knows abstract actions, which are actions that encapsulate a plan. Therefore, besides PG-rules, used to reason with goals, 2APL also has PC-rules, reasoning rules that are used to reason with plans. In GOAL there are no abstract actions, and it only has one type of reasoning rules, i.e., action rules. In contrast to the PG-rules and PC-rules in 2APL, an action rule can only select one action that is executable in the environment.

2.3 Providing feedback on training

In order to learn from training, it is important that trainees reflect on the events in a virtual training at some point. In traditional learning situations, instructors can monitor the performance of a trainee, provide feedback and explanations, and answer the trainee's questions. However, agent-based virtual training should be usable without continuous attention of an instructor. To encourage reflection on training by the trainee without an instructor, the training system should include feedback or explanation capabilities. In this section we will give an overview of providing feedback in intelligent tutoring systems, and discuss different choices to make when providing explanations in virtual training systems.

2.3.1 Intelligent tutoring systems

In the past twenty years, much research has been done on intelligent tutoring systems, which are systems that teach students how to solve a problem or execute a task by giving explanations (Polson and Richardson, 1988; Psotka et al., 1988; Murray, 1999). Van Lehn (2006) provides a general description of the behavior of tutoring systems. He describes tutoring systems as having two loops. The outer loop selects a task for the trainee, where a task usually consists of solving a complex, multi-step problem. The inner loop can provide feedback and hints to the trainee on each step. Van Lehn distinguishes three types of feedback. First, minimal feedback tells the trainee whether a step was correct, incorrect or not optimal. Second, error-specific feedback can be provided when the trainee makes an incorrect step, and aims to help the trainee to understand why his action was wrong. Third, hints on the next step can be provided when the trainee appears to get stuck. Furthermore, intelligent tutoring systems usually have an expert model and a user model. The expert model contains knowledge about how the task in the system should be performed. The user model contains descriptions of the trainee's past actions and his current estimated knowledge. In the inner loop, the actions of the trainee can be compared to the expert model, and based on that feedback can be provided. In the outer loop, a new task can be selected based on the estimation of the trainee's current knowledge.

Intelligent tutoring systems have been successfully designed for the training of well-structured skills and tasks such as LISP programming (Anderson et al., 1989) or algebra (Koedinger and Anderson, 1998). These tasks are relatively closed and involve little indeterminacy, meaning that there is only one or a few ways to reach the solution. Furthermore, in these tasks there is no need to deal with (unexpected) online events, and so they do not require real-time planning. Both of these task properties make it easier to provide feedback.

Attempts have been made to also build tutoring systems for training in open, complex and dynamic tasks like crisis management or military operations (Zachary et al., 1999; Livak et al., 2004). Designing training and feedback systems for such tasks poses new challenges. In traditional intelligent tutoring systems, as described above, a new task is selected for the trainee each time the previous one is finished. Training for complex and dynamic tasks, however, is often scenario-based. Instead of selecting a new task at fixed time points, the development of the scenario continuously has to be monitored and adjusted if necessary, e.g., by a director agent such as described in Subsection 2.4.2. Furthermore, providing feedback on the performance of a trainee in scenario-based training is also different from closed, well-structured tasks because often there is no single ‘right’ way to accomplish a task, but instead, there are multiple good solutions (Hutchins et al., 1999). This makes it particularly difficult to develop an expert model.

2.3.2 Providing explanations

As it is difficult to build intelligent tutoring systems for training of complex and dynamic tasks, alternative ways of evoking reflection on training have been proposed. As discussed, several approaches for explaining agent behavior have been approached. Jensen et al. (2005) proposed an explanation component that can provide explanations about important training points. With such explanation components, the trainee learns about the consequences of his actions through explanations about events and behaviors in the training scenario, instead of through direct feedback on his actions. Good explanations in virtual training can improve conceptual understanding (Van den Bosch, 1999), and prolong the duration of an acquired skill (Teichert and Stacy, 2002).

An explanation component for a virtual training system must of course be able to generate useful explanations. In Section 2.1 we discussed different types of explanations. Besides the content of an explanation, however, Nakatsu (2004) stresses that choices about the explanation interface and advisory strategies have to be made as well. In his framework, the interface of an explanation involves characteristics such as the ability to undo changes made to the system, and the ability for the user to request explanations with different levels of detail. Advisory strategies concern characteristics such as the timing of explanations, the provision mechanism, and the amount of explanations.

2.4 Agent-based virtual training

In the introduction chapter, we advocated to replace human players by intelligent software agents in virtual training systems in order to save costs and increase training flexibility. The use of intelligent agents in games and virtual training is a recent development (Dignum et al., 2009). Most games and virtual training systems are based on a scenario script describing the storyline of the game, including the behavior of the virtual game characters. Disadvantages of script-based virtual training are that (1) fixed storylines provide only little freedom for trainees to choose their actions, (2) they do not give much opportunity to show trainees the (long-term) effects of their actions, and (3) playing a scripted training twice will deliver nearly the same scenario.

Script-based virtual training becomes richer when the storyline branches in different directions, depending on the trainee's decisions and actions. Gordon et al. (2004), for instance, use scripted agents to play the main characters in a virtual training for leadership with branching scenarios. However, though branching storylines give the trainee more freedom, it tends to lead to an exponential growth of possible storylines as the scenario continues. All those possible storylines have to be devised in advance, which soon becomes unmanageable for a scenario writer.

An alternative to using a single scenario script is to represent the behavior of each virtual character in a separate cognitive behavior model that generates the character's behavior in real-time (Riedl and Stern, 2006). Agent technology can then be used to represent these behavior models. This way, not all possible courses of the scenario have to be written in advance, which makes it easier to provide the user with more freedom and achieve more diverse training scenarios. By allowing autonomy to the user as well as to the virtual agents, real interaction is accomplished. A second advantage is that the agents can easily be reused for different scenarios, as their behavior patterns remain equal over different scenarios. Examples of agents generating the behavior of virtual game characters are the Soar Quakebot for Quake II (Laird, 2001), a BDI Quake agent (Norling, 2003), a BDI agent for naval training (Van Doesburg et al., 2005) and a BDI agent in the Unreal tournament game engine (Davies and Mehdi, 2006).

Agent-based virtual training has strong advantages over script-based training, but also introduces two challenges. First, when agent technologies are used to implement the behavior of virtual characters, a connection between the agents and their virtual environment needs to be established. In Subsection 2.4.1, we will discuss some issues and possible solutions related to that. Second, a higher degree of autonomy for the trainee and agents makes it more difficult to maintain control over the storyline of a scenario, whereas such control is often desired for achieving efficient training. Thus, on the one hand, autonomy for agents and trainee is desired because it yields more natural behavior, but on the other hand, control over their behavior is desired to guide the storyline. This phenomenon is called the narrative paradox (Aylett, 1999). In Subsection 2.4.2, we will discuss the narrative paradox and show solutions suggested to solve the paradox.

2.4.1 Connecting agents and simulations

We use the term simulation to refer to the virtual environment of a game or virtual training, e.g., a house on fire, a room in which a crisis is managed, or a battle field. The simulation is usually implemented in a game engine and involves the visualization of the virtual world and representation of object properties, e.g., fires are extinguished by water and walking through walls is impossible. When connecting agents to a simulation, issues related to information representation, communication, and synchronization need to be solved (Dignum et al., 2009). We will provide an example of a potential problem for each of these three aspects.

Information is represented in different ways in agents and simulations. Agents reason with high-level concepts, such as beliefs and goals, for instance, they reason in terms of going to a building, finding a fire and extinguishing it. Simulations, in contrast, process more detailed information such as the exact positions of all the entities and objects in the game at every time-step. When an agent wants to execute an action, e.g., going to a building, its abstract decision needs to be translated to low-level descriptions required by the simulation, e.g., the coordinates of the agents starting position, exact path and final position. Similarly, the low-level information that is made available by the simulation is not immediately useful to the agents and must first be translated to more abstract concepts that are perceivable and understandable for the agent.

Most agent platforms offer facilities for the communication among agents. However, when the communication between agents is exclusively mediated by these facilities, unrealistic situations may arise. For instance, if there is a large amount of noise in an environment, agents should not be able to understand each other. However, the noise is only represented in the simulation, and not in the agent platform. So when the agents would communicate without interference of the simulation in this situation, they would seem telepathic.

When agents and simulation run on separate platforms, it is important that information is passed regularly and timely between the two to prevent synchronization problems. For instance, if an agent walks towards a burning building to extinguish the fire, but suddenly something inside the building explodes, it should immediately be able to turn and run away from the building. To do so, the simulation must send updates to the agent continuously so that the agent perceives its environment without a delay and can reason with adequate information. Furthermore, when the agent sends a message to the simulation to perform a certain action, the simulation should immediately process this action. Without such continuous synchronization, the agent in the example would run away way after the explosion, which is not realistic.

To deal with these issues, a middle-layer between agents and simulation can be used (Dignum et al., 2009). Such a layer is responsible for the translation of actions of the agent into actions within the simulation and translations of changes in the environment into percepts that can be handled by the agent. Furthermore, the communication between agents should be mediated by the simulation to overcome communication problems such as described above.

2.4.2 Scenario direction

Goal-directed, systematic training is more effective than ‘free play’, or learning-by-doing only (Blackmon and Polson, 2002). In order to make learning purposive and goal-directed, events in the simulation as well as the behavior of key players need to be carefully managed (Cannon-Bowers et al., 1998; Fowlkes et al., 1998). Players in the scenario should respond realistically to any situation emerging from the trainee’s actions, and the responses should keep the scenario on track of the learning goals. Moreover, the difficulty level of training should be adapted to the trainee’s competences so that training is challenging, but not too difficult (Westra et al., 2011).

In an agent-based approach to virtual training, the control over the development of a scenario is spread over several components. The scenario writer determines the initial situation and key events that will occur in the training, the agents and the trainee autonomously make their own decisions, and the simulation determines the effects of their actions in the virtual environment. Thus, the different elements comprising the virtual training system have a certain degree of freedom, but as they interact, they also influence each other. This interaction makes it difficult to predict the course and outcome of a scenario.

On the one hand, the freedom to act for agents and trainee is desirable, but on the other hand, it may create uninteresting training situations for the trainee. An instance of the latter is a trainee who makes an assessment error at the beginning of the situation, and, e.g., decides to deploy just one fire engine because of which a fire cannot effectively be extinguished. The trainee will later realize that he should have called for more resources, but missed many opportunities of dealing adequately with the events in the scenario. This problem of balancing the intentions of a story writer or designer with player freedom, the narrative paradox, is well recognized in the literature on interactive narratives (Aylett, 1999; Louchart and Aylett, 2005; Lockelt et al., 2005; Riedl and Stern, 2006; Swartjes and Theune, 2009).

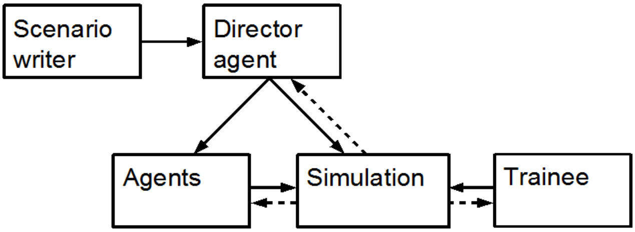


Figure 2.4: The elements in an agent-based virtual training.

More control over the course of the scenario in agent-based virtual training can be provided by a director agent (see, e.g., Van den Bosch et al., 2009; Riedl and Stern, 2006; Peeters et al., 2011). The concept originates from studies into interactive narratives where story directors or drama managers are used to maintain an interesting storyline (Louchart and Aylett, 2005; Lockelt et al., 2005). A director agent can be considered as an agent ‘behind the scene’. The director agent directs the agents and the simulation if the scenario moves in an unwanted direction. For instance, if a trainee lacks to call for extra fire engines, but that is crucial for an interesting course of the scenario, the director agent can let one of the trainee’s team members advise the trainee to call for extra fire engines. Thus, most of the time, the agents autonomously determine their actions, but if the scenario requires specific agent behavior to bring about a desired state, the agent receives instructions from the director agent to do so. Figure 2.4 shows the relations between a director agent, scenario writer, agents, simulation, and trainee in a virtual training system (adopted from Van den Bosch et al., 2009).

2.5 Chapter conclusion

In this chapter we provided a background for the rest of this thesis. We gave an overview of literature on explanation, introduced BDI-based agent programming, discussed the provision of feedback in virtual training, and discussed the use of intelligent agents in virtual training in general. With this, the ground is laid to introduce our approach to developing explainable BDI agents. In the next chapter, we will describe the general explainable agent approach of this thesis. In subsequent chapters, we will further refine and extend this general approach.

Chapter 3

Developing explainable agents

This chapter describes an approach for developing software agents that can explain their own behavior. The agents in this approach are modeled as BDI (Belief Desire Intention) agents (Rao and Georgeff, 1991, 1995). The behavior of BDI agents is represented by mental concepts such as beliefs, goals, and intentions. BDI agents determine their actions by a reasoning process on their mental concepts. There are several BDI-based agent programming languages that allow for the implementation of such agents (Bordini et al., 2005, 2009).

The choice for BDI agents is motivated by two observations in existing explanation research. As described in the previous chapter, psychological research shows that people explain and understand human (or human-like) behavior in terms of mental concepts like beliefs, goals, and intentions. We aim to develop explainable agents for virtual training that display human behavior, and therefore we believe that their explanations should exist of mental concepts as well. Second, explanation research in the field of AI shows that the behavior of intelligent systems is often (partly) explained by the steps that lead to it. This is efficient, representations that are used for the generation of behavior can also be used for the generation of explanations, but requires that behavior representations are meaningful for users. So when we want our agents to generate human-like explanations, their behavior should be represented by beliefs, plans, goals, and the like. BDI agents satisfy this requirement.

In this chapter we will first introduce a way to construct a behavior representation of an explainable BDI agent. Subsequently, we discuss its implementation in a BDI-based agent programming language, and show how such a BDI agent can be extended with explanation capabilities. Though the addition of explanation capabilities is only discussed at the end, the purpose of explanation is taken into account in earlier steps as well.

3.1 Constructing a BDI agent model

At the beginning of this thesis we described a training scenario for firefighting. The trainee in this scenario interacts with virtual agents playing the trainee's team members. To display realistic firefighting behavior under different circumstances, these agents need to be equipped with domain specific task knowledge, e.g., about extinguishing different fires or communication procedures. So-called Subject Matter Experts (SMEs) can provide such expert knowledge to agent developers, who in turn have to translate it into a BDI agent model. Norling (2004) points out that because experts tend to explain their actions in terms of beliefs, goals, and intentions, BDI models offer a natural way to capture expert knowledge. Still, the translation from expert knowledge to a BDI representation is not trivial and there is no commonly accepted methodology for constructing a BDI agent model.

We use hierarchical task analysis (HTA) to guide the process of knowledge elicitation from subject matter experts. HTA is a well established technique for cognitive task analysis in cognitive psychology, and has proven to be appropriate for the specification of complex human tasks (Schraagen et al., 2000). More specifically, HTA is the process of constructing a hierarchical task network by dividing a higher-level task into subtasks until the level of actual actions is reached. For instance, the main task of a firefighter is to take action when there is an incident, which can be divided into subtasks such as saving victims, extinguishing fires, and ensuring safety of bystanders, which in turn can be divided into more detailed subtasks. HTA involves the knowledge, thought processes, and goal structures that underlie observable task performance (Shepherd, 1998). It thus smoothly connects internal cognitive processes (such as goals and knowledge) to observable behavior (actions), which corresponds well to our goal of explaining the observable by the internal.

HTA has been examined extensively, and there is a great deal of documentation on it available. Therefore, in this thesis we do not focus on HTA itself, but we do describe what its result -a task hierarchy- should look like in our approach. In this section we will first discuss GPGP, one of the most extensive accounts of planning based on hierarchical task networks, on which we inspired our approach. Subsequently, we discuss the similarities between task hierarchies and BDI models. Finally, we introduce the goal hierarchy model which forms the core of our approach.

3.1.1 GPGP

The GPGP/TAEMS approach (Generalized Partial Global Planning / Task Analysis, Environment Modeling and Simulation) (Lesser et al., 2004) is currently one of the most extensive accounts of general (hierarchical task network) planning. Many accounts of planning in artificial intelligence are based on hierarchical task representations, usually called hierarchical task networks (Russell and Norvig, 2003). In the strict sense, a hierarchical task network is the decomposition of an abstract task into more detailed subtasks. Many accounts of hierarchical task networks, like GPGP, have additional features, e.g., information about which subtasks to select under given circumstances.

GPGP is a framework for the coordination of small teams of agents, and it makes use of local (from the perspective of one agent) and non-local task structures. TAEMS is the language used to represent these task structures. In the GPGP approach, coordination and scheduling are distinguished, where coordination refers to planning among agents and scheduling to the planning within an agent. The underlying model of the GPGP approach can be represented conceptually as an extended AND/OR goal tree where the leaves of a tree are primitive (non-decomposable) actions. A goal can have any number of subgoals.

GPGP distinguishes two types of coordination relationships between the nodes of a tree: task-subtask relations and non-local effects. For the task-subtask relations, Quality Accumulation Functions denote how the success of a main task is determined by its sub-tasks, where the success of a task is a gradual measure. A task-subtask relation is for instance *Qmin*, which means that the quality of the main task is the minimum single quality of all subtasks. Figure 3.1 gives an overview of all Quality Accumulation Functions distinguished by GPGP.

QAF	**Description**
Qmin	minimum single quality of all subtasks
Qmax	maximum single quality of all subtasks
Qsum	total aggregate quality of all subtasks
Qlast	quality of most recently completed subtask
Qsumall	as with Qsum, except all subtasks must be completed
Qseqmin	as with Qmin, except all subtasks must be completed in order
Qseqmax	as with Qmax, except all subtasks must be completed in order
Qseqlast	all subtasks must be completed in order, and overall quality is the quality of last task
Qseqsum	as with Qsum, except all subtasks must be completed in order
Qexactlyone	quality of single subtask, only one subtask may be performed

Table 3.1: Quality Accumulation Functions (QAFs) distinguished by GPGP.

The second collection of coordination relationships in GPGP exists of non-local effects, which are relations between two tasks at any place in the task hierarchy. Relations between more than two tasks are not used in GPGP. Non-local effects mentioned in the GPGP literature are: *inhibits*, *cancels*, *precedes*, *constrains*, *causes*, *enables*, *facilitates*, *hinders*. However, not all of them are worked out in detail. Of all coordination relations, *enables* and *facilitates* receive most attention in the GPGP papers. *Enables* is a hard constraint (the result of one problem-solving activity is required to perform another), and *facilitates* is a soft constraint (the result of one problem-solving activity may be beneficial but is not required to perform another). GPGP mainly focuses on non-local effects between tasks situated at different agents, since these define potential places for coordination.

Task hierarchy	BDI agent
State	Beliefs
Main task	Goal
Subtask	Goal or plan
Primitive task	Action (atomic plan)

Table 3.2: Task Hierarchies versus BDI agents.

3.1.2 Task hierarchies versus BDI models

Sardina et al. (2006) pointed out that hierarchical task networks have similarities with BDI models. The most important similarity is that both reduce high-level entities into lower-level ones. Table 3.2 shows the correspondence between the concepts used in task hierarchies and those in BDI agent models. For most task hierarchy concepts, the mapping to a related BDI concept is clear. However, whereas a task hierarchy only contains tasks, BDI models make a distinction between goals (desired world states) and plans (sequence of actions or sub-plans describing how a goal can be achieved). An agent's main task must be implemented as a goal because otherwise no plans are generated and the agent does not act. Primitive tasks must be implemented as (atomic) plans, again, because otherwise the agent does not act. But for tasks that are not main or primitive tasks, it is not immediately obvious whether they should be translated to either plans or goals.

A first difference between plans and goals in BDI agents is the way they are removed from an agent's mental state. Most BDI-based programming languages, e.g., AgentSpeak (Bordini et al., 2007), 2APL (Dastani, 2008), and GOAL (Hindriks, 2009), are governed by the *rationality principle*, which means that an agent cannot believe a certain fact and pursue that fact as a goal at the same time. Usually, a goal stays in an agent's goal base until the agent obtains the belief that the goal is achieved. Goals can also be explicitly dropped by the agent (as part of a plan). Plans, in contrast, are removed from an agent's plan base once they are executed. As a consequence, goals are more appropriate for the implementation of tasks which are achieved by an unknown number of actions (depending on the environment), e.g., monitoring plan execution.

A second difference between plans and goals concerns the way in which they are executed or achieved. The deliberation cycle of an agent determines which step the agent should perform next, e.g., execute an action or apply a reasoning rule. In 2APL (Dastani, 2008) for instance, all rules that are applicable to goals (PG-rules) are tried to be applied in the deliberation cycle. But for plans, in contrast, it is considered per plan in the plan base which rules (PC-rules) apply to it. Thus, the order of goal execution depends on the order of the rules, whereas the order of plan execution depends on the order of the plans in the agent's plan base. As a programmer it is easier to exert control over the order of rules than over the order of plans in a plan base because an agent's rules remain the same, but its plans change during program execution. This is similar for other BDI-based agent programming languages.

To conclude, for domains in which the number and order of tasks to be executed is

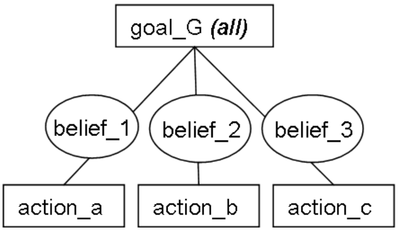


Figure 3.1: Example of a general goal hierarchy.

fixed, it is easier to implement tasks as plans because the agent program ensures that plans are executed in the given order and dropped after execution automatically. In general, however, implementing tasks as goals allows for more flexibility because the number and order of actions to achieve a goal may vary. Therefore, we have chosen to implement all tasks in a hierarchy as goals, except for primitive tasks. This makes that the underlying structure of our BDI agents is a goal hierarchy.

3.1.3 Goal hierarchy

In this subsection we introduce a goal hierarchy representation language defining a goal hierarchy structure for explainable agents. The language is a simplified version of GPGP. In contrast to GPGP, we do not have non-local effects. Non-local effects are used for coordination among different agents in GPGP, and we take a single agent perspective only. Furthermore, whereas GPGP uses a gradual scale for the achievement of goals, in our approach a goal is either achieved or not. There are several reasons for not allowing partially successful task execution. We give three of them. First, it is often hard to determine the measure of success of a task execution. Second, the task domains we aim at are of a procedural nature; next actions are selected based on procedures that only distinguishe whether a task is executed satisfactorily or not. Third, the easiest way to represent partially succeeded tasks would be with a numerical approach, which makes it more difficult to provide explanations.

We define a goal hierarchy as a number of goals which are related to each other by goal-subgoal relations. A goal is defined as *G(Relation,[(G1,B1),...,(Gn,Bn)])*, where *Relation* denotes the relation of goal *G* to it subgoals *G1,...,Gn*, and beliefs *B1,..,Bn* indicate the conditions under which the respective subgoal *G1,...,Gn* can be adopted. Subgoals can in turn be decomposed into further subgoals, etc. There are four goal-subgoal relations, namely *all*, *one*, *seq* and *if*. The leaves of the goal hierarchy are formed by actions that can be executed in the environment. Actions are represented as goals that do not have a relation and are not decomposed into subgoals *G(−,[])*. Figure 3.1 shows an example of a simple goal hierarchy, and Table 3.3 shows its formal representation.

Goals can be adopted, which means that they are tried to be achieved, and dropped, which means that they are either achieved or no longer tried to be achieved. For all goals but the top goal it holds that a goal Gi can only be adopted when its parent goal is

```
goal_G(all,[(action_a,belief_1), (action_b, belief_2),
      (action_c,belief_3)])
action_a(-,[ ])
action_b(-,[ ])
action_c(-,[ ])
```

Table 3.3: The example goal hierarchy in formal notation.

adopted, and the conditions *Bi* are believed to be true. Since we do not allow for gradual goal completion, we have less goal-subgoal relations than in GPGP. The different goal-subgoal relations *all*, *one*, *seq* and *if* describe when a goal is achieved. A goal with a relation of the type *all* is achieved when all its subgoals are achieved. A relation of the type *one* implies that a goal is achieved when exactly one of its subgoals is achieved. The term *seq* refers to sequential and it means that a goal is achieved when all subgoals are achieved, in a specific order. A goal with a relation of the type *if* is achieved when all *applicable* subgoals are achieved. Dependent on the beliefs of the agent, this may be all, some or none of the subgoals.

When a goal is believed to be true, it is achieved and can be dropped. A goal can become achieved because (1) the agent executed certain actions, or (2) through events in the environment. For instance, the goal to extinguish a fire is achieved and can thus be dropped when the agent observes that the fire is out. Sometimes the environment does not immediately give feedback on whether a task has been performed successfully. For instance, the task to report something can be achieved by sending an email, but one does not directly know if the email is read and understood. In such a case we use task execution to determine task performance, that is, if a task is executed it is assumed to be achieved.

3.2 Implementation of explainable agents

In this section we will start with an example by providing the goal hierarchy of a firefighter. Then we will show how this and other goal hierarchies can be implemented as BDI agents. For that, we use the BDI language 2APL (Dastani, 2008).

3.2.1 Goal hierarchy of a firefighting agent

The example in this section involves an agent that is part of a virtual crisis management training. In this training, the trainee is playing the head of a crisis management team that is confronted with a crisis. The trainee has to instruct and monitor the leaders of several teams, e.g., a firefighting team, a police team, and a health care team, which are played by intelligent agents. The agent in this example is a leading firefighter of which the tasks are to receive an attack plan from its head (the trainee), pass corresponding instructions to the firefighting team, monitor plan execution, and finally, report to the head that the incident has been solved. Figure 3.2 shows a picture of a part of the goal hierarchy of the firefighting agent. The boxes and circles represent the agent's goals and beliefs, respectively.

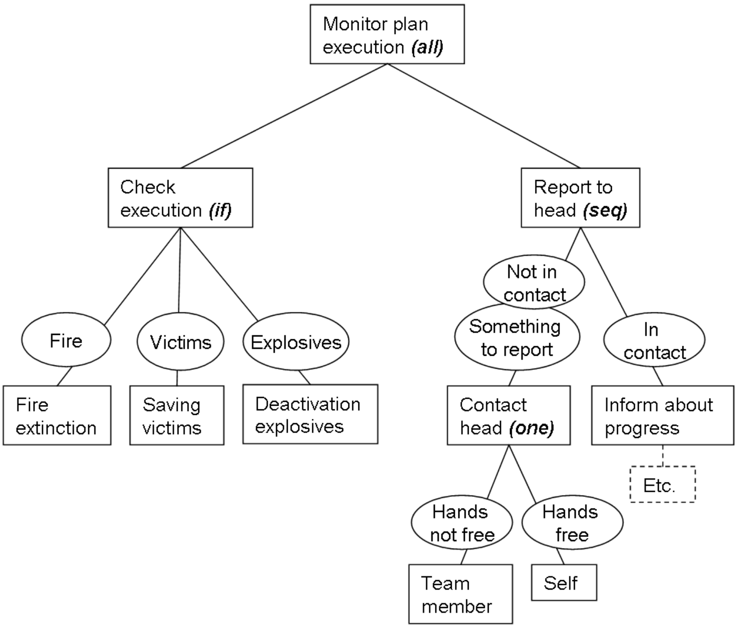


Figure 3.2: Part of the firefighter's goal hierarchy.

Table 3.4 shows the goal hierarchy of firefighter agent in Figure 3.2, in the formal representation introduced in Subsection 3.1.3. The figure and representation show that the agent's two main activities are checking the plan execution by the team, and reporting its observations to the head. Usually, the firefighter agent reports several times to the head during one incident. The *all* relation of the goal Monitor denotes that the Check(X) and Report goals are both adopted. The achievement of goal Check(X) depends on the circumstances of the incident, shown by an *if* relation. If there are fires, victims or explosives, the agent has to check fire extinction, saving of victims, or the deactivation of explosives, respectively. To report, the firefighter has to contact the head and report about the different aspects of the incident. These two goals have to be performed one by one and in this order, which is denoted by a *seq* relation. The fire-fighter agent can either contact the head on his own, ContactHead(Self), or let a team member do so, ContactHead(Member). The *one* relation represents that executing one of these options is sufficient to achieve the goal ContactHead(Y).

As mentioned at the beginning of this section, we do not focus on the construction of a goal hierarchy itself. The goal hierarchy of an agent, however, does determine the explanations that it will provide. For now, we assume that the available HTA methods

```
Monitor(all,[(Check(X),true),(Report,true)])
Check(X)(if,[(Check(Fire),fire),(Check(Victims),victims),
       (Check(Explosives),explosives)])
Check(Fire)(−,[ ])
Check(Victims)(−,[ ])
Check(Explosives)(−,[ ])
Report(seq,[(ContactHead(Y),somethingToReport and not inContact),
       (Inform,inContact)])
ContactHead(Y)(one,[(ContactHead(Member),not handsFree),
       (ContactHead(Self),handsFree])
ContactHead(Member)(−,[ ])
ContactHead(Self)(−,[ ])
Inform(−,[ ])
```

Table 3.4: The firefighter's goal hierarchy in formal notation.

are followed and that the resulting goal hierarchy represents an agent's tasks as well as possible, but we will get back at this point in Sections 4.4 and 7.3.

3.2.2 From goal hierarchy to 2APL agent

In this section we discuss the implementation of a goal hierarchy to 2APL code. For each of the goal types (all, if, seq, one) we show how the goal hierarchy representation can be implemented. To ensure that the program covers the goal hierarchy as desired, we use the fact that the interpreter considers PG-rules from top to bottom. More specific rules are implemented above (and thus tried before) more general rules, so that the most specific rule as possible is always applied.

All relation For goals of the type *all*, all subtasks are adopted. In the implementation, a PG-rule is added for each subgoal, thus an *all*-goal with n subgoals is implemented by n PG-rules. In our example, the goal Monitor has an *all* relation to its subgoals: Monitor(all,[(Check(X),true), (Report,true)]). The 2APL implementation of this representation looks as follows.

```
Monitor <- true | adopt(Check(X))
Monitor <- true | adopt(Report)
```

The first part of a rule is a check on the agent's goal base. Both PG-rules in this example are only applied if Monitor is one of the agent's goals. The second part of the PG-rule is a check on the agent's belief base. Here, the guards of both rules are always true, i.e., the adoption of the subgoals does not depend on the agents beliefs. The bodies of the two rules state that the goals Check(X) and Report have to be adopted, respectively.

Goals can be achieved (1) because a certain situation in the environment is true, or (2) because its subgoals have been achieved. In this paragraph we provide an example of the second situation. The next paragraphs (under if relation) show how to implement the first situation. To implement goal dropping resulting from achievement of subgoals, we make use of the rationality principle in 2APL, i.e., if a goal (a desired world state) is believed to be true, that goal is dropped. Thus, ensuring that a goal is dropped when both its subgoals are achieved can be implemented by the following code.

```
Monitor :- Check(X), Report.
```

This line states that when both the goals Check(X) and Report are achieved successfully, the goal Monitor can be dropped as well.

If relation For goals of type *if*, all applicable subtasks are adopted. In our example, the goal Check(X) has an *if* relation to its subgoals. The expression Check(X)(if, [(Check(Fire),fire), (Check(Victims),victims), (Check(Explosives),explosives)]) is implemented as follows.

```
Check(X) <- fire       | Check(fire)
Check(X) <- victims    | Check(victims)
Check(X) <- explosives | Check(explosives)
```

The first part of these three PG-rules says that the rules are only considered if the agent has the goal Check(X). The second part of these rules involve a check on the agent's belief base. In these rules it is checked whether the agent has the beliefs fire, victims, and explosives, respectively. If the agent indeed has (one of) these beliefs, that particular PG-rule fires, and the agent executes (one of) the actions Check(fire), Check(victims), and Check(explosives). Note that the subgoals of Check(X) are actions, and thus implemented as plans. If it would have been subgoals, the adopt(subgoal) action would have been used, like in the example with the all relation.

The goal Check(X) can be dropped when there are no more fires, victims, and explosives. This can be expressed by the following 2APL code.

```
Check(X) :- not fire, not victims, not explosives.
```

The rule ensures that if all subgoals are no longer applicable (e.g., the fire has been extinguished, the victims are saved and the explosives are deactivated), the agent starts to believe Check(X) and thus the goal Check(X) is dropped.

Actions are implemented as plans (Check(fire), Check(victims), Check(explosives)), and therefore automatically removed from the agent's plan base once executed. It may happen that the goal for which an action was executed is not achieved yet after its execution. In that case, the action will be adopted and executed again. For instance, to extinguish a fire, it may be necessary to use the content of several fire-extinguishers. Besides goals that are only dropped when certain conditions in the environment become true, there are goals that are always achieved by executing one or several actions. For example, the goal to report something to the dispatch center is always removed after the action of sending a message. For such goals, the last action to achieve the goal should also add a belief to the agent's belief base which indicates that the action has been executed.

Seq relation For goals of the type *seq*, all of their subgoals are adopted, but one by one and in a specific order. For example, the goal Report(seq, [(ContactHead(Y), somethingToReport and not inContact), (Inform, inContact)]) is implemented as follows.

```
Report <- somethingToReport,
                    not inContact | adopt(ContactHead(Y))
Report <- inContact               | Inform
```

Again, the heads of the rules contain the goal for which subgoals need to be achieved. The guards of the rules contain the conditions under which a rule can be adopted. For goals of type *seq*, the conditions specify unique circumstances, so that only one subgoal is executed at a time. Because the subgoals must be achieved in a specific order, the guards of the rules often specify the result of the previous goal in the sequence. Here, for instance, the firefighter agent only starts to inform the head when it believes it is in contact.

There is one exception. Namely, if the subgoals of a goal of type *seq* are actions, only one PG-rule is needed. For instance, a goal with three actions which have to be executed one by one in a fixed order is implemented as follows.

```
Head <- Guard | { Action1; Action2; Action3 }
```

As the order of the actions is fixed and they can be executed immediately, it is not necessary to use different PG-rules. The actions are added to the agent's plan base, and automatically executed in the right order.

In general, a goal with relation *seq* can be dropped if its last subgoal is achieved. In this case, the result of the last action Inform is that there is no longer something to report. The goal Report can thus be dropped by the following code in the agent's belief base.

```
Report :- not somethingToReport.
```

Note that it is not possible to add the belief Inform to the agent's belief base to drop the goal Report. Namely, the agent may need to report to the head several times, and therefore it has to be able to drop the goal Report again.

One relation For goals of type *one* it holds that only one of their subgoals is adopted. A *one* goal with n subgoals is implemented by n PG-rules. The goal ContactHead(Y)(one,[(ContactHead(Member),not handsFree), (ContactHead(Self),handsFree]), for example, is implemented as follows.

```
ContactHead(Y) <- not handsFree | ContactHead(member)
ContactHead(Y) <- handsFree     | ContactHead(self)
```

The guards of both rules, not handsFree and handsFree, denote exclusive situations to ensure that only one subgoal is adopted.

The goal ContactHead(Y) can be dropped when the agent believes that it is in contact with the head. This is implemented in the agent's belief base as follows.

```
ContactHead(Y) :- inContact.
```

In this example, the goal ContactHead(Y) is achieved because of a change in the environment. When a goal of type *one* is achieved because of the execution of one of its subgoals (with no immediate observable effects in the environment), two separate beliefs are needed to express the dropping of the main goal. Namely, a *one* goal is achieved by the achievement of only one of its subgoals.

3.3 Explanation of agent behavior

In this section we discuss how explanation capabilities can be added to BDI agents. In order be explainable, an agent should fulfill several requirements. We mention three of them. First, an agent must have an explanation-friendly behavior model, meaning that the concepts and processes generating the agent's behavior are understandable and meaningful for humans. In our approach this requirement is satisfied by using a BDI model for the representation of agent behavior. Second, an explainable agent should be able to introspect. Namely, an agent needs to have knowledge about its own states and processes in order to explain them. BDI agents are able to check their own belief and goal base, but only by performing explicit check actions. Third, explainable agents need to have an episodic memory. To explain its actions, an agent needs to know about its states and processes not only at the time they occur, but also at later points in time. BDI agents normally do not have an episodic memory. Most BDI-based programming languages create a log of the agent's reasoning steps and actions. However, these logs are created for debugging purposes and often not accessible by the agent for introspection. Moreover, the information in such logs is usually not organized in a practical way for generating explanations.

One way to provide an agent with an episodic memory is to log the agent's reasoning steps within its own program, e.g., by adding update actions to store the agent's reasoning steps and actions in its belief base. The disadvantage of this method is that the bookkeeping necessary for the explanation can interfere with the generation of 'real' plans and actions. For this reason, we have chosen to build a separate explanation module to store the information that may be needed for explanations.

Figure 3.3 shows the architecture of a BDI agent (as introduced in Subsection 2.2.1) extended with an explanation module. The explanation module contains a behavior log for remembering the agent's past actions and motivations, and an explanation generator that applies explanation algorithms to the behavior log to generate explanations when the agent receives an explanation request. We implemented the explanation module in 2APL, where the module was implemented as a 2APL environment (Harbers et al., 2010b).

3.3.1 Behavior log

When a BDI agent is executed, the agent perceives its environment and accordingly updates its beliefs. In the approach presented here, an agent's beliefs determine how it "walks through" its goal hierarchy. To provide explanations about the agent's actions,

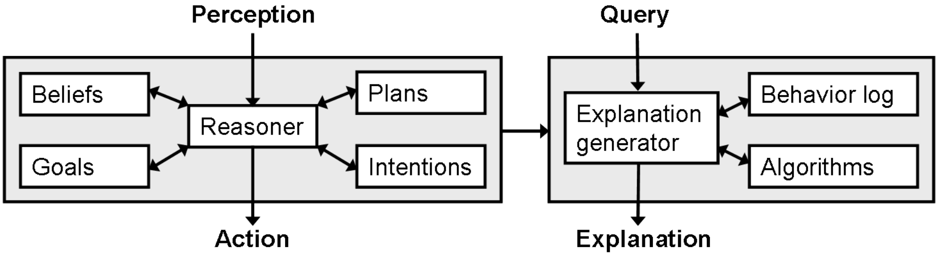


Figure 3.3: BDI architecture with an explanation module.

this path with reasoning steps and actions needs to be remembered. In most BDI-based programming languages, debugging tools allow for the observation of an agent's internal states during runtime, and browsing through a trace of the program execution during and after runtime. In principle, the execution traces should give sufficient information to find semantic bugs in an agent program. However, current debugging tools do neither provide help on finding the right information in the execution trace, nor are they organized in a practical way for explanation purposes. Therefore, an agent's reasoning steps and actions are updated to the explanation module during the execution of the agent, and thus a behavior log is created.

The explanation module can generate explanations about the behavior of agents with different goal hierarchies, implemented in different BDI-based programming languages. But though the content of the information updated to the behavior log may differ per agent, the structure of the updates has to be the same. A fixed update structure ensures that the agents' updates are 'understood' by the explanation module, or in other words, that they fit in the organization of the behavior log. For each goal or action that an agent adopts, the behavior log is updated with the following information.

```
< goal/action, parent, relation, [belief 1,...,belief n],
           [child 1,...,child n], time(adoption,dropping) >
```

The update is a tuple containing, in this order, the identity of the goal or action (e.g., the goal ContactHead(Y) from the goal hierarchy in Figure 3.2), the identity of its parent goal (Report), the relation between the goal/action and its parent (seq), its adoption conditions (somethingToReport, not inContact), the identities of the child goals/actions of the parent goal (ContactHead(Y),Inform), and the time at which is was adopted and dropped (t(adopt,drop)). Note that the list with children also includes the goal/action itself. The order of execution of the children is maintained in the behavior log.

An agent's goal hierarchy contains all goals and actions that the agent can possibly adopt. Consequently, in the BDI implementation, all goals that may be adopted are present in the plan library of the agent. A behavior log, in contrast, does not contain a specification of the agent's goal hierarchy. Before execution of the agent, the behavior log is empty. After execution of the agent, the behavior log contains a set tuples containing

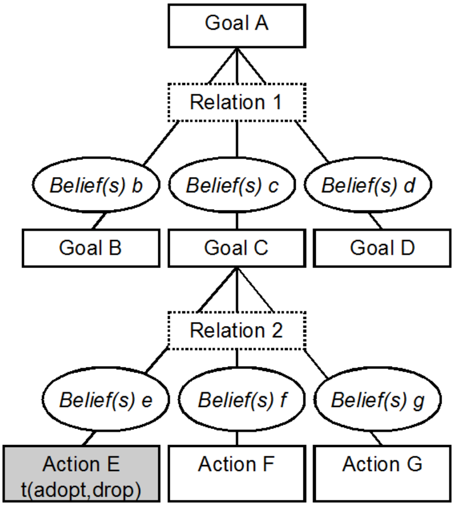


Figure 3.4: Goal hierarchy of an explainable BDI agent.

the goals/actions that the agent actually adopted and performed. The goals and actions in the behavior log are equal to, or a subset of the goals and actions in an agent's goal hierarchy. With the information in the behavior log, the 'used' part of the goal hierarchy can be reconstructed.

3.3.2 Explanation algorithms

We developed several explanation algorithms that can be applied to the behavior log. When a user sends a request for an explanation about a certain agent action to the explanation module, one or more of the explanation algorithms is (are) applied to select information from the behavior log. The result of this process, an explanation for the given agent action, is presented to the user. Note that explanations could be asked for during runtime, as the goal tree is built up continuously.

Actions can be explained in different ways, e.g., someone opened a door 'because he believed someone was outside' or 'because he wanted to know who was outside'. In our approach, an action can be explained by the beliefs and goals underlying that action. For example, in the goal hierarchy in Figure 3.4, when action e is executed, the agent must have had goals A and C, and beliefs c and e. However, providing the whole trace of beliefs and goals responsible for an outcome, in particular with big goal hierarchies, often does not create a useful explanation and may even result in an information overload. Keil (2006) noted that not all 'explaining elements' are equally useful in an explanation. Therefore, to generate useful explanations, explanation algorithms should not only select which goals and beliefs are responsible for an action, but also make a selection among those beliefs and goals.

Below, we introduce the following explanation algorithms: G+1, G+2, B+1, B+2, Gnext' and Gnext. The explanation algorithm Gnext' is an initial version of Gnext. Still, we do discuss Gnext' here because Chapter 4 describes an experiment in which Gnext' is used. In the following definitions, the A in action/goal$_A$ is a variable denoting the identity of the tuple in the log. Child$_A$ i refers to the i'th child of the parent of the action/goal in tuple A.

Algorithm G+1. According to the first algorithm, an action is explained by the goal directly above the action, i.e., the parent goal of an action.

```
if (Explain(action/goal$_A$)) then
    (explanation = parent$_A$ = action/goal$_B$)
```

Following this explanation algorithm, action E in Figure 3.4 is explained by goal C.

Algorithm G+2. Explanation algorithm G+2 explains actions by the goal two levels up in the goal tree, that is, the parent goal of the action's parent goal.

```
if (Explain(action/goal$_A$) and
    parent$_A$ = action/goal$_B$) then
        (explanation = parent$_B$ = action/goal$_C$)
```

In this case, action E is explained by goal A.

Algorithm B+1. Algorithm B+1 explains actions by the belief(s) that enabled the execution of that action.

```
if (Explain(action/goal$_A$)) then
    (explanation = belief$_A$ 1...n)
```

According this explanation algorithm, action E is explained by belief(s) e.

Algorithm B+2. This algorithm explains actions by the belief(s) that enabled the adoption of the parent goal of that action.

```
if (Explain(action/goal$_A$)) then
    parent$_A$ = action/goal$_B$) then
        (explanation = belief$_B$ 1...n)
```

According to explanation algorithm B+2, action E is explained by belief(s) e.

Algorithm Gnext'. Algorithm Gnext' checks if the relation of an action to its parent is of type *seq* and whether the action is not the last one of the sequence. If so, the action is explained by the next action or goal in the sequence, also called the *enabled* action or goal. If not, the algorithm does not generate an explanation.

```
if (Explain(action/goal$_A$) and action/goal$_A$ = child$_A$ i
    and relation$_A$ = seq) then
        (explanation = child$_A$ i+1)
```

Following algorithm Gnext', if relation 2 is of type *seq*, action E is explained by action F. If relation 2 is not of type *seq*, the algorithm does not generate an explanation.

Algorithm Gnext. Algorithm Gnext checks if the relation of an action to its parent is of type *seq* and whether the action is not the last one of the sequence. If so, the action is explained by the next action or goal in the sequence. If not, the algorithm tries to explain the parent of the action in a similar way. Each time no relations of type *seq* are found or the action/goal is the last in the sequence, the algorithm searches one level higher in the tree for an explanation. When the top goal is reached without finding an explanation the top goal is provided as an explanation. Thus, an action is explained by the first action or goal that must follow the action (independent of the conditions in the environment), and that becomes achievable or executable because of the action.

```
if (Explain(action/goal$_A$) and action/goal$_A$ = child$_A$ i
    and relation$_A$ = seq) then
        (explanation = child$_A$ i+1)
else if (Explain(action/goal$_A$) and (action/goal$_A$ =
    child$_A$ n or not relation$_A$ = seq)) then
        (Explain(parent$_A$))
else (explanation = action/goal$_A$)
```

Following algorithm Gnext, if relation 2 is of type *seq*, action E is explained by action F. If relation 2 is not of type *seq*, but relation 1 is, action E is explained by goal D. If both relation 1 and 2 are not of type *seq*, action E is explained by goal A.

The execution of explainable BDI agents that are designed as a goal hierarchy, also result into a tree-shaped structure in the behavior log, i.e., there is one main goal and each goal has a limited number of subgoals or actions. Other BDI agent programs may result into less regular tree shapes, e.g., one main goal with many subgoals, several separated trees when multiple independent initial goals are present, or several partly connected trees when multiple dependent initial goals are present. In principle, the explanation algorithms can be applied to all kinds of goal trees to generate explanations, but in this thesis we focus on tree-shaped structures.

3.4 Chapter conclusion

In this chapter we presented an approach for developing explainable agents. First, we provided a method for constructing an explainable BDI agent model. Then, we showed how such models can be implemented in 2APL. Finally, we introduced an explanation module that adds explanation capabilities to BDI agents.

We may conclude that an action often is the result of many underlying beliefs and goals, but it is not desirable to provide the whole trace in an explanation for that action. However, we do not know which beliefs and goals form the most useful explanations of an action, and thus, which explanation algorithms can best be applied. Psychological research provides no conclusive answers to which explanation types should be used to explain actions. Malle (1999), who proposed a framework about how people explain behavior (Subsection 2.1.1), states that people mostly give reason explanations for intentional behavior. He distinguishes beliefs and goals as reason explanations, but does not (yet) describe when and how which goals and beliefs are used. Therefore, in the next chapter, we will present three user studies that investigate which explanation types people consider most useful.

Chapter 4

Studying user preferences for explanations

In this chapter we present three studies evaluating the approach for developing explainable agents introduced in the previous chapter. In all studies, the subjects are provided with a training scenario, and then asked to provide, select or judge explanations for several of the actions of the player(s) in the scenario. The explanations presented to the subjects are generated according to our approach to explainable agents. In the studies, we investigate the preferences of instructors (Harbers et al., 2009e), novices (Harbers et al., 2010b), and experts (Broekens et al., 2010a). The domains in the studies are training for onboard firefighting, firefighting, and cooking, respectively. In the last section of this chapter, we discuss all results and provide general guidelines for the design and explanation of agent behavior (Harbers et al., 2010a).

4.1 Instructors' preferred explanations

The purpose of the first study is twofold. First, we want to examine whether instructors consider intentional explanations about agent behavior, i.e., explanations in terms of beliefs and goals and the like, useful for learning from virtual training. We believe that the opinion of instructors is important as they have knowledge on both the task domain and didactic aspects. If they do consider the explanations useful, then second, we want to investigate which types of intentional explanations they consider most useful. We will consider three properties of explanations: explanation length, abstraction level, and mental concept type. For instance, should the explanations be long or short, contain detailed or more general information, and consist of beliefs or goals?

In Chapter 2 we showed that human behavior is usually explained and understood by intentional explanations (Dennett, 1987; Malle, 1999; Keil, 2006), and we also mentioned that intelligent agents in virtual training usually display human behavior. Therefore, our proposition in this study is that intentional explanations about agent behavior in virtual training will be considered useful. Concerning explanation length, we hypothe-

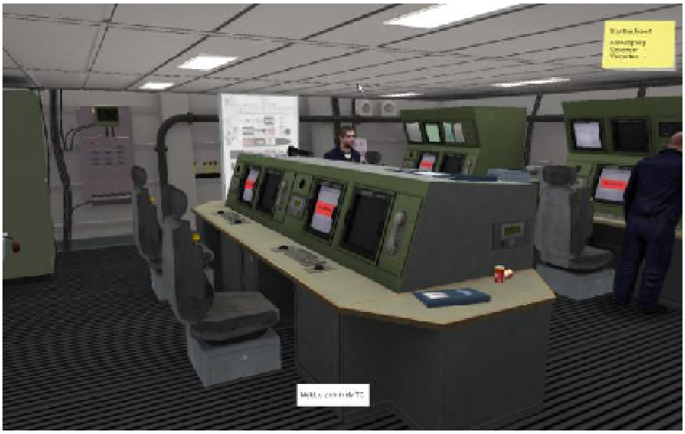


Figure 4.1: The CARIM virtual training system.

size that short explanations will be preferred over long ones because not all information explaining an action is relevant. This corresponds to earlier observations by others (Keil, 2006). Concerning abstraction level, our proposition is that explanations containing more detailed, low-level information will be preferred over explanations with more abstract, high-level information. We expect that trainees often know about the general intentions of players in a certain domain before they learn what happens on a more detailed level. And finally, concerning mental concept type, our proposition is that explanations in terms of beliefs and explanations in terms of goals will both be considered useful. More specifically, we expect that beliefs with a condition under which an action can be performed and goals that are achieved by executing an action are considered useful explanations for those actions.

4.1.1 Method

Domain and training task. The first study took place in the navy domain and involved a training for Officer of the Watch. An Officer of the Watch is the person in command when there is a fire aboard a navy frigate. We chose this domain because we expect explanations to be particularly useful in complex and dynamic environments. Moreover, we had access to the CARIM system[1], a virtual training system developed for the Royal Netherlands Navy to train the tasks of an Officer of the Watch, and to subjects that were expert on the tasks of an Officer of the Watch.

An Officer of the Watch works aboard a navy frigate and is the person in charge when there is an incident on the ship. From the Technical Center of the ship (see Figure 4.1) he collects information, makes an assessment of the situation, develops plans to solve the incident, instructs other people, monitors the situation, and adjusts his plans if

[1] The CARIM system has been developed by TNO and VSTEP

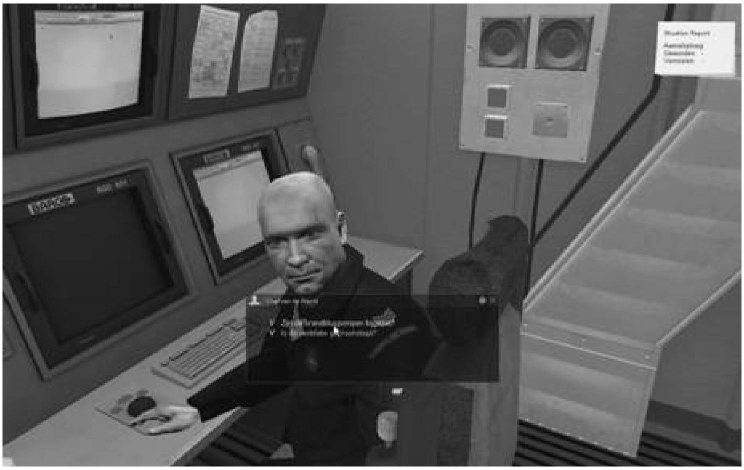


Figure 4.2: Interaction with virtual agents in CARIM.

necessary. The Officer of the Watch communicates with several other officers, of which the most important are the Chief of the Watch, the Leader Confinement Team, and the Leader Attack Team. In a typical incident scenario, the Officer of the Watch and the Chief of the Watch remain in the Technical Center, the Leader Attack Team is situated close to the location of the incident, and the Leader Confinement Team moves between both locations.

Subjects. The subjects, 15 in total, were instructors of the Royal Netherlands Navy. They all had expert knowledge about the task domain and experience in teaching.

Materials. The CARIM system (Cognitive Agents for Realistic Incident Management) provides virtual training for an Officer of the Watch. A single trainee plays the role of Officer of the Watch and can freely navigate through the Technical Center of the ship in the simulation. All equipment that the Officer of the Watch normally uses is simulated and available to the trainee, e.g., a damage panel containing a map of the entire ship, information panels and communication equipment. Agents communicate to the trainee through pre-recorded speech expressions, and a trainee can communicate to agents by selecting speech acts from a menu (see Figure 4.2). These menus are agent-specific and context dependent, and may change over the course of a training session. One training session takes about half an hour to complete.

The course of a training session in the CARIM system is guided by a scenario script. The script defines for each possible situation what should happen, which is either an event in the environment or an action of an agent. The trainee has certain freedom to act the way he wants, but if he deviates from the storyline in the scenario, the simulation redirects the trainee back to the intended scenario. For instance, if it is necessary that the trainee contacts the Leader Attack Team, the Chief of the Watch will repeat an advice to contact the Leader Attack Team until the trainee does so. In a more recent version of the CARIM system the behavior of agents is not scripted, but generated online by intelligent agents (Heuvelink et al., 2009). In this newer version, the trainee has more freedom in choosing his actions and the courses of different training scenarios are more diverse. Moreover, the intelligent agents can be reused in different scenarios. However, this new version of the CARIM system was not available yet at the time of the study. More details on the CARIM system can be found in two papers on the topic (Van den Bosch et al., 2009; Heuvelink et al., 2009).

Following our explainable agent approach presented in the previous chapter, we constructed goal hierarchies of three of the players in the CARIM scenario, the Chief of the Watch, Leader Confinement Team, and Leader Attack Team, based on interviews with experts and task descriptions provided by the Navy, and implemented them in 2APL (Dastani, 2008). Figure 4.3 shows a part of the goal hierarchy of the Leader Attack Team agent.

Ideally, the implemented agents would generate the behavior of the virtual agents online, but this was not possible because only a scripted version of the CARIM system was available. Therefore, instead, we simulated the events in the environment (without a visualization) and ensured that the agents we implemented would generate the same behavior as the scripted players in the CARIM system. Though the observable behavior of our explainable agents and the scripted agents was equal, the explainable agents made reasoning steps in order to generate behavior and the scripted agents did not. We used these actions and reasoning steps, logged in the agents' behavior logs, to generate explanations. More particularly, we applied the explanation algorithms B+1, B+2, G+1, G+2 and Gnext' to twelve actions, four actions of each of the three CARIM agents, which resulted in four or five explanations per action. In Figure 4.3, for instance, the action *Contact the OW* and its possible explanations are marked in gray.

We developed a paper-based questionnaire based on these twelve actions. In the first part of the questionnaire, the twelve actions were listed without explanations, and in the second part they were offered again with explanations. A question in the second part of the questionnaire was for instance which of the following explanations the subject considered most useful.

The Leader Attack Team contacted the Officer of the Watch because...
- there was a fire alarm
- he wants to initiate the fire attack
- he arrived at the location of the incident
- he wants to develop an attack plan
- he wants to report the situation

The five explanations in this example are of the types B+2, G+2, B+1, G+1 and Gnext', respectively. We translated the programming code explanations to sentences in natural language by hand.

In this study we used an older version of the explanation module in which the behavior log was kept in the agent's belief base (Harbers et al., 2009e). In the newer version,

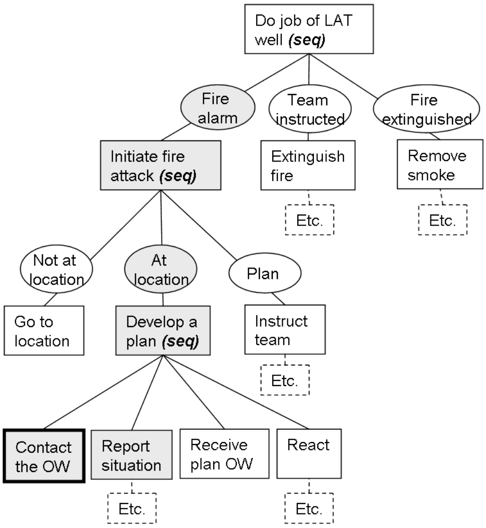


Figure 4.3: Part of the goal hierarchy of the Leader Attack Team.

logs are made in a separate explanation module implemented as an environment. Though logs are made in different locations in the old and the new version, the same information is stored. Therefore, the use of the older version was of no influence on the explanations that were generated. Moreover, the resulting explanations were exactly the same as when the explainable agents would have generated the behavior of the virtual CARIM players online.

Procedure. None of the subjects had used the CARIM system before. So to get acquainted to the system, they first received some time to practice with navigating their avatar, communicating with other agents, and marking an incident on a damage panel. Then all subjects played a training scenario with the CARIM system, and after that, filled in the questionnaire. In the first part, the open choice questions, the subjects had to explain twelve agent actions that occurred in the scenario. They were instructed to provide explanations from the perspective of the agent that executed the action, and to keep in mind that the explanations should help trainees to achieve a proper understanding of the situation. In the second part, the multiple choice questions, the same list of twelve actions with four or five possible explanations were presented to the subjects, and they were asked to select the explanation they considered most useful. Again, the subjects were instructed to keep in mind that the explanations should increase trainees' understanding of the situation. The subjects were deliberately asked to give their own explanations first to ensure that their responses were not influenced by the agents' explanations.

We conducted two experimentation sessions, with 8 subjects in the first and 7 subjects in the second session. In the first session, the subject could only choose between four possible explanation types (B+1, B+2, G+1 and B+2) in the multiple choice questions. The results of the first session showed that many of the subjects' own explanations contained explanations with enabled goals and beliefs about others' goals. Therefore, in the second experimentation session, we added explanations of type Gnext' to the multiple choice questions, and added questions with explanations containing a belief about others' goals. We will get back to these findings and adaptations in the results section.

4.1.2 Results

Though some of the subjects experienced difficulties with the navigation of their avatar at the beginning, all were able to solve the incident in the training scenario. In general, they were positive about the CARIM system as a training tool. We will first present the results of the open questions, followed by the results of the multiple choice questions. For the open questions, we only present the data of the subjects in the first experimentation session.

Open questions: explanation length. In the first part of the questionnaire, the subjects (n=8) were asked to provide explanations for twelve of the virtual agents' actions. For some actions, some of the subjects found it hard to come up with a useful explanation, and therefore we only obtained 88 instead of 96 explanations (8 subjects x 12 actions = 96). The first categorization of these explanations is according to their length. We defined explanation length by the number of elements, where an element is a goal, a belief, a fact, etc. Table 4.1 shows the frequencies of the number of elements in the subjects' explanations. The results show that most explanations contained only 1 element (70%). All others contained 2 elements (30%). No explanations with more than 2 elements were given.

Length	No. of explanations
1 element	62
2 elements	26
>2 elements	0

Table 4.1: Frequencies of the number of elements in the provided explanations (n=8).

Open questions: mental concept. A second way to categorize the explanations is according to mental concept type. More specifically, explanation elements can be categorized according to mental concept. Our aim was to examine whether the subjects' explanations are compatible with the intentional stance. Therefore, we tried to map the provided explanation elements to intentional concepts such as beliefs, goals, and intentions.

An examination of the provided explanations resulted into five mental concept types: (1) the condition for executing an action, (2) background information concerning an action, (3) others' goals that become achievable after executing an action, (4) the goal to be achieved with an action, and (5) the goal that becomes achievable after executing an action. A *condition* for an action was for example 'I went to the location of the incident because I heard the alarm message'. An example of *background information* is 'the Officer of the Watch and the Leader Attack Team communicate by a headphone'. An explanation referring to *another's goal* is, for instance, 'if I make the location voltage free, my colleague can safely use water in the room'. An explanation containing a *goal to be achieved* is, e.g., 'I put water on the fire to extinguish it'. Finally, an example of an explanation with an *enabled goal* is, e.g., 'I prepared fire hoses to extinguish the fire'.

Explanations with a condition, background information and another's goal can be considered as beliefs, and explanations with a goal to be achieved and an enabled goal can be categorized as goals. We do not claim that our classification is the only one possible. These results should rather be seen as an explorative examination of whether the provided explanations are compatible with the intentional stance. Table 4.2 shows the number of provided elements per mental concept type. Some of the explanations in Table 4.2 classified as enabled goals could also be classified as goals. For instance, the explanation 'the Leader Confinement Team goes to the TC to report to the Officer of the Watch' can be classified in two ways. Namely, the explaining element 'to report to the Officer of the Watch' can be seen as a goal of which going to the TC is a subgoal, but also as an enabled goal that can be achieved after the Leader Confinement Team arrived in the TC. In the first interpretation the explanation would be classified as a goal, and in the second as an enabled goal. In case of such ambiguity, we have chosen for the second interpretation, and classified the explanation as an enabled goal.

Mental concept	**No. of elements**
Condition belief	12
Background belief	14
Another's goal belief	26
Achieve goal	15
Enabled goal	47

Table 4.2: Number of explanations per mental concept type (n=8).

Multiple choice: abstraction level and mental concept. The second part of the questionnaire contained multiple choice questions. In the first experimentation session, the subjects had four choices, as described in the method section. However, the results in Table 4.2 show that many of the subjects' own explanations also contained enabled goals and beliefs about others' goals. Therefore, we decided to add these explanation types to part II of the questionnaire in the second session where possible. For seven actions (action 3,4,6,9,10,11,12) we initially offered the same four choices in both experimentation sessions. Table 4.3 shows which explanation types and abstraction levels were preferred for these actions, and whether at least 75% or 50% of the subjects (n=15) agreed on that. Thus, the italic numbers in the table are action numbers and not frequencies. The results show a divided picture. Out of seven actions, only for two actions (9 and 10) more than 75% of the subjects agreed on the preferred explanation type and abstraction level, which is reflected in a rather low kappa coefficient of 0.33. For action 4, no preference on which at least 50% of the sujects agreed was found.

Mental concept	**Abstraction**	**>75%**	**>50%**
Condition belief	Detailed	*10*	*3*
Condition belief	Abstract	-	-
Achieve goal	Detailed	*9*	*11,12*
Achieve goal	Abstract	-	*6*

Table 4.3: Preferred explanation types for actions 3,4,6,9,10,11,12 (n=15).

Multiple choice: abstraction level and mental concept with enabled goals. For five of the twelve actions (action 1,2,5,7,8) it was possible to derive an enabled goal explanation from the agents' goal hierarchies with the Gnext' algorithm. Note that during this study we had not yet developed the Gnext algorithm with which it is always possible to derive an explanation. In Figure 4.3, for instance, an explanation in terms of an enabled goal for the action 'The Leader Attack Team contacted the Officer of the Watch' is that 'The Leader Attack Team wants to report the situation to the Officer of the Watch'. For these five actions we added a fifth possible explanation to the answers of the multiple choice questions. Table 4.4 shows the results. The general agreement among the subjects expressed in a multi-rater kappa coefficient (Randolph, 2008) was 0.55. The results show that for some actions (action 2 and 5) a large majority of the subjects preferred an explanation in terms of an enabled goal. However, explanations in terms of enabled goals were not always preferred. Subjects even agreed for more than 75% that action 8 could best be explained in terms of a detailed condition belief.

Type	**Abstraction**	**>75%**	**>50%**
Condition belief	Detailed	*8*	-
Condition belief	Abstract	-	-
Achieve goal	Detailed	-	*1*
Achieve goal	Abstract	-	*7*
Enabled goal	Detailed	*2,5*	-

Table 4.4: Preferred explanation types for actions 1,2,5,7,8 (n=7).

Multiple choice: beliefs about others' goals. The results of the first session showed that, besides enabled goals, the subjects sometimes explained actions by beliefs about others' goals. Explanations in terms of another's goal are not derivable from the agents' own goal hierarchies, but it is possible to generate such explanations based on goal hierarchies of other agents. For each action, we selected a goal from another agent's goal hierarchy for which that action was relevant. We asked the subject's preferences about these explanations in separate questions, for instance, in the following multiple choice question.

The Leader Attack Team contacted the Officer of the Watch because...
- *< answer given in part a of the multiple choice question >*
- *the Officer of the Watch knows he is there*

Thus, after indicating their preference of four or five possible explanations, subjects were asked to compare their first answer to another (fifth or sixth) option involving an explanations in terms of another's goal. We used a separate question for this type of explanations because we did not use a formal explanation algorithm to generate them. The results of the follow up question are presented in Table 4.5. A kappa of 0.43 for the overall agreement among the subjects was found. The results show that for 9 out of 12 actions, the subjects preferred explanations in terms of another's goal over their first choice. For six of the actions (4,5,9,10,11,12) more than 75% of the subjects preferred an explanation in terms of another's goal and only for one action (1) more than 75% of the subjects agreed on a preference for an explanation not based on another's goal.

Type	**>75%**	**>50%**
First choice	*1*	*7,8*
Another's goal belief	*4,5,9,10,11,12*	*2,3,6*

Table 4.5: Preferences for explanations with another's goal belief (n=7).

4.1.3 Discussion

The first objective of the study was to examine whether preferred explanations are compatible with an intentional perspective. In part I of the questionnaire, the subjects were asked to provide explanations for actions freely. We were able to classify the elements in the subjects' own explanations in five mental concept types, which were all either belief-based or goal-based (Table 4.2). Though the explanations might be classifiable in other ways, possibly in non-intentional explanation types, it was possible to classify them consistently with an intentional perspective. We may thus conclude that the preferred explanations are compatible with the intentional stance, and that a BDI-based approach seems appropriate for developing self-explaining agents.

We formulated three propositions about the nature of preferred explanations. Concerning explanation length, our proposition was that short explanations would be preferred over long ones. In part I of the questionnaire, the subjects were asked to provide their own explanations, whereby no restrictions on explanation length were given. Table 4.1 showed that their explanations in most cases only contained one element, and never more than two elements. These results thus support our proposition. Consequently, self-explaining agents should make a selection of elements which they provide in an explanation.

Concerning preferred abstraction level, our proposition was that detailed explanations would be preferred over abstract ones. In this study, detailed and abstract explanations consisted of mental concepts low (just above action level) and higher in an agent's goal hierarchy, respectively. We cannot give a general conclusion concerning preferred abstraction level because the data only give information about the preferred abstraction level of two types of mental concepts, condition beliefs and achieve goals. For belief-based explanations, the results clearly show that detailed explanations are preferred over abstract ones (Table 4.4 and 4.3). For goal-based explanations, the results in Table 4.4 and 4.3 also show that detailed explanations are preferred over abstract ones, but not as strongly as belief-based explanations.

A possible explanation for the finding that the subjects hardly preferred abstract belief-based explanations is that condition beliefs are often directly related to events in the environment. Abstract beliefs take place earlier in time than detailed beliefs, for example, the fire alarm rings before the Leader Attack Team reaches the location of the incident, and it is plausible that more recent cues are preferred over older ones.

Our last proposition involved the mental concept type of preferred explanations. Our proposition was that most of the subjects' own explanations would contain condition beliefs and achieve goals. Though a part of the instructors' explanations in part I could be classified into one of these categories, the results in Table 4.2 show that three other mental concept types were also used, namely background beliefs, another's goal beliefs, and enabled goals. The results from part II of the questionnaire supported these findings. Namely, Table 4.4 and 4.5 show that the instructors sometimes selected other mental concept types than the ones we originally expected. Our proposition concerning mental concept type is thus only partly supported by the results.

4.2 Novices' preferred explanations

The first study corresponded to earlier findings that preferred explanations are compatible with the intentional stance, and that preferred explanations are relatively short. The results were more diverse regarding preferred abstraction level and mental concept type. In this second study, we want to test the proposition that preferred abstraction level and mental concept type depend on characteristics of the action to be explained. If this turns out to be the case, action characteristics can be used to develop guidelines for which explanation types should be given for which action types.

An important change with regard to the first study is that we used novices instead of instructors as subjects. First, to test if we can find differences between novices and instructors. And second, an advantage is that the subjects did not have to take the perspective of someone else when evaluating explanations, but could just indicate how useful the explanations were for them.

4.2.1 Method

Domain and training task. It was not possible to use the CARIM system again for the second study because there were not enough suitable novel students available at the Royal Netherlands Navy. Using subjects that were not training for Officer of the Watch was also not an option, as it requires a great deal of domain knowledge to train with the CARIM system. Therefore, we used a similar, but somewhat better known domain than onboard firefighting instead, namely, civil firefighting. Most people are more familiar to this domain than onboard firefighting, and the jargon is more common to most people than the terminology used in the navy.

In the scenario used in this study, a leading firefighter is confronted with a fire alarm. He goes to the location of the incident with his team, and once there, he has to assess the situation, develop a plan, instruct his team members, monitor plan execution, and initiate changes if necessary.

Subjects. The study was performed with 20 subjects (12 male, 8 female). All had higher education, the average age was 31 (range 24-58) and 17 of them were native Dutch speakers. The subjects were unfamiliar with the domain of firefighting and the tasks of a leading firefighter, they can thus be considered as novices.

Materials. We analyzed the task of a leading fire-fighter by reading protocols and consulting experts. Based on that, we constructed the goal hierarchy shown in Figure 4.4, and implemented the agent in 2APL. For the experiment, we used the actions in the numbered gray boxes in Figure 4.4. Action 4, 5, 6, 7 and 8 have a relation of type *all* to their parent goal, action 10, 11, 12, 13 have a relation of type *one*, action 14, 15 and 16 have a relation of type *if*, and action 1, 2, 3 and 9 have a relation of type *seq*. We applied four explanation algorithms, B+1, G+1, G+2 and Gnext, to the 16 actions to generate four explanations for each action. We composed a questionnaire in English containing the 16 actions with each four explanations. For each subject, the questions were placed in a (different) random order to correct for order effects. The order of the four explanations per action was also randomized. The question related to action 1 was for example:

Why did the leading firefighter collect his equipment?
- To prepare and go to the fire
- To lead the fire-fighting team
- Because he had not yet collected his equipment
- To get into the fire engine after that

The explanations are, respectively, of the types G+1, G+2, B+1 and Gnext.

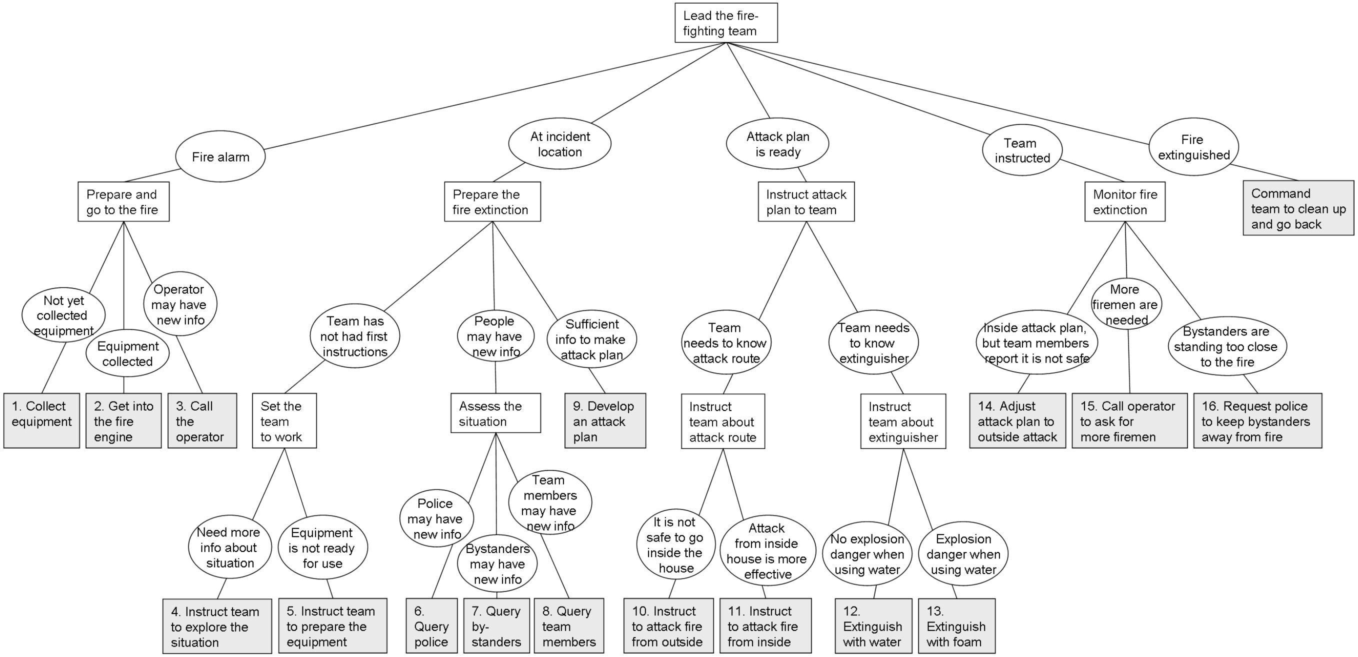

Lead the fire-fighting team
Fire alarm
At incident location
Attack plan is ready
Team instructed
Fire extinguished
Prepare and go to the fire
Prepare the fire extinction
Instruct attack plan to team
Monitor fire extinction
Command team to clean up and go back
Not yet collected equipment
Equipment collected
Operator may have new info
Team has not had first instructions
People may have new info
Sufficient info to make attack plan
Team needs to know attack route
Team needs to know extinguisher
Inside attack plan, but team members report it is not safe
More firemen are needed
Bystanders are standing too close to the fire
1. Collect equipment
2. Get into the fire engine
3. Call the operator
Set the team to work
Assess the situation
9. Develop an attack plan
Instruct team about attack route
Instruct team about extinguisher
14. Adjust attack plan to outside attack
15. Call operator to ask for more firemen
16. Request police to keep bystanders away from fire
Need more info about situation
Equipment is not ready for use
Police may have new info
Bystanders may have new info
Team members may have new info
It is not safe to go inside the house
Attack from inside house is more effective
No explosion danger when using water
Explosion danger when using water
4. Instruct team to explore the situation
5. Instruct team to prepare the equipment
6. Query police
7. Query by-standers
8. Query team members
10. Instruct to attack fire from outside
11. Instruct to attack fire from inside
12. Extinguish with water
13. Extinguish with foam

Figure 4.4: Goal hierarchy of a leading firefighter agent.

Procedure. Instead of performing an actual virtual training, the subjects received a description of the tasks of a leading firefighter. The description gave them a context to understand the leading firefighter's actions listed in the questionnaire. In the paper-based questionnaire, the subjects had to indicate for 16 actions which of four explanations they considered most useful for learning the task of a leading firefighter. Usefulness was explained as: 'which explanation helps you best in understanding the leading firefighter's motives for performing those actions?'. The subjects could only pick one explanation, and they were told that none of the four explanations was wrong.

4.2.2 Results

The subjects reported no problems with understanding the leading firefighter's actions, and all subjects were able to indicate a preferred explanation for all actions. Table 4.6 shows for each of the firefighter's actions ($1-16$): the action's relation to its parent task (all, seq, if, one), and per explanation algorithm (G+1, G+2 B+1, and Gnext), the number of subjects who preferred that explanation for the action.

To remind, algorithm G+1 explains an action by the parent goal of the action, algorithm G+2 explains an action by the parent goal of the parent goal of the action, algorithm B+1 explains an action by the enabling belief(s) of that action, and algorithm Gnext explains an action by the first action or goal that must follow the action.

If preferred explanation type is independent of action type, similar preferences for explanation types are expected over different action types. The results show different preferences, however, and we calculated whether these differences are significant. As the frequencies in Table 4.6 are dependent observations (the subjects make more than one observation, i.e., they select an explanation for several actions), the statistical analysis of the results is done per action. We performed a Chi-square goodness-of-fit to the results of every action, with an expected frequency of 5 per explanation algorithm. The tests on action 2 (χ^2 = 2.8, p = 0.42) and action 9 (χ^2 = 7.6, p = 0.055) indicate that the results are not significant, but the test on action 8 (χ^2 = 8.8, p = 0.032) shows significance at a level of 5% and the tests on all other actions are significant at a level of 1%.

We calculated kappa to express the agreement among subjects and found K = 0.23 (p = 0.15), indicating fair agreement. This result can partly be explained by the fact that algorithm B+1 was overall often chosen, which gives a high P(E) in the calculation of kappa. Some statisticians suggest using free-marginal kappa when raters are not forced to assign a certain number of cases to each category. A free-marginal kappa calculation gives K = 0.44, and when action 2 and 9 (not significant) are omitted even K = 0.50, indicating moderate agreement.

The pie charts in Figure 4.5 summarize the data in Table 4.6. Each pie chart shows for one action relation (all, one, if, seq) the percentages of preferred explanation types. The charts clearly show that for actions with a *one* or an *if* relation to their parent, explanations generated by algorithm B+1 are preferred, i.e., explanations containing an enabling belief. For actions with relations *all* and *seq*, also other explanation types are preferred. For all action relations it holds that explanations generated by algorithm Gnext were rarely chosen.

Actions		Explanation types			
Relation	**No.**	**G+1**	**G+2**	**B+1**	**Gnext**
seq	1	13	1	3	3
	2	7	6	5	2
	3	4	1	14	1
	9	10	3	5	2
all	4	1	2	13	4
	5	1	12	6	1
	6	10	0	9	1
	7	7	0	12	1
	8	6	2	10	2
one	10	1	1	18	0
	11	1	1	18	0
	12	2	3	15	0
	13	2	1	17	0
if	14	0	1	19	0
	15	0	0	20	0
	16	1	0	19	0
Total		**66**	**34**	**203**	**17**

Table 4.6: Frequencies of preferred explanation types per action.

4.2.3 Discussion

Our proposition that different actions yield different preferred explanation types is supported by the results. More specifically, we expected that the preferred explanation type of an action depends on the relation between the action and its parent goal. The results indeed show that for action relations *one* and *if*, subjects preferred explanations generated by algorithm B+1, and for action relations *all* and *seq*, subjects preferred a mix of different explanation types.

There are several factors that could explain the variation among preferred explanation types for the actions with *seq* and *all* relations. For some of the individual actions there is no general agreement on preferred explanation type. This holds for action 2 and 9, on which the preference results are not significant, and to a lesser extent for action 6 and 8, where not more than 50% of subjects preferred the same explanation. Possible explanations for this disagreement among subjects are that none of the offered explanations are considered useful and the results do not give much information (this would require a whole other type of explanation), or that a combination of the provided explanations would be considered useful. The low agreeability could also be caused by methodological errors. For instance, for action 2, ‘Get into the fire engine’, the word ‘fire engine’ may have caused problems for mostly native Dutch speaking subjects, with English as their second language. And it could be argued that action 9, ‘Develop an attack plan’, was misplaced in the goal hierarchy. Instead of being a subaction of the goal ‘Prepare

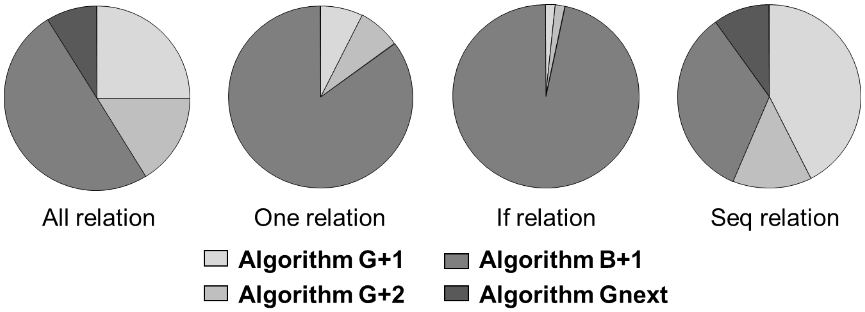


Figure 4.5: Percentages of preferred explanation type per action relation.

the fire extinction', it could also be a subaction of the main goal, 'Lead the firefighting team'. In that case, other explanations would have been generated. Finally, it could be that preferred explanation type does not (only) depend on the action relation, and that other factors are needed to account for preferred explanation type.

Concerning explanation algorithms, explanations generated by algorithm Gnext are rarely preferred and not discussed further here. Of the other three algorithms, G+1 and G+2 generate explanations containing a goal to be achieved, and B+1 generates explanations containing an enabling belief. The result of application of algorithm G+1 and G+2 strongly depends on the design of an agent's goal hierarchy. Actions can sometimes be placed at different levels in the goal hierarchy, as seen with action 9, and it is up to the designer how many subgoals are included in the hierarchy. Thus, the explanations generated by algorithm G+1 and G+2 are more similar to each other than explanations generated by algorithm B+1. When we collapse G+1 and G+2 into one category, of the four actions with a *seq* relation, B+1 explanations are preferred for action 3, but for the others at least 65% of the subjects prefers an explanation with a goal to be achieved (either one or two levels above the action). This gives a stronger indication that, in general, goal-based explanations (G+1 or G+2) are preferred for actions with a *seq* relation.

Though there are no clearly preferred explanation types for actions with relation *all* and *seq*, there is a clear difference between actions with relation *if* and *one* on the one hand, and *seq* and *all* on the other hand. A difference between the two groups is that *all* and *seq* actions have to be executed in order to achieve their parent goal, whereas the execution of *if* and *one* actions is not always necessary for the achievement of their parent goal. For *if* and *one* actions holds that their execution strongly depends on the conditions in the environment. For example, dependent on the risk of explosion, the fire is either extinguished with water (action 12) or with foam (action 13). And only if more firemen are needed, the leading firefighter has to call the operator (action 15). Conditions in the environment are reflected in the agent's beliefs and it is therefore not surprising that explanations containing a belief are preferred. Actions like getting into the fire engine (action 2) and instructing the team to prepare the equipment (action 5), in contrast, always need to be performed when there is a fire. Actions with a relation *all* and *seq* are often part of a procedure or embody a rule. Thus, when the performance of an action strongly depends on the state of the environment, a belief-based explanation is preferred. When that is not the case, other explanation types are also preferred.

4.3 Experts' preferred explanations

Like in the second study, the proposition tested in this study is that different action types require different types of explanation. We will test for actions with *all*, *one* and *seq* relations how useful and natural explanations generated by the algorithms B+1, G+1 and Gnext are considered. At the time we started to prepare this study, we did not distinguish actions with an *if* relation yet. The difference between this and the second study is that we use experts instead of novices as subjects. Furthermore, whereas in the previous two studies we only asked subjects to indicate which explanations they considered *most* useful, in this study we also ask them to indicate *how* useful. Besides the usefulness of explanations, we also considered the experienced naturalness of explanations. The exact description of naturalness follows in the procedure. Finally, in the first two studies we did not consider explanations containing more than one mental concept because the subjects provide short explanations themselves (see Table 4.2). In this study, however, the subjects do get the possibility to select different mental concepts to explain one action, e.g., a combination of a goal and a belief.

4.3.1 Method

Domain and training task. The domain of this study is cooking and the task being trained is baking pancakes. In contrast to the training tasks in the first two studies, this task is not very complex and dynamic, and it does not require interaction with other players. We chose this relatively simple task because most people are familiar with it. Not being dependent on scarcely available expertise allowed us to have a larger number of subjects.

Subjects. The study was performed with 30 subjects (18 male, 12 female) with an average age of 32 (sd=9). The subjects rated their own cooking skills on a 5-point Likert scale with an average of 3.6. Their education levels varied between Bachelor, Master and PhD.

Materials. Figure 4.6 shows the goal hierarchy of the cooking agent. The agent was implemented in the BDI language GOAL (Hindriks, 2009), and executed to generate explanations automatically for all of the agent's actions. The agent program was constructed such that it included actions of types *all*, *one* and *seq*. Action 1, 6, and 10 are of type *one* (mutually exclusive actions), action 2, 3, 4, and 5 are of type *all* (actions that all need to be executed), and action 7, 8, 9 and 11 are of type *seq* (actions that all need to be executed in a particular order). We let the explainable agent use the explanation algorithms B+1, G+1 and Gnext to generate three explanations for each action.

Post hoc analysis excluded action 11 from the statistical result analysis as this action was misplaced in the tree (see Results section).

Procedure. To investigate the effects of algorithm and action type on the perceived usefulness and naturalness of explanations, we used a 11x3 between-subjects design (11 actions, 3 algorithms) with usefulness and naturalness as dependent variables. Subjects were randomly assigned to the different explanation conditions with exactly 10 subjects per condition. All subjects scored all actions for a particular explanation condition, resulting in 10 measurements per action per condition.

The procedure of the experiment was as follows. Subjects were told to read the instructions (stating that the study was about developing smart agents for virtual training purposes), after which they received the first feedback form. Subjects were asked to write down their own explanations for the 11 actions listed on the form (see also the gray boxes in Figure 4.6), as if they were the cook explaining to a student how to bake pancakes. This feedback was aimed at extracting the 'ideal' explanations according to the subjects. When finished, subjects received a second form. This form asked for 5-point Likert feedback on the *naturalness* of each action's automatically generated explanation (1=not natural, 5=very natural). Subjects were asked to take the role of observer when judging the naturalness of the explanation. Naturalness was explained as follows: "With a natural explanation we mean an explanation that sounds normal and is understandable, an explanation that you or other people could give". When finished, a similar form was presented for 5-point Likert feedback on the *usefulness* of the explanations. Subjects were asked to imagine they were the student learning to cook when judging the usefulness. Useful was explained as follows: "Indicate how useful the explanations would be for you in learning how to make pancakes". Finally, subjects were presented with the goal tree (the graphical representation of the goal hierarchy as shown in Figure 4.6). We asked users to indicate all elements in the tree they deemed useful for giving an explanation of each of the 11 actions, by putting the action number next to the element. Subjects were asked to imagine they were the instructor cook when selecting elements. We call this tree-based feedback. The purpose of this measure was to extract information about what could be a good and feasible version of an explanation algorithm, given our way of automatically generating behavior logs.

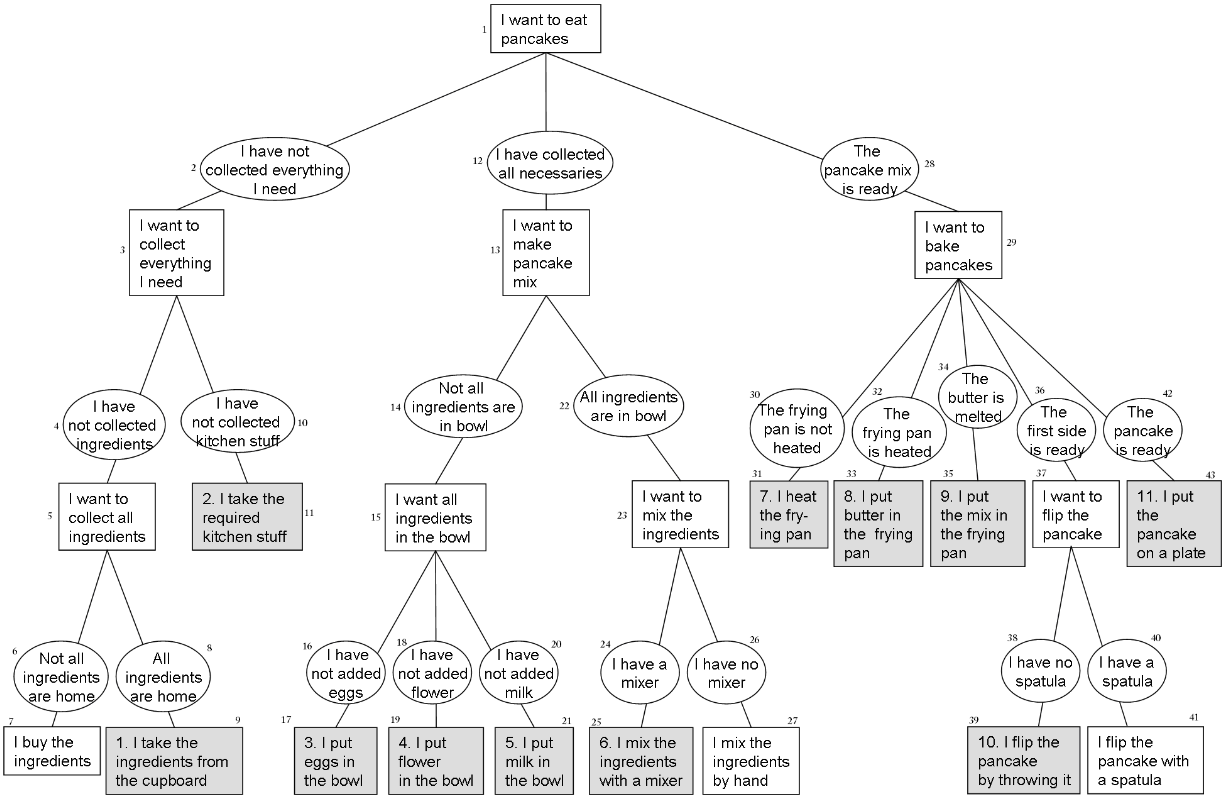

1 I want to eat pancakes
2 I have not collected everything I need
12 I have collected all necessaries
28 The pancake mix is ready
3 I want to collect everything I need
13 I want to make pancake mix
29 I want to bake pancakes
4 I have not collected ingredients
10 I have not collected kitchen stuff
14 Not all ingredients are in bowl
22 All ingredients are in bowl
30 The frying pan is not heated
32 The frying pan is heated
34 The butter is melted
36 The first side is ready
42 The pancake is ready
5 I want to collect all ingredients
11 2. I take the required kitchen stuff
15 I want all ingredients in the bowl
23 I want to mix the ingredients
31 7. I heat the fry-ing pan
33 8. I put butter in the frying pan
35 9. I put the mix in the frying pan
37 I want to flip the pancake
43 11. I put the pancake on a plate
6 Not all ingredients are home
8 All ingredients are home
16 I have not added eggs
18 I have not added flower
20 I have not added milk
24 I have a mixer
26 I have no mixer
38 I have no spatula
40 I have a spatula
7 I buy the ingredients
9 1. I take the ingredients from the cupboard
17 3. I put eggs in the bowl
19 4. I put flower in the bowl
21 5. I put milk in the bowl
25 6. I mix the ingredients with a mixer
27 I mix the ingredients by hand
39 10. I flip the pancake by throwing it
41 I flip the pancake with a spatula

Figure 4.6: Goal hierarchy of the cooking agent. The gray boxes denote the 11 actions used in the experiment.

Actions		Explanation types		
Relation	No.	G+1	B+1	Gnext
one	1	3.8	2.4	3.6
	6	3.4	2.3	2.0
	10	3.0	3.4	1.7
all	2	3.7	3.2	2.9
	3	2.8	2.9	3.6
	4	2.8	3.0	3.7
	5	2.9	3.0	3.7
seq	7	3.1	3.6	2.2
	8	2.2	2.9	2.4
	9	3.2	2.8	1.5
	11	1.2	4.1	4.1

Table 4.7: Naturalness of actions per condition.

4.3.2 Results

To test our proposition that different action types require different types of explanations, we performed a 3x10 2-way MANOVA with algorithm (3 conditions) and action (10 conditions) as independent variables, and usefulness and naturalness as dependent variables. Analysis showed a main effect of algorithm type ($F(4, 538) = 3.973, p < 0.01$), a main effect of action ($F(18, 538) = 1.917, p < 0.05$), and an interaction effect between action and algorithm ($F(36, 538) = 2.638, p < 0.001$). Post hoc testing (Tukey) for the influence of action alone on naturalness and usefulness revealed no significant differences. This indicates that the actions were explained equally well according to the subjects, meaning that no action had more natural and useful explanations than another. Post hoc tests for the effects of algorithm type revealed a significant effect on perceived usefulness. Inspection of the data showed that algorithm G+1 (parent goal as explanation) was rated significantly more useful ($p < 0.01$) than the other two algorithms (Mean(G+1)=3.1, Mean(B+1)=2.5, Mean(Gnext)=2.5). This indicates that there is a significant influence of algorithm type on the perceived usefulness of the explanation, and that explaining an action with its parent goal (Algorithm G+1) is the best default method. However, the interaction effect, indicating that different actions need different explanations (supporting our main proposition), is more important, as we will see next.

Tables 4.7 and 4.8 give an overview of the average naturalness and usefulness of the actions per algorithm type, respectively. Figure 4.7 displays a frequency table, showing for each action (listed vertically) which element(s) in the tree (listed horizontally) the subjects chose to explain that action. The elements and their corresponding numbers are shown in Figure 4.6.

As can be seen, actions 1, 2, 6, and 9 have higher scores on both measures when the parent goal is given as explanation (algorithm G+1), while actions actions 3, 4, and 5 have higher scores on both measures when the next action or goal is given as explanation ("I

Actions		Explanation types		
Relation	**No.**	**G+1**	**B+1**	**Gnext**
one	1	3.8	1.7	2.9
	6	3.3	2.1	2.0
	10	2.0	3.3	1.0
all	2	3.8	2.4	2.8
	3	3.0	2.5	3.5
	4	3.0	2.5	3.7
	5	3.1	2.5	3.7
seq	7	2.5	2.7	1.9
	8	2.5	2.5	2.4
	9	3.8	3.1	1.0
	11	1.3	4.3	3.7

Table 4.8: Usefulness of actions per condition.

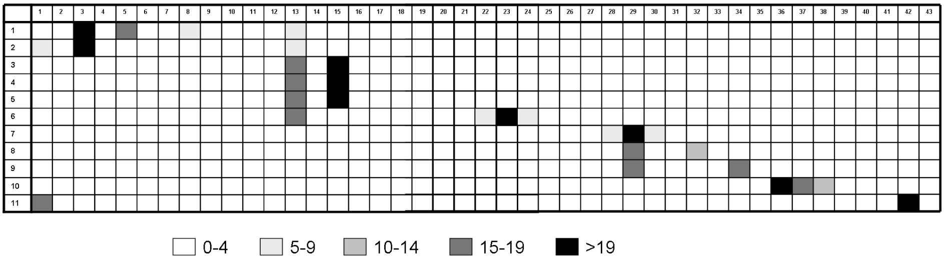


Figure 4.7: Frequencies of the tree elements that were selected as explanations for different actions. Actions number from 1 to 11, elements number from 1 to 43 and refer to numbers in Figure 4.6.

want to mix the ingredients", algorithm Gnext), and actions 7 and 10 have higher scores when the enabling condition (belief) is given as explanation (algorithm B+1). Action 8 does not have high scores on either of the algorithms. Action 11 has a high score when explained by algorithm Gnext, but this is a side effect of two factors. First, it can be argued that action 11 should have been placed under "I want to eat pancakes". Second, Gnext defaults to the top level goal when no next steps are available in the sequence, which in our case happened to be the most logical option for explanation. We therefore decided to exclude action 11 from our analysis.

Actions 2, 3, 4, and 5 are actions of the type *all*; they are all needed (in arbitrary order) to achieve the parent goal. For 3, 4, and 5, the parent goal is not very helpful for understanding the underlying actions, e.g., I put X in the bowl - because I want to put all ingredients in the bowl. As can be seen in Figure 4.7, subjects included in their own choice of elements the goal numbered 13 ("I want to make pancake mix"), indicating that subjects indeed need a more descriptive goal. Action 2 is well explained by its parent goal, as indicated by the naturalness and usefulness scores as well as the tree-based feedback.

Actions 1, 6, and 10 are actions of the type *one*. Action 1 and 6 score high on using the parent goal as explanation, but in addition to that they seem to require extra information for an adequate explanation. In Tables 4.7 and 4.8 we can see that for action 1 and 6 subjects preferred the goal two levels up in the hierarchy. Action 10 is well explained by algorithm B+1. This is reflected in the tree-based feedback, as for action 6 and 10 subjects use the enabling conditions for the action and for the parent goal. Action 6 seems to have a rather complex explanation structure using two goals and two conditions.

As indicated by the tree-based feedback, enabling conditions in combination with the parent goal are also used for action 7, 8, and 9; all three actions are actions in a sequence, type *seq*. However, action 8 and 9 use only the enabling condition for the action itself, while action 7 uses both the enabling condition for the action itself as well as the enabling condition for the action's parent goal. We will interpret these results in more detail in the discussion.

Finally, we have conducted several standard correlations between the subject demographics and usefulness and naturalness. We found four significant correlations. Two of the correlations were positive: the one between usefulness and naturalness ($p < 0.001, r = 0.491$), and the one between cooking skill and usefulness ($p < 0.001, r = 0.145$). The first correlation is as expected. It indicates that natural explanations are more useful and vice versa. The second is somewhat counterintuitive: more experienced cooks judge the explanations slightly more useful. This could be due to the fact that a better cook is better able to understand the explanation in the first place, but as the correlation coefficient is rather small, we do not pay further attention to this. Furthermore, we found two negative correlations: between action number and naturalness ($p < 0.001, r{-}0.200$), and between action number and usefulness ($p < 0.01, r = -0.178$). As actions were always scored from top to bottom, and this corresponds to the action number, this might indicate two different things: for the later actions it is more difficult to generate explanations automatically, or, subjects got tired of scoring explanations. As we cannot decide upon this, we do not pay further attention to it here.

4.3.3 Discussion

Our results indicate two things. First, the data support our proposition that different actions need different explanations. Second, we failed to find support for our proposition on how action types and explanation algorithms relate. This suggests that our notion is too simplistic. We expected that *all* actions would be explained best by the action's parent goal, that *seq* actions would be explained best by the next action/goal in the sequence, and *one* actions would be best explained by their enabling condition. However, looking at the tree-based feedback, most of the actions seem to need at least one additional element for explanation, in addition to their parent goal. The kind of additional information seems to depend on the action's role in the process and the action's type (*seq*, *all*, or *one*).

First, consider the actions of type *all*: action 2, 3, 4 and 5. According to the tree-based feedback, only action 2 is explained well by only one element, its parent goal. Action 3, 4 and 5 are well explained by the next action in the sequence according to Table 4.8, but as subjects indicated in their tree-based feedback, they choose for a combination of the parent goal and the parent's parent goal. We currently cannot explain this difference, but it does indicate that neither the enabling condition nor the parent goal are sufficiently descriptive in this particular case, i.e., on their own they do not comprise a useful explanation.

Second, consider actions 1, 6 and 10 which are of type *one*. The way this type of action is modeled in the tree is such that the parent goal presents a choice, while the enabling condition of the action's parent explains why the choice has to be made. From the experiment we learned that for this action type, the parent goal is not sufficiently descriptive to provide a satisfying explanation. Instead, both the enabling condition of the action and the enabling condition of the parent goal are needed.

Third, consider the actions 7, 8, 9, and 10 which are part of the same sequence (note that 7, 8 and 9 are of type *seq*, but 10 is of type *one*). According to the tree-based feedback (Figure 4.7), these actions should be explained by their parent goal and their enabling condition, contrary to our expectation that such actions would need the next action/goal in the sequence. In addition, actions 7 and 10 also need the enabling condition of their parent's goal in their explanations. A possible explanation for this difference is that action 8 and 9 are in the middle of a sequence. Their parent goal explains what is to be done, and the enabling condition explains where we are in the process. Action 10 does need its parent goal and its enabling condition because it is an action of type *one*. The enabling condition of its parent goal needs to be given because it is also, though implicitly, part of the sequence involving action 7 to 10. Action 7 can be explained in the same way. It is the first action of a next phase in the process (baking). The parent goal of action 7 is about that next phase, but it does not explain why we ended up in this phase. This is what the parent goals' enabling condition is about, therefore, action 7 needs again two enabling conditions (its own and that of its parent goal).

4.4 Discussion and explanation guidelines

We will first provide a discussion on the results of the three studies, focusing on their differences and similarities. We refer to the studies with instructors (Section 4.1), novices (Section 4.2) and experts (Section 4.3) by Study 1, 2, and 3, respectively. Then, based on the findings of these studies, we provide a set of guidelines for modeling explainable agents and explaining their behavior.

4.4.1 Discussion

From the literature we learned that people adopt the intentional **explanatory stance** when they explain (intentional) human behavior. In other words, human(-like) behavior is explained by mental concepts such as beliefs and goals. The results of Study 1 show that it is possible to categorize the subjects' explanations in beliefs and goals, i.e., they are *compatible* with the intentional stance (we do not claim that this is the only way to categorize these explanations). In Study 3, the subjects' explanations were not categorized systematically, but an examination of the explanations provides a similar picture. Thus, our results support the earlier findings showing that people explain human-like virtual player behavior by the underlying beliefs and goals.

The results also support earlier observations that preferred explanations are relatively short. We expressed **explanation length** by the number of elements in an explanation, where an element is a fact, a goal, etc. In Study 1, the subjects' explanations had an average length of 1.3 elements, and in Study 3 the subjects selected an average of 1.7 elements from the goal hierarchy. The lower average in Study 1 might be due to the fact that the subjects had to write down complete explanations, whereas in Study 3 they only had to mark numbers of elements. So as expected, people's explanations about virtual player behavior usually only contain one or two elements.

The results discussed so far support our proposition that explanations contain a selection of beliefs and goals. Therefore, it makes sense to examine people's preferred **abstraction level** and **mental concept type**.

In Study 1, except for explanations of type B+2, all explanation types (G+1, G+2, B+1) were considered most useful for at least one action by the majority of the subjects.

In Study 2, for actions of type *one* and *if*, explanations containing a belief (B+1) were clearly preferred, and for actions of type *all* and *seq*, also other explanation types (G+1, G+2, B+1) were preferred for at least one action by most of the subjects. The results of Study 2 are consistent with those in Study 1, in which only *all* and *seq* actions were examined.

In Study 3, unlike Study 2, for all action types, explanations of type G+1 were on average rated higher than those of type B+1. Like in Study 2, for action types *one* and *seq*, Gnext explanations received relatively low ratings, and for actions of type *all*, they were highly rated. The usefulness of type Gnext explanations is closely related to the goal hierarchy to which it is applied. This will be discussed in more detail in the next section. Interestingly, in the last part of Study 3, subjects often selected a combination of a belief and a goal as their preferred explanation.

A remarkable difference between Study 1 and 3 on the one hand, and Study 2 on the other hand is that goal-based explanations were generally stronger preferred in the former, and belief-based explanations in the latter. A possible reason is that the subjects in Study 2 (novices) were unfamiliar with the domain, whereas the subjects in Study 1 and 3 (instructors and experts) were familiar with the training task. Literature suggests that, on average, beliefs carry more idiosyncratic information and are harder to infer than goals (Malle, 1999). For subjects unfamiliar with a training task, belief-based explanations may provide more information that they are not able to derive from the context themselves than goal-based explanations. Experts may not realize that goal-based explanations are easier to infer for trainees. A second explanation is that experts, more than non-experts, focus on the general characteristics of a virtual character's behavior (Klein, 1998). The instructors in Study 1 ought to know which explanations are useful for trainees as they have, besides being expert on the training task, didactic knowledge.

To conclude, action type is sometimes, but not always, predictive for preferred explanation type. Of all studies, only Study 3 indicates to what extent explanations are preferred. The highest usefulness scores on action type *all*, *one* and *seq* are 3.4, 3.0 and 2.9, respectively. The scores are not low (all above the average of 2.5), but not very high either. In the experiments, we only provided subjects with explanations containing one element, but the results seem to indicate that both beliefs and goals carry important information. Thus, though the results of the three studies do not show which explanation type is preferred for each and every situation, they provide directions for modeling and explaining virtual player behavior. In the next subsection we present a set of guidelines for designing and explaining agent behavior.

4.4.2 Modeling and explanation guidelines

The design and explanation of agent behavior are closely related in our approach. First, the elements in an agent's behavior representation determine the content of its explanations. Namely, in our approach explanations consist of the beliefs and goals underlying an action. Though a virtual agent's beliefs and goals remain unknown for users that only see its behavior, they become explicit when the agent's behavior is explained.

Second, the way the elements in an agent's behavior representation are organized also determines the content of its explanations. Two behavior representations that contain same elements, but that are structured in a different way, may produce the same observable agent behavior, but generate different explanations about it. Figure 4.8, for instance, shows two possible positions of action E in a goal hierarchy. As both relations in this hierarchy are of the type *seq*, the position of action E does not affect the agent's observable behavior, but it may influence the way action E is explained. When action E is explained by using explanation algorithm G+1, it will be explained by either Goal A (left goal hierarchy in Figure 4.8) or Goal B (right goal hierarchy in Figure 4.8).

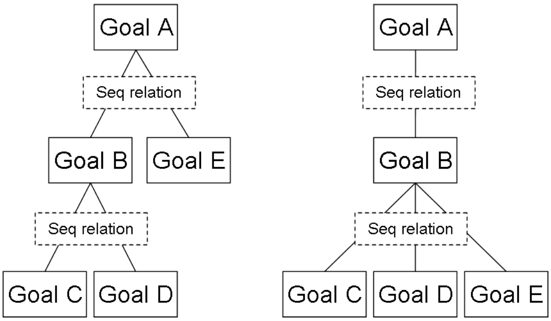


Figure 4.8: First example of two goal hierarchies that generate the same behavior, but different explanations.

A second example in which the organization of the behavior representation matters for the generation of explanations, but not for the generation of behavior is the following. Consider the two goal hierarchies in Figure 4.9. Goal B and C can be modeled as two neighboring goals or as goal and subgoal, e.g., when goal A, B, and C represent *Report* *to head officer*, *Go to the head officer* and *Report new information*, respectively. In the first case, achieving goal B enables the achievement of goal C, and in the latter, goal C is achieved by achieving B.

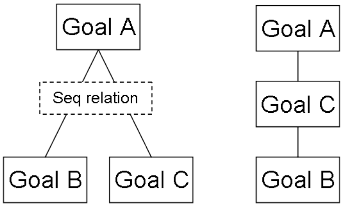


Figure 4.9: Second example of two goal hierarchies that generate the same behavior, but different explanations.

Of course, developing agents for virtual training always deserves ample attention. But as illustrated, when developing explainable agents it is particularly important to pay attention to the elements in a behavior representation and the way they are organized. Thus, though they may seem obvious, the following two modeling guidelines are crucial for developing useful explainable agents.

- **Guideline 1:** the goals and beliefs in a goal hierarchy should be meaningful.
- **Guideline 2:** the internal structure of a goal hierarchy should be meaningful.

At the end of the overall discussion, we concluded that both beliefs and goals carry important information for explanations. The results showed that beliefs directly above an action (B+1) were considered most useful for explaining that action. Regarding goal-based explanations, the studies are less conclusive; several goal-based explanation types were considered useful (G+1, G+2 and Gnext) for different actions. But all together, goal-based explanations of type G+1 were most often preferred and highest rated. Moreover, people tend to use explanation types B+1 and G+1 together. Therefore, we propose the following default explanation guideline. This guideline applies to all action types and is supported by the results of all three studies.

- **Guideline 3:** an action should be explained by its goal G+1 and belief(s) B+1.

In addition to this general guideline, we propose two more specific guidelines that take action type into account. The first guideline concerns the addition of a Gnext goal to the default explanation of G+1 and B+1. In contrast to G+1 and G+2 explanations, Gnext explanations do not contain goals from a particular level above the action. The level of the Gnext goal depends on the relations in the goal hierarchy. In Study 3 we found that Gnext explanations were considered useful for actions of type *all* whose parents had a *seq* relation to their parents. Addition of a Gnext goal to the explanation may also be useful for other action types, but we have no evidence for that. Thus, we propose the following guideline that forms an exception to guideline 3.

- **Guideline 4:** an action of type *all* whose a parent goal has a *seq* relation should be explained by its goals G+1 and Gnext, and belief(s) B+1.

Another exception to guideline 3 concerns actions of the type *one*. The left side of Figure 4.10 represents a situation where action B is followed by action C or D, for example, the action *Take money* is followed by either *Cycle to the shop* or *Drive to the shop*. Action C and D are explained by goal A (G+1), e.g., *Buy ingredients*. However, a goal can only have one relation to its subgoal/actions, so the goal hierarchy in the left side is not allowed. The right side of Figure 4.10 shows how this situation should be represented. Goal A has a relation *seq* to its children, and a new goal X is introduced, e.g., *Go to the shop*, with a relation *one* to its children. Now, when action C and D are explained by their parent goal X, the explanation is not informative (I cycle to the shop because I want to go to the shop). In this case, it would be better to provide goal A as an explanation (I cycle to the shop because I want to buy ingredients).

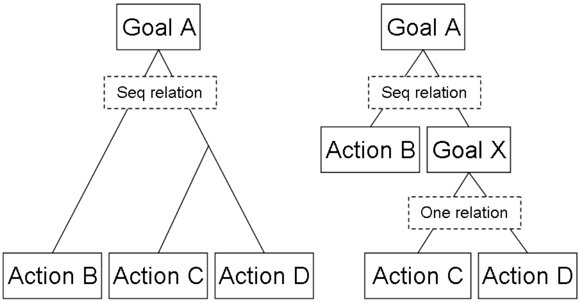


Figure 4.10: Incorrect (left) and correct (right) goal hierarchy with *one* relations.

Although it may result in redundant goal-subgoal relations, we believe that from an explanation point of view a goal should have only one relation to its subgoals, as this simplifies interpretation of the behavior representation. The following guideline also overrules guideline 3, and states that instead of goal G+1, such as dictated by guideline 3, goal G+2 should be provided.

- **Guideline 5:** an action of type *one* should be explained by its goal G+2 and belief(s) B+1.

Guidelines 4 and 5 are two examples of how explanations following guideline 3 can be improved, that is, by providing other beliefs and goals than dictated by the default guideline, or by providing extra beliefs or goals in addition to the default guideline, respectively. Based on our studies, we were only able to propose two action-type-specific guidelines, but there may be more guidelines that would improve the usefulness of explanations.

4.5 Chapter conclusion

In this chapter we presented and analyzed the results of three user studies investigating preferred explanations of virtual player behavior. From the analyses, we extracted a set five of guidelines for developing and explaining cognitive models. In general, the beliefs and goals in a goal hierarchy should be meaningful, as well as the way they are organized. By default, an action should be explained by its goal G+1 and belief(s) B+1. In addition to this general rule, we also introduced two guidelines for specific action types.

The aim of the studies presented in this chapter was to evaluate and improve the explainable agent approach proposed in Chapter 3. The explainable agents developed according to our approach, in turn, aim to help trainees to learn from virtual training. Therefore, in the next section we will validate our approach by testing whether explanations do affect performance.

Chapter 5

Effects of explanation on performance

In Chapter 3, we introduced an approach for developing explainable agents. In Chapter 4, we investigated which of these agents' explanations users prefer. To evaluate our explainable agent approach, including the explanation guidelines introduced in Chapter 4, we performed three empirical studies investigating the effects of explanations on performance, where the explanations were generated according to our BDI-based approach.

The first two studies aimed to investigate the effects of BDI-based explanations on learning from virtual training. We set up two studies in parallel in two domains, onboard firefighting and negotiation, respectively. In the first study, we intended to conduct a between-subjects experiment with an experimental (explanation) and a control (no-explanation) condition, and measure understanding of the training task of all subjects. We were, however, not able to demonstrate a within-subjects learning effect in the experimental condition, and therefore, did not conduct the control condition. Similarly, in the second study, before we conducted a between-subjects experiment comparing an explanation and a no-explanation condition, we tested for a learning effect of virtual training in general. Again, we were not able to demonstrate a learning effect. In Section 5.1 and 5.2 we will describe the two experiments, and discuss possible reasons for not finding a learning effect.

In the third study, described in Section 5.3, we explored whether our approach to explainable agents can also be used for other purposes than training. Namely, we investigated the effects of the explanations on coordination in human-agent teams (Harbers et al., 2011a).

5.1 Learning in firefighting training

In this section, we describe a within-subjects, repeated measurement experiment, testing whether explanations generated according to our approach have an effect on trainees' learning in virtual training.

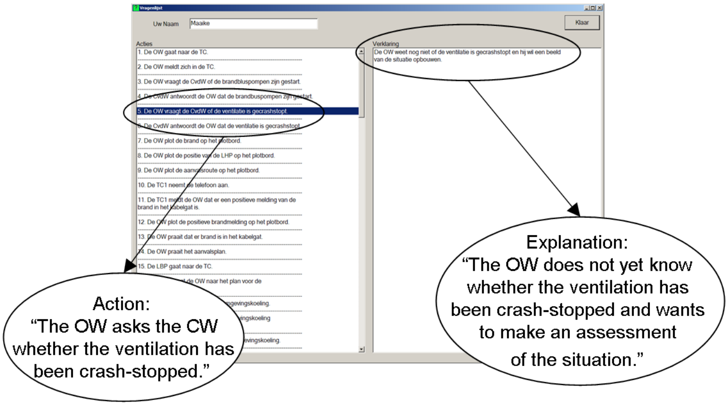
5.1.1 Method

Training task. The training task in the experiment was to deal with a fire aboard a ship. We used CARIM, the virtual training system for onboard firefighting as described in Section 4.1. CARIM is used to train the tasks of an Officer of the Watch (OW), who is in charge of handling fire incidents aboard a navy frigate. From the Machinery Control Room of the ship, the OW contacts his team, develops a plan to contend the incident, gives orders, monitors the events, and adjusts plans if necessary.

Design. The experiment has a within-subjects design, in which understanding of the tasks of an OW is measured twice, before and after the subject receive explanations about agent behavior.

Subjects. The study was conducted with 10 subjects (5 male, 5 female) with an average age of 25 (sd=3.5). The subjects had BSc or MSc education levels, and they were all unfamiliar with the training task.

Materials. We recorded a virtual training session in CARIM in which the OW successfully dealt with a fire aboard a ship. The video takes about 15 minutes. We used a movie instead of letting the subjects play a scenario because they had insufficient task and domain knowledge to finish a complete training session themselves. Moreover, the course of a CARIM training scenario depends on the player's actions, and by using a movie the scenario was equal for all subjects.



Uw Naam Maaike

Acties

1. De OW gaat naar de TC.
2. De OW meldt zich in de TC.
3. De OW vraagt de CvdW of de brandbluspompen zijn gestart.
4. De CvdW antwoordt de OW dat de brandbuspompen zijn gestart.
5. De OW vraagt de CvdW of de ventilatie is gecrashstopt.
6. De CvdW antwoordt de OW dat de ventilatie is gecrashstopt.
7. De OW plot de brand op het plotbord.
8. De OW plot de positie van de LHP op het plotbord.
9. De OW plot de aanvalsroute op het plotbord.
10. De TC1 neemt de telefoon aan.
11. De TC1 meldt de OW dat er een positieve melding van de brand in het kabelgat is.
12. De OW plot de positieve brandmelding op het plotbord.
13. De OW praait dat er brand is in het kabelgat.
14. De OW praait het aanvalsplan.
15. De LBP gaat naar de TC.

de OW naar het plan voor de

mgevingskoeling.

vingskoeling

evingskoeling.

Verklaring

De OW weet nog niet of de ventilatie is gecrashstopt en hij wil een beeld van de situatie opbouwen.

Action:
"The OW asks the CW whether the ventilation has been crash-stopped."
Explanation:
"The OW does not yet know whether the ventilation has been crash-stopped and wants to make an assessment of the situation."

Figure 5.1: The interface of the explanation application with a list of all actions (left) and an explanation (right).

We developed an application through which the subjects could request explanations about the virtual players' actions (see Figure 5.1). These explanations were generated according to the approach described in Chapter 3 and 4. The interface of the application showed a list with all actions of all players (including the OW) in chronological order. By clicking on an action, the corresponding explanation appeared. Explanations can be provided during the training or after the training in an after action review. An advantage of providing explanations during the training is that the user receives guidance during the session and can use the explanations immediately. However, pausing the scenario to provide explanations may interrupt the flow of the game (video in this case), whereas providing explanations while the scenario continues could cause an overload of information to the user. An after-action review allows the trainee to digest explanations calmly without interrupting the scenario. Therefore, we chose to develop an application providing explanations after the training was over.

We developed a test to measure a subject's understanding of the training task. The test contained 40 sentences of which the subjects had to indicate whether they were true or false. Questions were for instance: "The leader attack team and the leader containment team see each other during the scenario" (true) and "Different attack teams are extinguishing the fire simultaneously" (false).

Finally, we developed a questionnaire asking the subjects' opinion on the quality and usefulness of the explanations. The questionnaire contained five statements, and the and subjects could indicate to what extent they agreed with the statements on a scale from 1 to 5, where 1 = completely disagree, 2 = disagree, 3 = neutral, 4 = agree, and 5 = completely agree.

The materials used in this experiment, the video, explanation application and two questionnaires, were all in Dutch. The examples used in this section and the questions mentioned in the results section are thus translated from Dutch to English.

Procedure. First, the subjects watched the recorded training session in CARIM. They were told to make sense of what they were seeing as good as possible, and that they would have to answer questions about the domain and the tasks of an OW. After watching the video, the subjects were asked to make the test with 40 multiple choice questions (pre-test). Subsequently, they were instructed to request explanations about each and every action of all virtual players' except the OW. They had to click on the actions so that they would be actively involved in requesting explanations. Then, the subjects were administered the same 40-item test (post-test). Finally, the subjects were asked to give their opinion on the quality and usefulness of the explanations.

5.1.2 Results

The first time the subjects were administered the test, the average number of errors was 11.8 (sd=3.2) out of 40, and the second time the average number of errors was 11.7 (sd=3.3). The difference is not significant (paired t(9)=0.14, p=0.90).

Table 5.1 shows the results of the final questionnaire, indicating the subjects' opinions. The first question was answered quite positive. In general, subjects indicated that they understood what was happening in the movie. On average, the subjects were neutral about the difficulty of the test. Results on the third and fourth question show that the subjects did on average slightly disagree on the statements that the explanations helped to understand the scenario and answer the questions. As a group, the subjects were neutral

Statement	Score
1. I understood what was happening in the movie.	3.9 (sd=1.2)
2. I found it easy to answer the true/false questions.	2.9 (sd=1.1)
3. The explanations helped to understand the scenario.	2.5 (sd=1.0)
4. The explanations helped to answer the questions.	2.5 (sd=0.8)
5. If I would train for OW, I would request explanations.	3.1 (sd=1.1)

Table 5.1: Average ratings on a 1-5 scale (n=10).

on whether they would make use of the possibility to request explanations if they would train for OW.

5.1.3 Discussion

The goal of this experiment was to test whether explanations generated according to our approach increase trainees' learning. The results show that the explanations did not help trainees to achieve a better understanding of the task. It is likely that these results are, at least partly, explained by a limited usefulness of the explanations. Namely, besides that the explanations did not lead to increased understanding, they were also not perceived as being highly useful.

The results may have been influenced by the way the explanations were offered. Namely, the explanations were provided after the entire video clip of the training session, and not immediately following a virtual player's action. It might be that an explanation is more useful when provided immediately after an action, as the trainee will better remember the context of the action. Though explanations may have a larger effect when provided immediately after an action, several subjects remarked that viewing the mere list of actions in the explanation application was helpful in understanding the scenario (see Figure 5.1). It could be that a combination of explanations during a training scenario and a summary of the played scenario afterwards is most preferable.

In a pilot session before the actual experiment, we asked three subjects to request as many explanations as they liked. These subjects only requested a few explanations. To test the effect of the explanations, we therefore instructed the subjects in the actual experiment to request explanations for each and every action. This is possible in an experiment, but in an actual application it is likely that trainees quit requesting explanations when they are not perceived as being useful.

Some of the subjects remarked that not all answers could be literally derived from the video and the explanations. This was done on purpose, however, to measure general understanding of the task instead of the subjects' memory. Remarkably, several of the subjects said that they found the test easier when they made it a second time, but they did not answer more questions correctly. An explanation may be that these subjects had less difficulty in understanding the questions themselves the second time because they got used to the terminology. So the questions seemed easier, but that was not because the subjects had a better understanding of the domain.

5.2 Learning in negotiation training

In this section we describe a study that investigates whether there is a learning effect of virtual training with explainable agents (Broekens et al., 2011). The study was performed in collaboration with the TU Delft, and as part of another research project, the virtual training also involves emotion expressions of the agents. In order to be complete, we will describe the questions and results related to emotion expression, but the focus is on the role of explanation in virtual training.

There are several differences between this and the previous study. First, the domain in this study is negotiation instead of onboard firefighting. Second, the subjects in this experiment actually play a training scenario and interact with an explainable virtual agent, in contrast to watching a video of a training session. Third, explanations are offered during the training instead of afterwards. Our proposition is that virtual negotiation training with an explainable agent improves negotiation skills.

5.2.1 Method

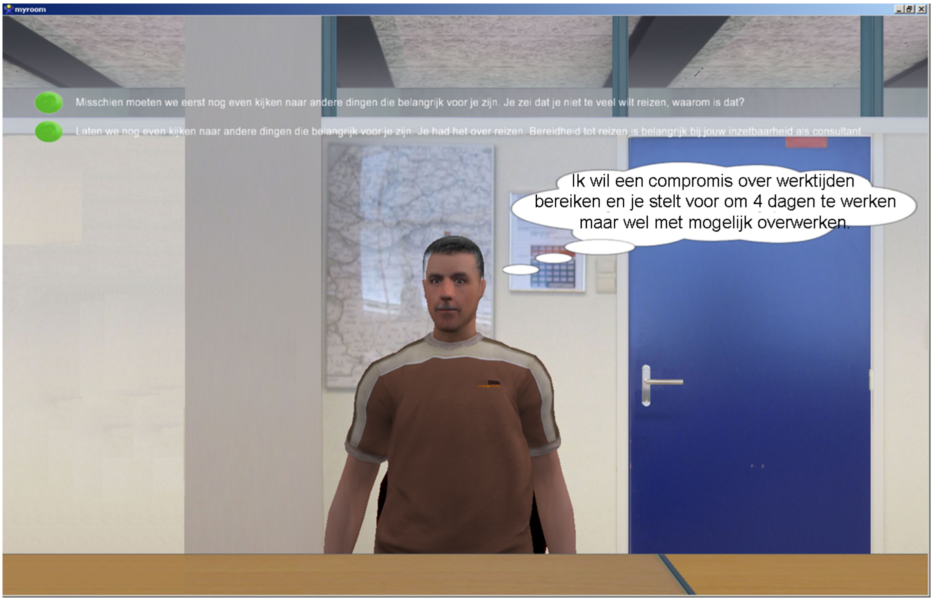
Domain and training task. Literature on negotiation distinguishes (1) a preparation, (2) a joint exploration, (3) a bidding, and (4) a closing phase (Broekens et al., 2010b). The preparation takes place before the negotiation partners meet, and involves the collection of information about one's own and the partner's desires. In the exploration phase, the negotiation partners start to explore each others' wishes. Subsequently, in the bidding phase the negotiation partners can make actual bids, and in the closing phase the partners leave each other with or without an agreement.

A mistake often made by novices is that in the joint exploration phase, they only explore preferences on *issues*, e.g., the height of a salary, and forget to ask about the other's *interests*, e.g., the need for money to pay the mortgage. It is important to learn that by exploring someone's interests, solutions on issues that are profitable for both partners can often be found, e.g., a lower monthly salary but with a yearly bonus. The training scenario we used in this study targets at helping trainees to become aware of the importance of interests in achieving a good outcome of a negotiation.

Design. We performed a between-subjects experiment with an experimental and a control condition. The subjects in the experimental condition received virtual training with explanations and emotion expressions, and the subjects in the control condition did not receive virtual training. The subjects were randomly assigned to a condition.

Subjects. A total of 18 subjects (12 male, 6 female) with an average age of 27.0 (sd=4.0) participated in the experiment, 9 subjects in each condition.

Materials. We used a virtual training system for negotiation that has been developed at the TU Delft in the context of a large research project on negotiation support systems (Hindriks and Jonker, 2008). The training scenario and the explanation capabilities of the virtual agent were developed for this study in particular.



Misschien moeten we eerst nog even kijken naar andere dingen die belangrijk voor je zijn. Je zei dat je niet te veel wilt reizen, waarom is dat?

Laten we nog even kijken naar andere dingen die belangrijk voor je zijn. Je had het over reizen. Bereidheid tot reizen is belangrijk bij jouw inzetbaarheid als consultant.

Ik wil een compromis over werktijden bereiken en je stelt voor om 4 dagen te werken maar wel met mogelijk overwerken.

Figure 5.2: Impression of the virtual negotiation training system. A translation of the Dutch text in the thinking cloud is: “I want to reach a compromise on working hours, and you just proposed to work 4 days but with the possibility of overtime work.”

The training scenario involves a negotiation about terms of employment. A human player, the trainee, has the role of an employer, and a virtual agent is playing the candidate employee. The trainee can interact with the agent by selecting a sentence from two to four offered options (see Figure 5.2). The virtual agent communicates in natural speech, pre-recorded by a professional voice actor. Furthermore, the agent is able to express three basic emotions as feedback to the trainee’s selected response option: happiness, sadness, and anger. Happiness signals a -for the virtual character- positive outcome of a chosen option, sadness signals a potentially bad outcome, while anger signals an actual bad outcome. The system keeps track of how often a trainee makes the agent happy, sad, and angry in one session. These are the emotion counts. The training scenario was reviewed and approved by a professional negotiator.

The training scenario focuses on the exploration phase of a negotiation. The trainee and the character explore each others’ standpoints on four topics, i.e., contract type, salary, working hours, and traveling time. The sentences that the trainee can select throughout the scenario either address interests or issues. The issues and interests in the scenario are based on a diary study in which 8 subjects were asked to keep track of their negotiation for a new job or a new house. When all topics are explored, the trainee constructs one complete bid by selecting a partial bid for each of the four topics. The partial bids that the systems offers the trainee to select from, depend on his behavior in

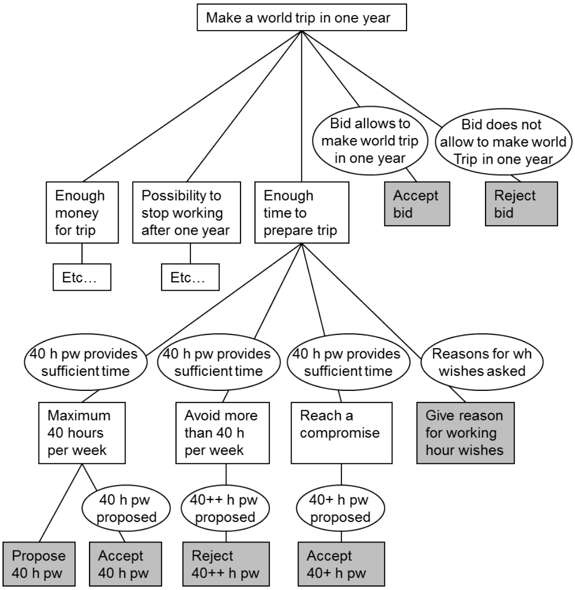

Make a world trip in one year
Bid allows to make world trip in one year
Bid does not allow to make world Trip in one year
Enough money for trip
Possibility to stop working after one year
Enough time to prepare trip
Accept bid
Reject bid
Etc…
Etc…
40 h pw provides sufficient time
40 h pw provides sufficient time
40 h pw provides sufficient time
Reasons for wh wishes asked
Maximum 40 hours per week
Avoid more than 40 h per week
Reach a compromise
Give reason for working hour wishes
40 h pw proposed
40++ h pw proposed
40+ h pw proposed
Propose 40 h pw
Accept 40 h pw
Reject 40++ h pw
Accept 40+ h pw

Figure 5.3: Part of the goal hierarchy of the negotiation agent.

the exploration phase. Only when the trainee completely explored the agent's interests on a topic, a partial bid leading to a win-win solution appears among the options. When the trainee did not completely explore the agent's interests on a topic, only options with compromise agreements and no agreements appear. The value of the trainee's complete bid is expressed by a number between 0 and 8, representing the outcome utility of the deal.

The agent was developed according to our explainable agent approach, and thus able to explain its actions to the trainee. Figure 5.3 shows a part of the goal hierarchy of the negotiation agent. Note that in this domain, the agent's actions are all speech acts. The action to reject a bid of more than 40 hours per week is for instance explained by the goal to avoid working more than 40 hours per week, and the belief that the employer just made an offer to work more than 40 hours per week.

In contrast to the previous study, we chose to offer explanations about agent behavior during the training because they are most relevant at that time. A disadvantage of providing feedback during a training is that it can interrupt the flow of the scenario. We aimed to avoid that by offering the explanations in a natural way, i.e., in the form of thinking clouds (see Figure 5.2).

Procedure. Table 5.2 shows the procedures for both conditions. First, all subjects were asked to answer four questions about their negotiation experiences, the *negotiation questions*. They rated their (1) daily life self-assessed negotiation skills, (2) negotiation liking,

(3) self-assessed negotiation frequency, and (4) negotiation perseverance when negotiating. Rating was on a 5-point Likert scale.

Second, the subjects in the experimental condition played the negotiation scenario with explanations and emotions, which we refer to by *training*. Subjects were instructed to play as well as possible. This is the experimental manipulation.

Third, after the training, the subjects in the experimental group had to answer (5-point Likert) questions on whether they understood the agent's considerations, whether the agent made clear what it wanted and thought, whether emotions played a role, and whether the virtual agent expressed emotions. These four questions are called the *perception questions*.

Fourth, we asked all subjects to watch five short recorded negotiation scenes and to describe what happened during the scene. We call this the *scenes with questions*. The subjects' descriptions will be evaluated in future work by a professional negotiator on how much they show negotiation proficiency. We thus cannot provide the results of this measure here.

Fifth, all subjects ended the experiment playing the negotiation scenario without emotion expressions and explanations, this is the *test*. Subjects were again asked to play as well as possible. Explanations and emotions were omitted in this scenario to be able to test the influence of emotion and explanation in training in future experiments.

Sixth, subjects in the control condition answered the perception questions after the test. Thus, subjects in both groups answered the perception questions immediately after the first time they did the negotiation scenario.

Measures. As outcome measures (dependent variables) we compared subjects performance on the test between the experimental and control condition. We counted the number of times the subject made the agent happy, sad, and angry (even though these emotions were not expressed by the avatar in the test scenario) as well as recorded the outcome utility of the deal (a number between 0 and 8). As mentioned above, future work includes subjects' scene descriptions coded by a professional negotiation expert.

Experimental condition	**Control condition**
Negotiation questions	Negotiation questions
↓	
Training (expl+emo)	
↓	↓
Perception questions	
↓	
Scenes with questions	Scenes with questions
↓	↓
Test	Test
	↓
	Perception questions

Table 5.2: Procedures in the experimental and control condition.

5.2.2 Results

Table 5.3 shows the subjects' performances on the training and test. To investigate our proposition that virtual negotiation training with an explainable agent improves negotiation skills, we first performed a within-subjects analysis (n=9) of the training using a repeated multivariate analysis with as output variables the outcome utility and the emotion counts (happy, angry and sad). The results showed that the subjects' performances were not significantly different for the training versus the test, thus no measurable learning effect of training (F(4, 5)=0.311, p>0.1). To further test our proposition, we then performed a between-subjects analysis (n=18) comparing the performance on the test between the subjects in the control and the experimental condition. This again did not show a significant result (F(4, 13)=1.02, p>.1).

	Exp. group Training	**Exp. group Test**	**Control group Test**
Outcome utility	3.6 (sd=1.3)	4.3 (sd=1.9)	3.7 (sd=1.3)
Agent made happy	8.6 (sd=5.0)	10.8 (sd=4.5)	7.4 (sd=2.7)
Agent made sad	3.6 (sd=1.9)	5.6 (sd=2.2)	7.0 (sd=2.1)
Agent made angry	1.7 (sd=0.9)	1.2 (sd=1.0)	1.8 (sd=1.4)

Table 5.3: Average outcome utilities (0-8) and emotion counts of subjects in the experimental and control group on the training and test.

In order to test whether explanations during negotiation affected the subjects' understanding of the agent, we performed a multivariate between-subjects analysis (n=18) with the condition (experimental versus control) as independent, and the scores on the four perception questions as dependent variables. This showed a significant effect (F(4, 13)=3.45, p=0.04). Looking at the univariate effects on the four questions separately, we found that the effect on whether emotion played a role was in the opposite direction of what was expected. Namely, emotion played a more important role in the test (m=3.9) than during the training (m=2.9) (F(1,16)=7.3, p=0.015), while during the training the agent expressed emotions and during the test it did not. The effect on whether the agent explained what he wanted was in the expected direction, i.e., higher (m=4.4) for the training than for the test (m=3.8), (F(1,16)=3.27, p=0.89). Apparently, subjects tended to understand agents showing their considerations and goals better than agents showing no underlying thoughts.

Finally, we performed a correlation analysis of the negotiation questions with the negotiation performance (see Table 5.4). The analysis showed that only the correlation between self-reported negotiation frequency and the number of times the subject made the agent sad was significant, while the correlations between frequency and utility as well as the number of times the subject made the agent happy approached significance. This indicates that performance on the virtual reality simulation depends on actually doing negotiations in real life showing convergent validity of the training.

	Utility	Happy	Sad	Angry
Skill	0.070 0.782	0.050 0.844	0.069 0.786	-0.361 0.141
Liking	0.163 0.519	0.246 0.326	-0.151 0.550	-0.290 0.243
Frequency	0.418 0.085	0.443 0.066	**-0.498** **0.036**	-0.204 0.416
Perseverance	0.289 0.245	0.174 0.489	0.080 0.754	-0.357 0.146

Table 5.4: Pearson correlations (on top) and significances (below) between measured performance and perceived subjects' negotiation behavior (n=18).

5.2.3 Discussion

Our results show that, even if a virtual training is setup in a rigorous way (expert evaluation of scenario, based on case studies of subjects, validated affective expressions, validated explanation method, a simple scenario setup with only one training goal), it is hard to show measurable effects of virtual training alone. We give five reasons that could explain this finding.

First, we used a relatively small number of subjects (n=18), and the results may be caused by a lack of power. Second, there is evidence in education literature that learning from pure exploration is less efficient than from guided exploration (Kirschner et al., 2006). We did not provide additional instructions to the subjects because we were primarily interested in the effects of the training itself. Third, subjects were asked to perform as good as possible in both the training and the test. It may be, however, that during the test, the subjects explored other possibilities than in their first session. Thus, they may have explored more in the test instead of trying to optimize their performance. When their training performance was already quite good, this exploration may have suppressed a possible within-subjects effect. Fourth, the analysis of five negotiation scenes before the test may have influenced the subjects' test performance in a positive way. This influence may have overshadowed the effect of the training. Fifth, our external negotiation expert has not yet analyzed the textual replies on the questions about the movie scenes. It is well possible that there is a measurable effect here.

We have provided five reasons that may explain the lack of measuable effects of virtual training alone. To determine which of them explain(s) our results, more research is needed.

5.3 Team coordination

The previous two studies investigated the effects of explanations on learning. In contrast to those, the study described in this section tests the effects of BDI-based explanations about agent behavior on coordination in human-agent teams. Humans easily adapt their behavior to entities with other cognitive abilities than their own. For instance, most people automatically choose simpler words when they talk to children than when they talk to adults, and many people are able to interact well with their pets even though those cannot speak at all. Also in human-agent interaction, the key of good interaction is not that the other, the agent in this case, acts as human-like as possible. Instead, agents should facilitate humans' adaptation to them by being transparent and observable (Bradshaw et al., 2011). Knowing more about an agent, e.g., its capabilities, goals, and intentions, allows humans to understand and predict the agent's behavior better, and to adapt their own behavior to that of the agent more easily. Therefore, we believe that explanations about agent behavior can improve coordination in human-agent teams.

This idea links to the teamwork-centered approach when designing autonomous systems, called *coactive design*, proposed by Johnson et al. (2011). According to the coactive design approach, the design of autonomous agents should be led by the underlying interdependence of the joint activity in the human-agent system. In other words, understanding of the agents' joint activities should be used to shape the implementation of agent capabilities. When human-agent teams have to perform tasks that involve a large amount of coordination, it is important that the human(s) can understand and predict the behavior of the agent(s). Therefore, in such situations, agents should be capable of providing explanations that increase human understanding in their behavior.

To test the effects of explanations on human-agent coordination, the BlocksWorld for Teams (BW4T) testbed for team coordination was used (Johnson et al., 2009). In BW4T, a team of players has to perform a joint task in a virtual environment, and the performance of the team strongly depends on the level of coordination among the players. BW4T allows for games with human-human, agent-agent, and human-agent teams of variable sizes. Our proposition is that human-agent teams perform better on the BW4T task when agents explain their behavior.

5.3.1 Method

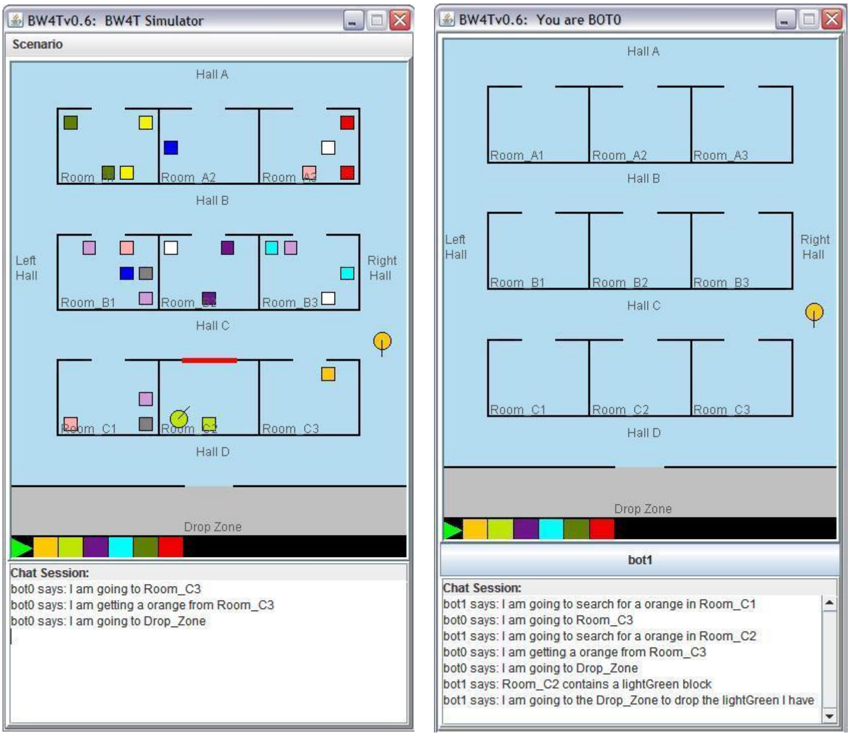
Design. The experiment has a within-subjects design with an explanation and a no-explanation condition. In the explanation condition, the subjects cooperate with an agent explaining its behavior, and in the no-explanation condition, subjects cooperate with an agent that does not explain its behavior. The order of the two conditions, explanation and no-explanation were assigned counter-balanced to the subjects, to correct for possible learning effects from the first to the second trial.

Subjects. A total of 16 subjects (male = 14, female = 2) with an average age of 27 (sd=3.5) participated in the experiments.

Materials. In the BW4T coordination test-bed[1], the team's goal is to deliver a sequence of colored blocks in a particular order as fast as possible. A complicating factor is the that players (human or agent) cannot see the blocks from every position, and they cannot see each other. Figure 5.4 displays a screenshot of a BW4T game session, showing the

[1] The BW4T test-bed has been developed at the Florida Institute for Human and Machine Cognition (IHMC)

environment in which the players have to search for blocks. The left picture displays all blocks and players in the game, and the right picture shows what one player can see. A player can only see the blocks in a room when he is inside that room. The status bar below the Dropzone (gray area) shows which blocks need to be delivered.



Chat Session:
bot0 says: I am going to Room_C3
bot0 says: I am getting a orange from Room_C3
bot0 says: I am going to Drop_Zone

Chat Session:
bot1 says: I am going to search for a orange in Room_C1
bot0 says: I am going to Room_C3
bot1 says: I am going to search for a orange in Room_C2
bot0 says: I am getting a orange from Room_C3
bot0 says: I am going to Drop_Zone
bot1 says: Room_C2 contains a lightGreen block
bot1 says: I am going to the Drop_Zone to drop the lightGreen I have

Figure 5.4: Simulator view (left) and player view (right). The blocks that need to be delivered are respectively orange, light green, dark purple, light blue, dark green and red. Bot0 in the right hall is holding an orange block and bot1 in room C2 is holding a light green block.

To deliver a block successfully, a player has to pick up a block of the right color and drop it in the Dropzone. A player can only carry one block at a time. When a player drops a block of the wrong color in the Dropzone or any block in a hall, the block disappears from the game. Human players can perform actions in the environment through a menu that appears on a right mouse-button click. The menu offers options to go to a place (room, hall or Dropzone), pick up a block, drop a block and send messages.

A team's performance on the BW4T task is measured by the speed of completing the task. Time of task completion is the dependent variable in the experiment. BW4T is designed such that the task involves a large amount of interdependence among the players, and requires coordination to achieve a good performance. For instance, it is inefficient when one player is searching in a room that has just been checked by another. And if a player is going to deliver a particular block, the others should not do that as well.

To coordinate, players can send messages to each other, which appear in the chatbox below the Dropzone. Players can inform others about what they do, where they are and what they see. Furthermore, players can see the same status bar. So when a player delivers a block of the right color, the other players will know. Finally, only one player can be inside a room or the Dropzone at the same time. When a player tries to enter a room that is occupied, a red bar appears indicating that someone is inside.

For this study we designed a cooperative agent, which assumes that other players are cooperative as well. The agent displays the following behavior. It starts to check rooms and once it knows about a block that can be delivered, it starts to deliver that block. The agent uses information about blocks in rooms received from other players. When another player announces that he is going to check a particular room, the agent will not check that room. When another player tells that he is going to deliver a block, the agent will start to search or deliver the next block in the sequence. The agent is able to deal with humans that vary their strategy, make mistakes, and forget to tell things. Namely, the agent revises its plans when a room contains other blocks than it expected, and when the agent holds a block that is not needed anymore, it will drop the block in a room.

As the aim of this study is to explore the effect of explaining agent behavior on coordination in human-agent teams, we need to be able to manipulate the agent's communication behavior. Inspired on the KaOS policy framework (Bradshaw et al., 2003), we use policies to regulate the agent's behavior. This is practical because it avoids changing the agent's programming code in order to adapt its communication behavior. We distinguish the following three communication policies.

- Inform other players about your observations
- Inform other players about your actions
- Provide explanations for your actions

The first policy entails that if the agent observes something in the virtual environment, it sends a message to inform all other players about its observation. Such messages are, for example, 'Room A1 contains a pink block and a dark blue block' and 'Room B2 is empty'. The second policy prescribes that if the agent performs an action, it has to send a message to inform all other players about it. Messages informing about actions are for instance 'Im going to Room C1', 'I picked up a red block' and 'I just dropped a gray block'. The third policy prescribes the agent to explain an action, that is, to provide the underlying goal of that action. Examples explanations for actions are 'I am going to Room B3 to search for an orange block' and 'I am going to Room C2 to deliver a light green block'.

Note that explanations only contain a goal (generated by algorithm G+1), and no beliefs (which could be generated by algorithm B+1). In this context, the agent's beliefs are its observations, and the agent's communication about its beliefs are controlled by a separate communication policy, i.e., the first policy. We believe that it is important to separate the communication about beliefs and goals in this application because that allows the agent to inform players about its observations when they are made, and not just when explaining an action.

BW4T is implemented in Java and offers a basic agent class in which a BW4T agent's behavior can be implemented. At the TU Delft, a connection between BW4T and the BDI-based (Belief Desire Intention) programming language `GOAL` (Hindriks, 2009) has been established, which also allows for the implementation of BW4T agents in `GOAL`. Following our explainable agent approach, we implemented our BW4T agent in `GOAL` so that it was able to explain its behavior in terms of the goals and beliefs underlying its actions.

Procedure. The subjects received an explanation of the BW4T task and how to direct their 'bot'. Subsequently, they had to play a familiarization session, in which they had to deliver three blocks on their own. This session was included to make sure that the subjects understood the game, knew how to control their bot, and to give them time to think about their strategy in the actual trials. No agent participated in the training session yet, to prevent that it would shape the subjects' expectations about the agents in the trial sessions.

For the two trial sessions, subjects were instructed to perform the task with the agent as a team, as fast as possible. They were told that the agent could show any kind of behavior, e.g., not search in the right places or not take the subject's messages into account, but that the agent would not lie to them. In both trial sessions, the human-agent team delivered six blocks of different colors. The colors and positions of the blocks differed per session, but the total traveling distance to deliver all blocks was the same. After both sessions, the subjects were asked to fill in a short questionnaire.

5.3.2 Results

The time of completing the BW4T task was used as a measure for team performance. In the explanation condition, the average time (n=16) to complete the task was 596 seconds (sd=118), and in the no-explanation condition the average time was 593 seconds (sd=81). These averages are obviously not significant (paired t(15)=0.07, p=0.95).

We also examined whether there was a learning effect from the first to second session. The average time (n=16) to complete the sessions was 617 seconds (sd=118) for the first session, and 572 seconds (sd=76) for the second session. These results show that the subjects completed the task faster in the second session than in the first session, but the difference is not significant (paired t(15)=1.2, p=0.26).

In the questionnaire administered after each session, we asked subjects to judge their own, the agent's, and their common performance on a scale from 1 to 7. Table 5.5 shows the averages in both the explanation condition (EX) and the no-explanation condition (NE).

Although the average rated performance is higher in the explanation condition than in the no-explanation condition, statistical analysis shows that these differences are not significant (paired t(15)=0.4, p=0.67; paired t(15)=0.9, p=0.36; paired t(15)=0.8, p=0.41, respectively).

In order to investigate how well subjects evaluate performance, we calculated the correlations between the self-evaluations in Table 5.5 and the actual team performances. Surprisingly, the subjects' self-evaluations have a low or even negative correlation with

	EX	NE
I was effectively performing the task	5.9 (sd=0.7)	5.8 (sd=1.1)
The agent was effectively performing the task	6.0 (sd=1.3)	5.5 (sd=1.3)
We were effectively performing the task as a team	5.7 (sd=1.6)	5.1 (sd=1.7)

Table 5.5: Estimations of performance on a 1-7 scale (n=16).

the actual performances. Three of the negative correlations are significant (α=0.05): evaluated human performance and actual team performance in the no-explanation condition (R=-0.49), evaluated agent performance and actual team performance in the explanation condition (R=-0.50), and evaluated team performance and actual team performance in the explanation condition (R=-0.55). The results show that subjects make better estimates of their own performance in the explanation condition, and better estimates of the agent's and the team's performance in the no-explanation condition.

In the questionnaire we also asked the subjects to judge how well they understood the actions and motivations of the agents, and how well the agents seemed to understand their actions and motivations. The results in Table 5.6 show that the subjects in the explanation condition had a significantly better idea of what the agent was doing than subjects in the no-explanation condition (paired t(15)=2.4, p=0.030). Though the other results are not significant, for all questions understanding was on average rated higher in the explanation than in the no-explanation condition (paired t(15)=0.3, p=0.74; paired t(15)=0.5, p=0.65; paired t(15)=0.7, p=0.47, respectively).

	EX	NE
I had a good idea of what the agent was doing	6.1 (sd=1.0)	5.1 (sd=1.4)
The agent seemed to have a good idea of what I was doing	5.8 (sd=1.1)	5.7 (sd=1.0)
I understood the reasons behind the agent's behavior	5.9 (sd=1.2)	5.7 (sd=1.5)
The agent seemed to understand the reasons behind my behavior	5.6 (sd=1.0)	5.3 (sd=1.9)

Table 5.6: Average understanding of behavior on a 1-7 scale (n=16).

Finally, we asked subjects if the agent provided *too little*, *just enough*, or *too much* information. In the explanation condition, one subject indicated that the agent provided too little information, and all other subjects indicated that the agent provided just enough information. A chi-square goodness of fit test shows that the result is significant (χ^2=26.4, p<0.001). In the no-explanation condition, ten subjects indicated that the agents provided too little information, while six subjects indicated that the provided information was just enough. This result is significant as well (χ^2=9.5, p=0.009). Thus, in general subjects preferred the amount of information in the explanation condition over the amount of information in the no-explanation condition.

5.3.3 Discussion

We found no significant differences between human-agent team performance in the explanation and the no-explanation condition. Therefore, the results do not support our proposition that explanations about agent behavior improve human-agent team performance. The experience of the subjects, however, was affected by the agent's explanations. The subjects' ratings of their idea of what the agent was doing was significantly higher in the explanation condition than in the no-explanation condition. Furthermore, a significant number of subjects believed that the agent in the no-explanation condition provided too little information, whereas a significant number of subjects indicated that the agent in the explanation condition provided just enough information.

With a larger number of subjects, more of the results obtained from the questionnaire may have been significant. Namely, all ratings were, on average, higher for the explanation condition than for the no-explanation condition, both concerning self-evaluations on performance as understanding of each other's actions. It is not probable that the difference in performance on both conditions will become significant if tested with a larger number of subjects, since the performances on both conditions are rather similar.

There are several possible explanations for the similar team performances on both conditions. We provide five of them. First, subjects may have lost time in processing the agent's explanations, which then was compensated by a more efficient task completion. The robots in BW4T move slowly on purpose to provide players sufficient time to communicate, and think and process information. However, at some points in the game many actions have to be done at once (enter a room, go to a block, pick up a block, go to the Dropzone, and communicate about your actions) despite of the slow speed of the robots. Thus, at those time points, processing explanations may lead to time loss.

Second, the subjects may have anticipated a cooperative agent. Though we told them that the agent could perform any behavior, several of the subjects reported that their strategy was to behave as if the agent was cooperative until they would find out otherwise. With such a strategy, explanations do not contribute to a quicker adaptation to the agent's behavior as the subject's initial behavior already makes the right assumptions about the agent's behavior. It would be interesting to conduct an experiment with a less cooperative or capable agent, e.g., one that cannot process certain messages or is colorblind, to see if explanations help subjects to adapt quicker to the gaps in the agent's capabilities.

Third, the task may involve too much random variability obscuring the effect of the manipulation. Some of the subjects, for instance, reported that they mistook one color for another (e.g., yellow and light green), which caused a serious delay in task completion. Other subjects said that they changed their strategy after the first trial, e.g., they let the agent deliver all blocks. Furthermore, though the blocks are evenly spread over the rooms in different trials, there is a luck factor involved in finding the right blocks. This factor can be decreased by letting the team deliver more blocks, but adding blocks also gives the subjects more time to learn the agent's behavior, which decreases the expected effect of providing explanations. In conclusion, noise factors like these may have wiped out the effects of explanation on team performance.

Fourth, the task may be too simple to show an effect. In most situations, the rationale behind the agent's behavior can be deduced from its actions.

Fifth, the agent always explained its actions by the goals they aimed to achieve. The advantage of such explanations is that they are immediately derivable from the mental state of a BDI agent. Possibly, when extending the agent's explanation capabilities, e.g., by adding information about the agent's strategies, the explanations would become more useful and have a bigger effect on team performance.

5.4 Discussion

The three studies in this chapter show no effects of explanations about agent behavior on learning from virtual training (Section 5.1 and 5.2) or on team coordination in human-agent teams (Section 5.3). We provided possible reasons for that in the discussion sections of each study. In addition to these reasons provided per study, we discuss some overall findings and observations of the three studies here.

First, we would like to remark that it is not only difficult to show the effect of explanations in virtual training on learning, it is even hard to demonstrate the effects of virtual training on learning in general. Though virtual training is often reported as being effective, rarely are such systems tested for a clear learning effect (however, for an exception see Graesser et al., 2005). For instance, there are several accounts of virtual training for negotiation that involve emotions and/or explanations (e.g., Core et al., 2006a,b; Cassell and Bickmore, 2003; Reilly and Bates, 1995; Traum et al., 2003, 2008), but it is not demonstrated if and when such systems actually produce a learning effect in the trainees.

Second, we made two observations that could indicate that explanations in virtual training are useful. The first observation is as follows. In the third study we tested the effect of an agent's explanations on coordination in human-agent teams. We observed that explanations are especially important when a human starts to cooperate with an agent. Subjects reported that, in the beginning, they paid much attention to the agent's explanations in order to understand its behavior. Later in the process, they already knew how the agent would behave, and the explanations became less important. This observation supports the use of explainable agents in virtual training. Namely, trainees are usually novices in the training domain and still have to learn how to cooperate or deal with the other players they will encounter. The explanations are thus provided at a moment they can be most useful.

Our second observation is the following. In the third study we also observed that the agent's explanations affected the subjects' experiences in a positive way. The subjects better understood the agent's actions when they explained their behavior, and they were also more satisfied with the amount of information the agents provided when they explained their behavior. We did not formally ask the subjects, but it could be the case that, even though no effects on learning are found, trainees may enjoy training more when virtual agents explain their behavior. This may lead to increased motivation and prolonged training time, which would be a reason to add explanation capabilities to agents in virtual training.

Third, we provide two remarks about the way in which explanations were offered to the trainees. Namely, though we focus on the *content* of explanations about agent behavior in this thesis, in a validation study of an explainable agent approach for virtual training choices about the way explanations are offered have to be made. Our first remark concerns the timing of explanation provision. An important difference between the first and the last two studies is that in the first study, explanations were offered after the training session was over, and in the last two studies, they were offered immediately following actions. It seems that the type of explanation influences the preferred timing. For instance, explanations that explain an agent's strategy, may be more relevant after the trainee has seen more of the agent's behavior than just a single action. In our approach, in contrast, actions are explained by the goals and beliefs of an agent at that time. Therefore, the explanations are probably more useful when offered immediately after the actions.

Our second remark related to explanation provision concerns on whose initiative explanations are provided. In a pilot of the first study, explanations were only provided on request of the trainee. However, the three subjects we tested did not request many explanations. As trainees may not always be aware that an explanation could help them, it seems better to provide them on the initiative of the system, like in the second and third study. In the second study, we used thinking clouds. In the third study, the explanations were actually part of the agent's communication behavior, instead of a separate facility. None of the subjects reported any problems or issues with the way explanations were provided in these studies, and both ways seem to be a nice solution to provide explanations without interrupting the training session.

5.5 Chapter conclusion

In this chapter we have not been able to demonstrate the usefulness of explanations of agent behavior on performance, i.e., learning in virtual training and team coordination. However, taking all the issues raised in the discussion sections into account, we cannot conclude that the explanations are not useful either. Based on our experiences and observations during the studies, however, we believe that the explanations become more useful when agents take the perspectives of other agents more into account. This is supported by the CARIM evaluation study in Chapter 4, in which subjects most of the time indicated that they preferred explanations that involved a goal or belief attributed to another agent. In the next chapter, we will further explore this possibility.

Chapter 6

Theory of mind-based explanations

In this chapter we will investigate how to equip agents with a theory of mind. A theory of mind is a ‘theory’ about other agents’ minds. In other words, someone with a theory of mind has the ability to attribute mental states such as beliefs, intentions, and desires to others in order to understand, explain, predict or manipulate others’ behavior better. By extending explainable agents with a theory of mind, they will become able to use assumptions about the mental states of other agents in explanations about their own behavior. We believe that this will improve the usefulness of their explanations for trainees.

Typical mistakes that occur during incident management include (1) giving incomplete or unclear instructions, (2) forgetting to monitor task execution, and (3) failing to pick up new information and quickly adapt to it. Many of these errors involve situations in which people make false assumptions about others’ knowledge or intentions. The tendency to attribute incorrect knowledge and intentions to others appears in stories of professionals (Flin and Arbuthnot, 2002), but it is also a well described phenomenon in general in cognitive sciences (Nickerson, 1999; Keysar et al., 2003). Thus, in order to create interesting learning situations, virtual agents should have the ability to (fail to) take others’ assumed knowledge and intentions realistically into account. Moreover, to help trainees to learn from such situations, they should be able to include assumed knowledge and intentions of others in explanations about their behavior. This is especially important in teamwork (Yen et al., 2004), i.e., when players are dependent on each others’ actions for achieving their own tasks.

In this chapter we will first provide an example of a training scenario to illustrate the use of agents with a theory of mind in virtual training. Subsequently, we provide an overview of theory of mind literature. Then, we introduce two ways to model an agent with a theory of mind, and perform a simulation study in which the two approaches are compared. Next, we discuss two extensions to the BDI-based programming language 2APL that facilitate the implementation of agents with a theory of mind. We end this chapter with an overview of related research and a conclusion.

6.1 An example training scenario

The present example is part of a virtual training scenario for on-board firefighting (the scenario is inspired on the CARIM system, discussed in sections 4.1 and 5.1). The trainee plays the role of H-Officer, the person in command when there is a fire aboard of a navy frigate. Besides the trainee, two other players are involved, an A-Officer and an E-Officer, played by intelligent agents. The H-Officer leads the incident management from the Technical Center of the ship. His tasks involve assessing the situation, developing a plan, instructing other officers, monitoring task execution, and adapting plans if necessary. The E-Officer is also located at the Technical Center and is responsible for the electricity at different compartments of the ship. The A-Officer leads the fire attack at the location of the incident and can only use water in compartments where the electricity has been switched off. The H-Officer can communicate with all officers and vice versa, but there is no direct communication between the E-Officer and A-Officer possible.

In the optimal situation, if there is a fire, the E-Officer switches off the electricity in the right compartments and reports this in person to the H-Officer. Subsequently, the H-Officer broadcasts the message to the ship, and the A-Officer orders his team to attack the fire with water. As a result, the fire will be extinguished, which the A-Officer reports to the H-Officer. In this scenario course, the agents understood each others' and the trainee's goals, and acted proactively to support each other. The trainee received positive feedback in the form of a good end result, and explanations of the agents can even increase his understanding of the played session. For instance, the E-Officer may explain that he switched off electricity to ensure that the A-Officer could safely attack the fire with water. By such explanations the trainee learns not only which, but also why certain procedures have to be followed.

When trainees start training with scenarios, they are expected already to have knowledge about the procedures in the domain, e.g., the division of tasks, and where to find information. In the beginning, it will be challenging for them to apply this knowledge in a realistic scenario in which all agents act as they should. Agents may even help the trainee when he fails to undertake required actions, e.g., by giving advice. For example, the trainee may fail to broadcast the E-Officer's message. In such a case, it might be useful if the E-Officer advices the trainee to broadcast the message or if the A-Officer asks the trainee whether the electricity has been switched off. The trainee will become aware of his omission and no longer delay the fire attack. A useful explanation for the A-Officer's action could be that it believed that the trainee would know about the status of the electricity.

At a later stage, when the trainee can easily play a scenario in which all agents act as they should, the scenario can be made more challenging. For more advanced trainees, mistakes of virtual agents can create interesting learning situations. The E-Officer could for example fail to switch off electricity, forget to report to the trainee, or switch off electricity in a wrong compartment. The trainee is challenged to correct the agents, for instance by asking the E-Officer whether he already switched off electricity. An explanation of the E-Officer's failure could be that he believed that the A-Officer did not plan to use water for his fire attack.

Though the given situation is a simple one, several capabilities are required to provide training as described above. The intelligent agents should be able to attribute mental states to others, know when to help the trainee, make believable mistakes, and explain their own actions by sharing their assumptions about other agents' states. In the example, interaction plays an important role and the different agents (including the trainee) are dependent on each other for successful task execution. In order to generate and explain the behaviors in the example, the agents have to be aware of the others' tasks and the consequences of their actions for others. In other words, the agents need some theory about the other agents' mental states: a theory of mind.

6.2 Background: theory of mind

To understand the social world around them, people interpret others' and their own actions in terms of mental states. A theory of mind is the ability to understand others as intentional agents, and to interpret their minds in terms of intentional concepts such as beliefs and desires, e.g., *R believes that M intends him to persuade A that p*. The term 'theory of mind' originates from Premack and Woodruff's famous paper 'Does the chimpanzee have a theory of mind?' (1978). Since then, the term has also been used in other research disciplines to denote the ability to explain and predict one's own and others' behavior. Namely, besides biologists, researchers in the neurosciences, psychology, and philosophy have been involved in theory of mind research as well.

Humans are not born with a fully developed theory of mind, but acquire one during their childhood. The false-belief task (Wimmer and Perner, 1983) is often used by developmental psychologists to determine whether someone has a fully developed theory of mind. To test whether a child passes the task, an experimenter puts an object in a box in presence of the child and another person. The other person leaves the room and when she is gone, the experimenter puts the object in a different box. When the person returns the child is asked where she will look for the object. The child fails if it answers that the person will look in the second box. Though the child knows that the object is in the second box, to pass the task it should be able to understand that the other person did not see that the object was replaced and thus will look in the first box. Experiments demonstrate that children obtain the ability to perform this task well around the age of four years old (see, e.g., Wimmer and Perner, 1983).

A second contribution of psychology to theory of mind research are studies about the absence of a theory of mind, also called mind-blindness, with autists (Baron-Cohen, 1995). A mind-blind person has difficulties to determine the intentions of others and lacks understanding of how his behavior affects others.

Though psychologists studied theory of mind acquirement and theory of mind impairment, most of them did not focus on the question how a fully developed theory of mind in adults works. Philosophers, in contrast, are focusing on exactly this question. Currently, the debate involves two prominent accounts on human, adult theory of mind: theory-theory and simulation theory. According to theory theorists (see, e.g., Carruthers, 1996), people have an implicit *theory* of the structure and functioning of the human mind. This theory involves a set of concepts, e.g., beliefs, desires, and plans, and principles about how these concepts interact, e.g., people act to fulfill their desires. This theory allows us to understand, explain, and predict our own, and other people's behavior. The mental states attributed to others are unobservable, but knowable by intuition or insight. Theory-theory relates to folk psychology, which refers to the way humans *think* that they reason (Bratman, 1987). Namely, humans use concepts such as beliefs, goals, and intentions to understand and explain their own and others' behavior.

Simulation theory (e.g., Goldman, 1992; Gordon, 1996) was proposed as an alternative to theory-theory. According to simulation theorists, theory of mind is the ability to project ourselves into another person's perspective, i.e., attribute beliefs and goals to the other, and simulate his or her mental activity with our own capacities for practical reasoning. Thus instead of a theory, theory of mind is a kind of knowledge that allows one to mimic the mental state of another person. In order to simulate another's mental processes, it is not necessary to categorize all the beliefs and desires attributed to that person as such. In other words, it is not necessary to be capable of complete introspection.

Whether human theory of mind follows the theory-theory or simulation theory approach cannot be determined by just observing human adult behavior. Therefore, philosophers became interested in theory of mind development and took different views on it (Carruthers and Smith, 1996). According to some theory theorists, acquiring a theory of mind is a matter of maturation of an innate module, which happens automatically. Others think it is instantiated through social interactions. According to simulation theorists, the ability to simulate is innately given. Children only have to learn which of their mental states to vary when simulating, in order to adopt the right perspective.

There are several proposals for a mix of theory-theory and simulation theory (e.g., Heal, 1996; Perner, 1996). Simulation theory is defended on grounds of simplicity. According to simulation-theorists, simulation is more efficient than acquiring a complete theory. For these reasons, some adherers of theory-theory admit that at least some form of simulation must take place when people reason about others, and incorporate simulation aspects into a theory-theoretic account. Though this makes theory-theory acceptable for some, others remain convinced that simulation forms the basic mechanism of theory of mind. Critics of simulation theory however argue that in order to simulate, it must be known what to simulate and for that a theory is needed. This resulted in approaches stating that others' behavior is predicted by simulation, but in addition, a body of theoretical knowledge is needed to govern these simulations.

6.3 Two ways to model agents with a theory of mind

In this section, we present a theory-theory and a simulation theory approach for modeling agents with a theory of mind. We also discuss the implementation of both approaches in existing BDI-based agent programming languages.

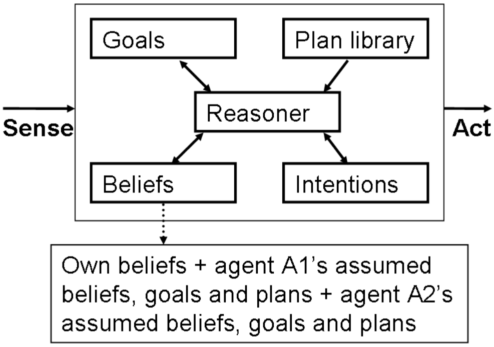

Goals
Plan library
Reasoner
Beliefs
Intentions
Sense
Act
Own beliefs + agent A1's assumed beliefs, goals and plans + agent A2's assumed beliefs, goals and plans

Figure 6.1: Architecture of a BDI agent with a theory of mind based on theory-theory.

6.3.1 A theory-theory approach

Folk psychology, in which behavior of others is understood in notions like beliefs, desires, and intentions, forms the basis of both the theory-theory account of theory of mind and BDI-based agent programming (see Section 2.2). Therefore, a BDI language is a logical choice for the implementation of an agent with a theory of mind based theory-theory.

The upper part of Figure 6.1 shows the general architecture of a BDI agent. In order to add a theory of mind ability based on theory-theory, beliefs about other agents can be added to an agent's own belief base. In Figure 6.1 this is shown by the box below the general BDI architecture. Besides its own beliefs, the agent may have beliefs with mental concepts attributed to other agents, e.g., beliefs about attributed beliefs and goals of agent A1 and A2 like in the figure. The belief *A1(B(ϕ))*, for instance, represents that the agent believes that agent A1 believes ϕ, and *A2(G(ψ))* that the agent believes that agent A2 has goal ψ. An agent's behavior is determined by its goals and beliefs. Thus, when an agent has beliefs about other agents, its behavior is also based on the believed beliefs and goals of others.

Besides beliefs about others' beliefs and goals, the agent must have a theory about how these elements interact. For instance, to predict someone's behavior, an agent needs to be able to make combinations of believed beliefs and goals, and derive new believed (sub-)goals, plans or actions. In this theory-theory-based agent model, the rules according to which the elements combine are also added as beliefs to the agent's belief base. In other words, beliefs that make combinations between beliefs about another agent's beliefs and beliefs about that agent's goals are added. Such a belief is for example *if (A1(B(ϕ)) and A1(G(ψ))) then A1(P(α))*, meaning that if the agent believes that agent A1 believes ϕ and has goal ψ, one can assume that agent A1 will execute plan α. With these beliefs, the agent is able to predict and explain other agents' behavior. To do so, the agent does not use its own practical reasoning power (the reasoner in Figure 6.1), but instead, it uses its epistemic reasoning power for making inferences of its beliefs (the epistemic reasoner is part of the agent's belief base, and is not explicitly shown in the figure).

6.3.2 A simulation theory approach

The essence of simulation theory is that an agent uses its own reasoning power to reason about other agents, and thus not all of the other's reasoning steps have to be incorporated in a theory. Figure 6.2 shows a schematic picture of a theory of mind model based on simulation theory. Like in the theory-theory model, the agent has a reasoner that deliberates over the content of its mental state. Besides a representation of its own mental state, the agent has representations of mental states attributed to other agents (dashed boxes). The agent can take its own decision-making system off-line, and start deliberating with the mental state of another agent to make predictions about its behavior. In other words, it applies its own reasoner to the attributed mental states.

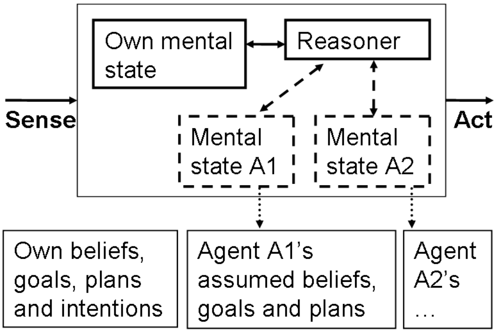

Own mental state
Reasoner
Sense
Act
Mental state A1
Mental state A2
Own beliefs, goals, plans and intentions
Agent A1's assumed beliefs, goals and plans
Agent A2's ...

Figure 6.2: Architecture of an agent with a theory of mind based on simulation theory.

Simulationists argue that in order to have a theory of mind, one does not need to have access to all reasoning rules according to which the other is reasoning. Radical simulationists even claim that the mental state of the other agent does not necessarily have to be organized in terms of beliefs and goals (Gordon, 1996). Therefore, in Figure 6.2 we did not specify how a mental state is represented. To connect to our BDI-based approach for explainable agents, however, we will assume that an agent's mental state is specified in terms of beliefs, desires, and intentions. This is shown in the boxes outside of the agent's internals.

The architecture in Figure 6.2 can best be implemented in BDI-based agent programming language that allows for modularity, e.g., Jack (Busetta et al., 2000), Jadex (Braubach et al., 2005), GOAL (Hindriks, 2007), or modular 2APL (Dastani, 2009). In all of these languages, modularization is considered as a mechanism to structure an individual agent's program in separate modules. In this context, thus, a module is the specification of a mental state: beliefs, goals, plans, and intentions. In modular languages, each mental state, the agent's own and those attributed to other agents, can be represented in a separate module. By using modules, the same practical reasoner can be used to reason with different mental states without interferences among them. If an agent wants to make a prediction about someone else's behavior, it just applies its reasoner to the assumed mental state of that agent. The agent thus reasons with another agent's mental concepts as if they are its own. The agent can use its assumptions about the other agent as input for its own reasoning process and let its actions depend on them.

6.3.3 Discussion

The two approaches just presented can both be implemented in a BDI-based agent programming language. For the TT approach, any BDI-based language can be used, and for the ST approach only those that allow for modularity. The most important difference between the TT and ST approach is the way to reason with attributed beliefs and goals. In the TT approach an epistemic reasoner is used, and in the ST approach a practical reasoner.

On the one hand, it is an advantage of the ST approach that a practical reasoner can reason with attributed beliefs, goals, and reasoning rules immediately, and the same mental state representations can be (re)used as 'normal' mental states and attributed mental states. In the TT approach, in contrast, 'normal' mental state representations have to be transformed to a different representation in order to reason with them as attributed mental states, which provides extra work for the programmer.

On the other hand, a reasoning process on an attributed mental state should have a result that the agent can use in its own reasoning process, e.g., a belief in its belief base. In the TT approach, the attributed mental states are already represented by beliefs and no extra action is needed. In the ST approach, however, an extra action is needed to update the agent's own belief based with the result of a reason process on an attributed mental state. This makes the agent program more complex.

6.4 Simulation study

In this section we describe a study which evaluates agents with (1) no theory of mind, (2) a theory of mind based on theory-theory, and (3) a theory of mind based on simulation theory. There is no common methodology for validating models representing human behavior. First, because not much attention has been paid to the validation of human behavior representation models and the field is still immature (Harmon et al., 2002; Van Doesburg, 2007). Second, there are different model types which each require their own validation (Young, 2003). Third, three different perspectives can be distinguished (Van Doesburg, 2007).

The first perspective is that of the end user. Currently, most models are evaluated by their intended use, that is, from the perspective of the end user (Chandrasekaran and Josephson, 1999). In this study we will consider the user perspective by examining whether the different agents are able to generate the behavior and explanations required for the user.

Besides the perspective of the end user, human behavior representation models can also be viewed from a psychological perspective. The psychological perspective considers how well the generation of human behavior in agents matches the generation of actual human behavior. We will not consider the psychological perspective, as the agents in virtual training systems do not have to generate behavior that is as human as possible. The agents should behave human-like, but they may e.g., make more errors than an average human if that serves a learning goal. Moreover, as discussed in Section 6.2, there is no agreement on how the human theory of mind works.

The third perspective to evaluate human behavior representation models is the developer's perspective, which concerns the effectiveness and efficiency of model creation. In the discussion, we will take the developer perspective into account. There are standard works for the assessment of software quality, e.g., the IEEE Standard 1061 (IEEE, 1998), but these are not specialized for human behavior representation models. Therefore, instead of using a standard method, we will discuss our own experiences with the implementation of the agents in the case study.

6.4.1 Methods

In order to compare agents with different theory of mind models, we used training scenarios specifying which behavior the agents should perform. For the study, we specified three variants of the training scenario described in Section 6.2: (1) an optimal, (2) a supporting and (3) a challenging version. In the optimal scenario nothing goes wrong, in the support scenario the trainee makes mistakes and the agents give support, and in the challenge scenario the agents make mistakes due to an incorrect theory of mind. Besides the agents' actions in the scenario, we also specified the corresponding explanations. The different scenarios will be described in more detail later in this section.

All three scenarios involved three agents: an A-Officer, an E-Officer, and a trainee (playing the H-Officer). Three versions of the A-Officer and E-Officer agents were implemented: agents with no theory of mind (NT), agents with a theory-theory of mind (TT), and agents with a simulation theory of mind (ST). We implemented the agents in such a way that they would generate the actions and explanations in the specified scenarios as much as possible. The implementation of the agents will also be discussed in this section.

After specifying the scenarios and implementing the agents, we run different simulation sessions with the agents. In the last part of this section we provide an overview of the different simulation runs. In the simulations, for an agent to perform well, its actions and explanations in the simulations should match the actions and explanations specified in advance.

Scenario specification

Table 6.1, 6.2, and 6.3 show the specification of the events and agents' actions and explanations in the optimal, support, and challenge scenario, respectively. In all three tables, the left column shows the actions and events of that scenario in chronological order, and the right column shows the desired explanations for actions of the A-Officer and E-Officer. A, E, and H refer to A-Officer, E-Officer, and H-Officer, mes(ne) stands for the message there is no electricity in compartment 37, mes(e) stands for the message there is electricity in compartment 37, and mes(fe) stands for the message the fire in compartment 37 is extinguished.

Actions / *events*	**Explanations**
Alarm: fire in comp 37	
E switches off elect. comp 37	then A can ext. fire with water
E reports mes(ne) to H	then H can broadcast mes(ne)
H broadcasts mes(ne)	-
A enters comp 37	to ext. the fire in comp 37
A ext. fire with water	no electricity in comp 37
Fire extinguished	
A reports mes(fe) to H	then H can broadcast mes(fe)
H broadcasts mes(fe)	-

Table 6.1: Actions, events and explanations in the optimal scenario.

In the first scenario, none of the agents made a mistake. In the second scenario, the H-Officer, that is to be played by the trainee agent, forgets to broadcast the message that the electricity has been switched off in compartment 37. The E-Officer supports the H-officer by advising him to do so. Later, the H-Officer again forgets to broadcast a message. Then the A-Officer advices him to inform the crew that the fire has been extinguished. In the third scenario, the E-Officer and the A-Officer both make one error. The E-Officer does not switch of electricity in compartment 37 by itself, the H-Officer explicitly has to order him to do so. The A-Officer forgets to update the H-Officer when the fire has been extinguished.

Actions / *events*	**Explanations**
Alarm: fire in comp 37	
E switches off elect. comp 37	then A can ext. fire with water
E reports mes(ne) to H	then H can broadcast mes(ne)
Nothing happens	
E advices H: broadcast mes(ne)	then A can ext. fire with water
H broadcasts mes(ne)	-
A enters comp 37	to ext. the fire in comp 37
A ext. fire with water	no electricity in comp 37
Fire extinguished	
A reports mes(fe) to H	then H can broadcast mes(fe)
Nothing happens	
A advices H: broadcast mes(fe)	then crew will be informed
H broadcasts mes(fe)	-

Table 6.2: Actions, events and explanations in the support scenario.

Actions / *events*	**Explanations**
Alarm: fire in comp 37	
Nothing happens	
H asks E about elect. comp 37	-
E reports mes(e) to H	H asked about elect. comp 37
H orders E: switch off elect.	-
E switches off elect. comp 37	H ordered to switch off elect.
E reports mes(ne) to H	then H can broadcast mes(ne)
H broadcasts mes(ne)	-
A enters comp 37	to ext. the fire in comp 37
A ext. fire with water	no electricity in comp 37
Fire extinguished	
Nothing happens	
H asks A about status fire	-
A reports mes(fe) to H	H asked about status fire
H broadcasts mes(fe)	-

Table 6.3: Actions, events and explanations in the challenge scenario.

Agent implementation

To develop the agents, we used our explainable agents approach as described in this thesis. For the implementation of the NT and TT agents we used 2APL, and for the implementation of the ST agents we used modular 2APL. In section 6.5, we will discuss the implementation of agents with a theory of mind ability in more detail, and propose two extensions to modular 2APL, to make the language more appropriate for developing agents with a theory of mind.

NT agents. Figure 6.3 shows a goal hierarchy of the E-Officer agent without a theory of mind. The boxes represent the agent's goals and the ovals represent its beliefs. The E-Officer's main goal is to manage the electricity in the ship, which is divided into the subgoals (1) to switch off electricity and (2) to report to the H-Officer that the electricity has been switched off. The beliefs in the hierarchy denote when the subgoals become active.

We made a similar goal hierarchy of the A-Officer with no theory of mind and implemented both NT agents 2APL. Initially, the NT E-Officer agent has the following mental state.

```
Goals:
  manageE
Reasoning rules:
  manageE | comp(X,fire) <- switchOffE(X)
  manageE | comp(X,noE) <- reportToH(noE)
```

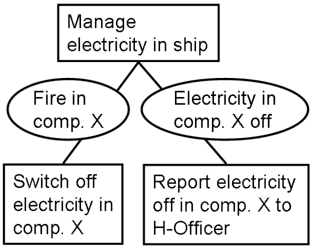

Manage electricity in ship
Fire in comp. X
Electricity in comp. X off
Switch off electricity in comp. X
Report electricity off in comp. X to H-Officer

Figure 6.3: E-Officer agent without a theory of mind.

At the start of the scenario, the agent has one goal, managing the electricity, two reasoning rules, and no plans or beliefs. Only once the agent obtains the belief that there is a fire in compartment X or that it switched off the electricity in compartment X, it will generate plans.

Though the NT agents can perform actions that have a positive effect on others' task execution, the agents' reasoning does not involve possible mental states of other agents. Information about other agents is thus implicitly present in the NT agents' mental states.

TT agents. To develop agents with a theory of mind (both TT and ST), we extended the NT agents with a theory of mind ability. Figure 6.4 shows that a theory of mind ability, theory-theory-based or simulation theory-based, delivers extra beliefs due to theory of mind reasoning. By that, the adoption conditions for subgoals change. Namely, the conditions of goals which achievement have effect on other agents' task execution, involve believed mental concepts about the others. In the example, the E-Officer only switches off electricity in a compartment if it believes that someone else intends to use water to extinguish a fire in that compartment.

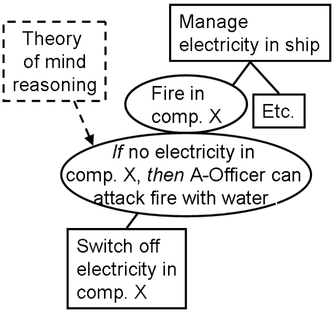

Theory of mind reasoning
Manage electricity in ship
Fire in comp. X
Etc.
If no electricity in comp. X, *then* A-Officer can attack fire with water
Switch off electricity in comp. X

Figure 6.4: E-Officer agent with a theory of mind.

The implementation of theory of mind reasoning differs for TT and ST agents. For

TT agents, we implemented their theory of mind in their belief base. The following 2APL code shows part of the E-Officer's theory of mind about the A-Officer in its belief base. Note that in 2APL, an agent's belief base is a Prolog program.

```
Beliefs:
  a_off(g,extinguishFire).

  a_off(b,comp(X,noE)).
  a_off(b,comp(X,fire)).

  a_off(p,attackFire(X,water):-
    a_off(g,extinguishFire),
    a_off(b,comp(X,noE)),
    a_off(b,comp(X,fire)).
```

The first line of code represents a belief about a goal attributed to the A-Officer, the second and third beliefs are attributed beliefs, and the fourth belief incorporates a reasoning rule telling which plan the A-Officer will probably adopt when it has these beliefs and goal. In other words, the E-Officer believes that the A-Officer will attack a fire in compartment X with water, when the electricity in that compartment has been switched off.

The E-Officer uses its theory of mind ability when predictions about the A-Officer's behavior will influence its own choices. This can be accomplished by adding extra belief checks its reasoning rules. The E-Officer's first reasoning rule then becomes as follows.

```
Reasoning rules:
  manageE | (comp(X,fire) and
    a_off(p,attackFireWithWater)) <-
    switchOffE(X)
```

The belief `a_off(p,attackFireWithWater))` is added to the belief check of the reasoning rule.

The TT agents have a first order theory of mind, which means that their theories of mind do not involve other agents' theories of mind. Thus, the agents have no beliefs like 'I believe that agent A believes that I have goal Y'. In this scenario, it was not necessary to implement agents with a second or higher order theory of mind. Though human reasoning rarely involves more than two theory of mind depth levels (Mol et al., 2005), in theory it is possible to let modules own other modules until an arbitrary depth.

ST agents. Like TT agents, ST agents are based on NT agents, but extended with a theory of mind (see Figure6.4). The difference between TT and ST agents is that the theory of mind ability of ST agents is implemented by using modules. We implemented the A-Officer and E-Officer agent based on simulation theory in modular 2APL (Dastani, 2009). A modular 2APL multi-agent program consists of one or more modules. Each module specifies a set of beliefs, goals, plans, and practical reasoning rules. When the multi-agent program is executed, one or more of these modules are identified as the specification of the initial state of individual agents. In other words, the indicated modules specify the beliefs, goals, plans, and reasoning rules of agents before their deliberation processes have started. Whereas modules specify a particular mental *state*, an agent is a deliberative *process* that continuously perceives its environment, updates its state, and reasons about its state to decide which action to perform.

During its execution, an agent can instantiate a (pre-specified) module by the action *create(mod-name,mod-ident)*. The agent owns the modules it instantiated and can perform several operations on them. Namely, the agent can update a module's goal base and belief base, query a module's goal base and belief base, execute a module, and release a module. When a module is executed, a 2APL deliberation process on the module instance is started and continues until a pre-specified stopping condition is fulfilled. It is possible that during its execution, the module instance creates new module instantiations itself, which it then owns. During the execution of a module, all processes in the owning module (the module instance that performed the execute action) are suspended.

The following modular 2APL code represents the E-Officer's plan for creating, updating and executing a module with a theory of mind of the A-Officer.

```
Plans:
  create(a_off, a_off);
  a_off.updateBB(comp(X,noE));
  a_off.execute(B(planAoff(Y)));
  Update(comp(X,noE),planAoff(Y))
```

The first action creates an instantiation of the module a_off which also has the name a_off. The second action updates the instantiation with the belief `comp(X,noE)`. Then, the module a_off is executed till the stopping condition `B(planAoff(Y))` is satisfied, i.e., the belief `planAoff(Y)` can be derived from the module's belief base. The variable Y can have different values, representing a prediction of what the A-Officer's will do. During this execution, the execution of the agent owning the module, the E-Officer, is paused. In the last line of code, the result of the execution is updated to the agents own belief base, e.g., resulting in the belief `a_off(comp(X,noE), planAoff(attackFireWithWater))`, which means that if the A-Officer believes that the electricity is switched off, he will attack the fire with water. Similar to TT agents, the E-Officer agent uses its theory of mind when the adoption of goals for switching off electricity depend on beliefs with predictions about the A-Officer's behavior.

Like the TT agents, the ST agents also have a first order theory of mind. We used the implementation of the NT agents to give the ST agents attributed mental states. The ST A-Officer's theory of mind modules contained the NT E-Officer's mental states and vice versa. Also for ST agents holds that it is possible to implement agents with second or higher order theory of mind.

Simulation runs

We ran several simulations to test whether the implemented agents were able to generate the actions and explanations specified in the three scenarios. Each agent type (NT, TT and ST) played each scenario (optimal, support and challenge) once, so in total nine (3x3) simulations were run. The A-Officer and E-Officer were always of the same type in one simulation run (both NT, both TT or both ST) because a combination of different agent

Specified behavior	**Actual behavior**		
Actions	**NT**	**TT**	**ST**
E switches off elect. comp 37	√	√	√
E reports mes(e) to H	√	√	√
A enters comp 37	√	√	√
A ext. fire with water	√	√	√
A reports mes(f) to H	√	√	√
Explanations	**NT**	**TT**	**ST**
then A can ext. fire with water	X	√	√
then H can broadcast message(e)	X	√	√
to ext. the fire in comp 37	√	√	√
no electricity in comp 37	√	√	√
then H can broadcast message(f)	X	√	√

Table 6.4: Desired and actual behavior of the NT, TT and ST agents in the optimal scenario.

types (e.g., NT and ST) would not have influenced the results. To run the challenging scenario, we adapted the implementations of the A-Officer and E-Officer agents so that they would make mistakes.

All characters in the scenarios were played by agents, and there were no humans involved in the simulations. Therefore, it was not needed to create a visualization of the agents and their environment. However, to simulate the three scenarios, besides the A-Officer and E-Officer, a trainee and an environment were needed.

We implemented two versions of the trainee agent in 2APL. A trainee agent making mistakes was used for the support scenario, and one not making mistakes was used to run the optimal and the challenge scenario. Both versions of the trainee agent consisted of some rules reasoning rules, had no theory of mind, and could not give explanations.

The role of the environment was minimized in the scenarios. The only two events in the environment in all three scenarios are a fire alarm and the extinction of the fire. Therefore, instead of implementing a separate environment, we represented the events in the belief bases of the agents. All agents believed that there was a fire in compartment 37 at the beginning of each simulation run. And the A-Officer's action to command its team to attack a fire with water led to addition of the belief *extinguishedFire* to its belief base. We thus assumed that actions could not fail.

6.4.2 Results

During each simulation run, A-Officer and E-Officer's actions and explanations were logged, and these logs were compared to the specified scenarios. For each of the three scenarios, three simulations were run (with NT, TT and ST agents). Table 6.4, 6.5, and 6.6 show the results of the optimal scenario, support scenario, and challenge scenario, respectively.

The left columns of Table 6.4, 6.5, and 6.6 show a part of the desired actions and explanations in the specified scenarios. The last three columns show whether the agents' actions and explanations did (√) or did not (X) match the specified ones. The tables only show actions and explanations of the A-Officer and E-Officer, as those were the agents to be evaluated. Events, and actions of the H-Officer are not displayed.

The results show that all of the agents' actions matched the specified ones. In all nine simulations we found that the agents' actions in the simulation matched the specifications for 100%. Thus, independent of whether the agents had a theory of mind and which theory of mind model, they were all able to display the specified actions, including support actions and making mistakes due to an incorrect theory of mind.

The results in also show that the explanations of the agents with a theory of mind, the TT and ST agents, matched all of the specified explanations. The agents were able to incorporate beliefs and goals of others in their explanations. The explanations of the agents without a theory of mind, the NT agents, did not always match the specified ones. The NT agents only gave explanations in terms of their own beliefs and goals. For some actions these explanations matched the specified ones (e.g., the third and fourth explanation in Table 6.4), but they did not when the actions had consequences for other agents (e.g., the first two explanations in Table 6.4). Thus, agents with a theory of mind were able to explain the consequences of their actions for other agents, also for support actions and mistakes, and agents without a theory of mind were not.

Specified behavior	**Actual behavior**		
Actions	**NT**	**TT**	**ST**
E switches off elect. comp 37	√	√	√
E reports mes(e) to H	√	√	√
E advices H: broadcast mes(ne)	√	√	√
A enters comp 37	√	√	√
A ext. fire with water	√	√	√
A reports mes(f) to H	√	√	√
A advices H: broadcast mes(ne)	√	√	√
Explanations	**NT**	**TT**	**ST**
then A can ext. fire with water	X	√	√
then H can broadcast message(e)	X	√	√
then A can ext. fire with water	X	√	√
to ext. the fire in comp 37	√	√	√
no electricity in comp 37	√	√	√
then H can broadcast message(f)	X	√	√
then crew will be informed	X	√	√

Table 6.5: Desired and actual behavior of the NT, TT and ST agents in the support scenario.

Specified behavior	**Actual behavior**		
Actions	**NT**	**TT**	**ST**
E reports mes(e) to H	√	√	√
E switches off elect. comp 37	√	√	√
E reports mes(e) to H	√	√	√
A enters comp 37	√	√	√
A ext. fire with water	√	√	√
A reports mes(f) to H	√	√	√
Explanations	**NT**	**TT**	**ST**
H asked about elect. comp. 37	√	√	√
then A can ext. fire with water	X	√	√
then H can broadcast message(e)	X	√	√
to ext. the fire in comp 37	√	√	√
no electricity in comp 37	√	√	√
then H can broadcast message(f)	X	√	√

Table 6.6: Desired and actual behavior of the NT, TT and ST agents in the challenge scenario.

6.4.3 Discussion

The results show that agents with a theory of mind (TT and ST) have advantages over agents without a theory of mind (NT). Though all three agent types generated equal behavior, the agents with a theory of mind were able to give explanations involving other agents' assumed mental states, and the agents without a theory of mind were not. Concerning observable agent behavior (including explanations), there was no difference between the theory-based and the simulation-based approach, and there are no reasons to assume that the outcome would be different for other scenarios. Thus, from a user perspective, there is not preference of one approach over the other.

For a developer, however, there are differences between TT and ST agents. A first observation concerns the reuse of code. When implementing the theory of mind of a TT agent, we had to translate the BDI representation of its mental state to a Prolog representation, and practical reasoning rules to epistemic reasoning rules. Namely, a TT agent's theory of mind is about a BDI agent, but represented only by beliefs. For the implementation of an ST agent, we did not have to make such a translation. Instead, we could reuse the code of one agent to implement the theory of mind of another. Though the extra work of implementing TT agents compared to ST agents was not much in our case study, the advantage of reuse of code increases with more complex agent models. As a first finding, it may thus be concluded that concerning the reuse of code, the ST approach is preferred over the TT approach.

A second finding involves the introduction of errors related to theory of mind use into the agent models. The introduction of single errors was comparably easy to implement

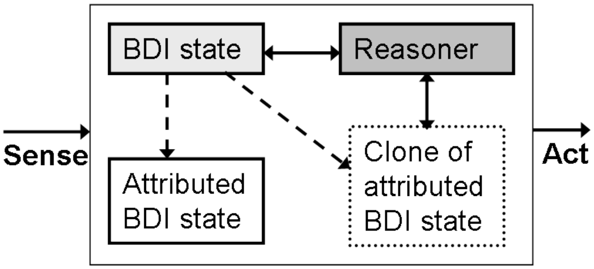


Figure 6.5: The improved architecture of a BDI agent with a theory of mind based on simulation theory.

in both agent models. However, in the TT approach we could only include errors individually, whereas in the ST approach it was possible to introduce some structural errors. A structural error is for example that an agent does not take its theory of mind about another agent into account at all, or that an agent bases its behavior on a theory of mind of the wrong agent. Also on this point, the ST approach is preferred over the TT approach.

We implemented the ST agents for the simulation study in modular 2APL. In the next section we will discuss two problems we encountered during the implementation of the ST agents. And as none of the existing modular BDI-based agent programming languages offers a solution for these problems, we also introduce two extensions to modular 2APL that overcome these issues.

6.5 Extending Modular 2APL

The simulation study in the previous section showed that an approach based on simulation theory is preferred over an approach based on theory-theory. We argued that agents with a theory of mind based on simulation theory can best be implemented in a BDI-based agent programming language that allows for modularity. Namely, modularity makes it possible to represent the agent's own BDI state and each BDI state attributed to another agent in a separate module, and the same reasoner can be applied to each module.

We developed the ST agents in the simulation study according to the model shown in Figure 6.2, and implemented them in modular 2APL as described in Subsection 6.4.1. A disadvantage of this approach is that when an agent uses its theory of mind ability, it makes updates to and reasons directly with the attributed mental state of another agent. It could be, however, that the changes made to the attributed mental state in this process should be made undone. For instance, when the agents wants to predict what the other agent will do by adding belief a (and not b) and by adding belief b (and not a). A way to achieve this is by letting the agent reason with a copy of the attributed mental state, as shown in Figure 6.5. After the agent has reasoned with the cloned attributed mental state, it can decide whether it wants to change to the actual attributed mental state.

A second disadvantage of the current version of modular 2APL is that it is not possible to test a plan base, whereas to predict another's agents actions, one would like to check its plan base. It is possible to ensure that when a plan is adopted in an attributed mental state, also an update is made to the belief base (it is possible to check a module's belief base in modular 2APL). However, this leaves a lot of responsibility to the programmer to make sure that the right updates to belief bases are made and no actual actions are executed in the environment. An easier way is if the agent could directly check a module's plan base.

Module cloning and plan testing was not possible in modular 2APL as originally proposed (Dastani, 2009), and neither in any of the other modular BDI-based programming languages, e.g., Jack (Busetta et al., 2000), Jadex (Braubach et al., 2005), GOAL (Hindriks, 2007). We therefore extended modular 2APL with two functionalities: module cloning and plan testing.

6.5.1 Module cloning and plan testing

To extend modular 2APL with module cloning, we added the action *clone(mod-name,mod-ident)*, which is quite similar to the action *create(mod-name,mod-ident)*. Like the create action, the clone action creates a new module instantiation with the name mod-ident. The difference between both actions is that the create action can instantiate pre-specified modules, and the clone action can create an instantiation of an already instantiated module at any time. A cloned module contains the mental state of the instantiated module (with the name mod-name) at the time the clone action was executed. This mental state may differ from mod-name's initial mental state, i.e., its mental state at the time it was instantiated, due to actions performed on mod-name. The clone action can only be performed by an agent or module instance (created by the agent) which already owns an instantiated module (mod-name). The new module instantiation is owned by the agent or module that created it. The owning module can perform the same actions on a module that was instantiated by a clone action as on a module that was instantiated by a create action (updating, querying, executing, etc.).

When a cloned module is executed, similar to created modules, all processes in the owning module are suspended. Theory of mind reasoning is thus not more expensive in terms of computation than running a 2APL agent without modules. The deliberation process returns to the owning module when a given stopping condition is reached, and if not the stopping condition is not reached, the created or cloned module remains in control. It is the programmer's responsibility to specify these stopping conditions correctly so that control goes back to the owning module when desired. When a module is used to reason with an attributed mental state, no actions of the cloned module should be executed. Also here, it is the programmer's responsibility to choose the stopping condition such that no actions can be executed before it is reached. This is made possible by plan testing.

To summarize, the clone action makes it possible to reason with an attributed mental state and query it afterwards, while saving the mental state prior to the reasoning. The use of cloning related to theory of mind will be illustrated with an example in the next section.

The second function we added to modular 2APL is plan testing. Plan testing can be used to specify the stopping condition of an execute action. In the original version of modular 2APL the stopping condition φ in the execute action *m.execute(φ)* can only be a goal test or a belief test. With the addition of plan testing, φ can also be a plan or an action. That makes it possible to specify stopping conditions that ensure that the control goes back to the owning module before any action is executed.

Plan testing also makes it possible to query a module's plan base. In the original version of 2APL, the owner of a module instance m can access the belief base and goal base of that module through the actions *m.B(φ)* and *m.G(φ)*, respectively. In the extended version of modular 2APL, the owner of a module instance m can also access the plan base of a module through the action *m.P(α)*. This plan test succeeds if there is at least one plan of which the first action matches the specified one (φ). With a plan query action, the owner of a module can test which action the module would execute in a following deliberation step. In the next section we will show how this enables an agent to predict the behavior of other agents.

6.5.2 Illustration

We illustrate the extended modular 2APL by showing part of the implementation of the E-Officer from the on-board firefighting training scenario in Section . The agent is implemented in the extended version of modular 2APL, including module cloning and plan testing. The E-Officer is located at the Technical Center of the navy ship, and from there, manages the electricity in the whole ship. Initially, the E-Officer attributes the following mental state to the A-Officer, which is the leader of the attack team.

```
Goals:
  lead(attack_team)

PG-rules:
  lead(attack_team) <- fire(X) and electricity(X,off) |
    send(team,instruct,l,o,instruct(extinguish(fire)))
```

The code below `Goals` represents that the A-Officer's goal is to lead the attack team. The code below `PG-rules` means that if the commander has the goal to lead the attack team, and the beliefs that there is a fire in compartment X and the electricity in that compartment is switched off, the A-Officer will instruct his team to extinguish the fire. A 2APL send-action has five parameters: receiver, speech act, language, ontology, and content. In this example only the first and last parameter are of importance.

The part of the E-Officer's own mental state that is relevant in this example is the following.

```
BeliefUpdates:
  { true } Update(X,Y) { prediction(X,Y) }

Beliefs:
  fire(37).

Goals:
  manage(electricity)
```

```
PG-rules:
  manage(electricity) <- fire(X) |
    {
   a_officer.updateBB(fire(X));
    clone(a_officer,a_officer_clone);
    a_officer_clone.updateBB(electricity(X,off));
    a_officer_clone.execute(P(sendaction(_,_,_,_,_)));
    a_officer_clone.P(sendaction(_,_,_,_,Message));
    Update(a_officer,Message);
    release(a_officer_clone)
   }
```

The E-Officer has the belief fire(37), i.e., there is a fire in ship compartment 37, and the goal manage(electricity). Consequently, the reasoning rule with the belief fire(X) and goal manage(electricity) in its head and guard will apply (the X is substituted by 37), and the plan in the body of this rule will be executed. The plan consists of the following actions.

- Update the mental state attribution of the A-Officer with the belief fire(37) because the agent assumes that the A-Officer also believes that there is a fire, e.g., because there was a public announcement.

- Clone the attributed mental state of the A-Officer to predict what the A-Officer will do when the electricity in compartment 37 is switched off.

- Update the cloned module with the belief electricity(37,off).

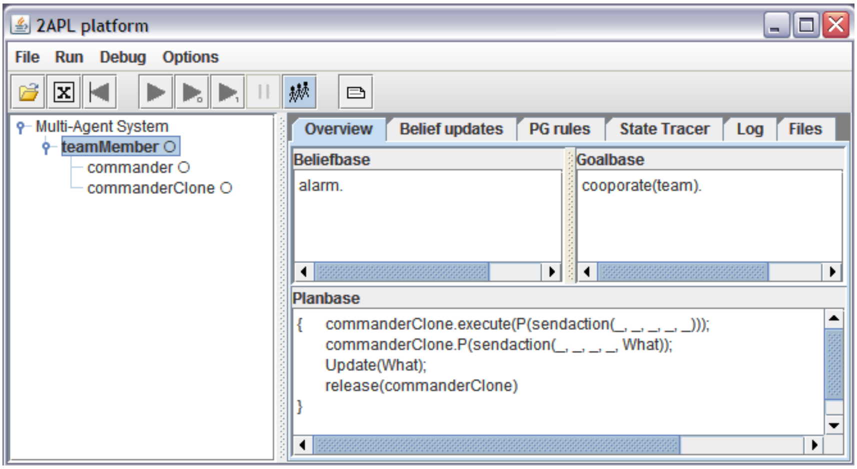
- Execute the cloned module until it fulfills the stopping condition P(sendaction(_,_,_,_,_)), meaning that it has a communication action in its plan base.

  – When the clone is executed, a deliberation process on the attributed mental state of the A-Officer starts. As the beliefs fire(37) and electricity(37,off) have been updated to the mental state, the PG-rule in the cloned module will apply and the action to instruct the team members to extinguish(fire) will be added to the plan base. Once this has happened, the stopping condition P(sendaction(_,_,_,_,_)) is fulfilled and the control goes back to the agent owning the cloned module, the E-Officer.

- Query the attributed mental state clone about the first action in the plan base (sendaction(_,_,_,_,Message)).

- Update the content of the send action (Message) to the E-Officer's belief base. In this case, the action will result in the addition of the belief prediction(a_officer, extinguish(fire)).

- Release the cloned module.



```
alarm.
```

```
cooporate(team).
```

```
{     commanderClone.execute(P(sendaction(_, _, _, _, _)));
      commanderClone.P(sendaction(_, _, _, _, What));
      Update(What);
      release(commanderClone)
}
```

Figure 6.6: Screenshot of the 2APL development environment.

In order to obtain social behavior, the E-Officer's actions should depend on whether it has the belief prediction(a_officer,extinguish(fire)) or not. For instance, the E-Officer decides to switch off electricity in compartment 37 so that the A-Officer's team will start extinguishing the fire. Social explanations can then easily be generated; the E-Officer's action, switching off electricity, can be explained by its belief that the A-Officer's team will then start to extinguish the fire.

Figure 6.6 shows a screenshot of the 2APL development environment during the execution of the E-Officer agent. The left panel visualizes the agent's modules: the mental state attributed to the A-Officer (commander) and a clone of it (commanderClone). The panels at the right side of the figure show the E-Officer's own beliefs, goals, and plans in its Beliefbase, Goalbase, and Planbase, respectively.

6.6 Related work

An agent with a theory of mind has the ability to form models about other agents' mental states. These models can in turn contain models of other agents, which can contain models of other agents, etc. This is a form of recursive modeling, which was first introduced by Gmytrasiewicz and Durfee (1995).

The theory of mind ability discussed in this chapter focuses on forming and using attributions of beliefs, goals and plans of single agents. There are several executable theory of mind models which capture other aspects. For instance, Scasselati (2002) described an account of theory of mind for robots which focuses on gaze behavior, Peters (2005) introduced a theory of mind approach for conversation initiation in virtual environments, and Boella and Van der Torre (2004) presented an approach involving the attribution of mental attitudes to groups instead of single agents.

Sindlar et al. (2011) proposed a nonmonotonic reasoning mechanism that can be incorporated into BDI agents, allowing them to reason about observed behavior to infer others beliefs or goals. They show how the behavior-generating rules of agents can be translated into a nonmonotonic logic programming framework, and provide a formal analysis of their approach.

Bosse et al. (2011) introduced a formal BDI-based agent model for theory of mind, and showed its use in modeling social manipulation, animal cognition and virtual character behavior. Like the approaches presented in this chapter, Bosse et al. represent both the mental state of the attributing agent and its mental state attributions in terms of beliefs, desires and intentions. In Bosse et al.'s approach, agents reason *about* attributed mental concepts (in contrast to reasoning *with* attributed mental concepts as if it are one's own). This corresponds to the theory-theory approach described in this chapter. Agents in Bosse et al.'s approach do not use their own reasoning power for reasoning with attributed mental concepts, like the agents based on simulation theory as presented in this chapter.

PsychSim is a simulation tool for modeling interaction between agents with a theory of mind (Pynadath and Marsella, 2005). PsychSim agents have a decision-theoretic world model, including beliefs about their environment and recursive models of other agents. Where the models presented in this chapter are based on a BDI model, the PsychSim agents are based on quantitative models of uncertainty and preferences. The PsychSim approach thus involves less symbolic representations of agents' mental states and is therefore less appropriate for explanation purposes than the BDI-based approaches presented here.

Laird (2001) introduced an account of theory of mind with the goal to add anticipation to a Quakebot. In Laird's approach, an agent creates an internal representation of what it thinks the enemy's internal state is, based on its observation of the enemy. The agent then predicts the enemy's behavior by using its own knowledge of tactics to select what it would do if it were the enemy. As in the simulation theoretic approach presented in this chapter, the agent reasons as if it were the other. The Quakebot in Laird's approach is implemented in Soar, which is not BDI-based. Again, BDI agents like presented in this chapter are more appropriate for the generation of folk psychological explanations than Soar agents.

Aylett and Louchart (2008) presented an account of intelligent agents with theory of mind which is based on simulation theory. The agents are implemented in the emotionally driven agent architecture FAtiMA, in which agents assess the emotional impact of events in the world around them when deciding on their own actions. The agent's own mind is used to simulate what other agents might feel as a result of a possible action, and based on that the agent determines its own actions. The difference between this approach and the simulation-based approach presented in this chapter is that the first focuses more on the emotional impact of behavior, and the latter focuses mostly on intentional behavior.

6.7 Chapter summary

In this chapter, we introduced two approaches for modeling agents with a theory of mind, based on the theory-theory and the simulation theory of mind. We performed a simulation study to compare agents with no theory of mind (NT), a theory-theory of mind (TT) and a simulation theory of mind (ST) in an actual training scenario. It was found that all agent types were able to display the specified behavior, but only the agents with a theory of mind were able to provide explanations in which others' mental states were involved. From the perspective of the end user, there is no difference between the two theory of mind approaches, but from a developer's perspective, the simulation theory has several advantages over the theory-theory approach. Namely, the simulation theoretic approach makes it easier to introduce realistic errors in agent behavior due to an impaired theory of mind, and it promotes the reuse of code. Finally, we introduced two extensions to modular 2APL which makes the language more appropriate for implementing agents with a theory of mind based on simulation theory.

Chapter 7

Discussion

In this thesis we proposed a new approach to explaining agent behavior in virtual training. Our approach differs from other accounts to explaining agent behavior by connecting the generation and explanation of behavior (Section 1.1). Most existing approaches propose explanation components that are independent from specific agent behavior representations and virtual training systems. Such explanations components need to be provided with new behavior representations each time they are used in a different application. In our approach, in contrast, virtual agents and their explanation capabilities are co-developed. Namely, agent behavior is represented in such a way that the behavior representations can be used for both the generation and the explanation of behavior, i.e., an explanation-friendly behavior representation. Such explanation-friendly behavior representations are an advantage, since they save developers from representing agent behavior twice.

To develop our explainable agent approach, we performed a literature study (Chapter 2), worked out a BDI-based model for explainable agents (Chapter 3), and performed evaluation studies to determine explanation algorithms (Chapter 4). Despite this rigorous approach, we have not been able to demonstrate a positive effect of the agents' explanations on trainees' learning (see Sections 5.1 and 5.2). However, such effects have neither been reported for other approaches explaining agent behavior in virtual training, i.e., approaches in which the generation and explanation of behavior are not connected.

The goal of our research was to develop agents with explanation capabilities in a more efficient way than previous approaches, which produce explanations that are at least as good as those in other approaches. In this chapter, we will discuss the advantages and disadvantages of our approach regarding the development process of explainable agents. Subsequently, we discuss four application domains of explainable agents, of which two have not been mentioned before in this thesis. We discuss under which circumstances the applications benefit most from the advantages of our approach. Finally, we provide a broader perspective on explanations of agent behavior, aiming to support the development of explainable agents.

7.1 Advantages and disadvantages of our approach

An advantage of our approach is that the behavior of explainable agents is represented in an efficient way. The distinguishing feature of our approach for developing explainable agents is that the generation and explanation of behavior are connected. In the introduction, we argued that if agents are modeled in an explanation-friendly way, an agent's behavior representation can also be used to explain its behavior. The reuse of behavior representations is more efficient than building separate behavior representations both in the training system and the explanation component, as is done in most other approaches.

A possible disadvantage of our approach is that the information needed for the generation of behavior may not match with the information needed for the explanation of behavior. Namely, our approach requires representations of agent behavior that contain sufficient information for both the generation and explanation of behavior. In the agents we modeled for our studies, the elements needed for the generation of behavior not always matched elements needed for the explanation of behavior. On the one hand, some of the elements in a goal hierarchy needed for the generation of agent behavior were not useful as explanations. On the other hand, the information that was useful in explanations was not always present in the goal hierarchies. We will illustrate both cases with an example, and also suggest a way to overcome this disadvantage for both cases.

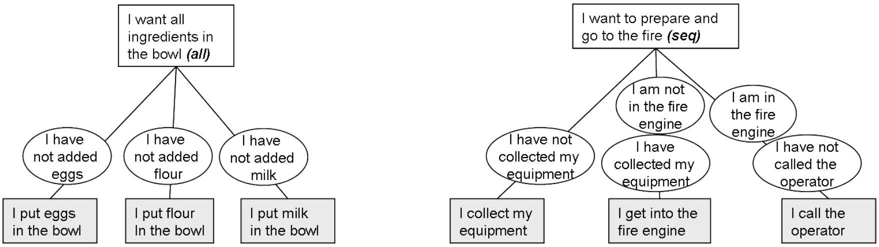

I want all ingredients in the bowl *(all)*
I have not added eggs
I have not added flour
I have not added milk
I put eggs in the bowl
I put flour In the bowl
I put milk in the bowl
I want to prepare and go to the fire *(seq)*
I am not in the fire engine
I am in the fire engine
I have not collected my equipment
I have collected my equipment
I have not called the operator
I collect my equipment
I get into the fire engine
I call the operator

Figure 7.1: Information needed for generation, but not for explanation of behavior.

Figure 7.1 shows two examples of goal hierarchies with elements that are needed for the generation of behavior, but that are not useful in an explanation. The explanation algorithm B+1, explaining an action by the beliefs directly above it, usually generates useful explanations (Chapter 4). However, in the parts of goal hierarchies depicted in the figure, the beliefs above the actions do not contain useful information for a trainee. The left part of the figure shows a goal with an *all* relation to its actions, meaning that all of the actions need to be performed in order to achieve the goal. For the generation of behavior it is important that there are beliefs that show which actions have been performed to avoid that the agent performs the same action over and over again. Applying the common explanation algorithm B+1 to this goal hierarchy, however, generates uninformative explanations like 'I put eggs in the bowl because I have not added eggs'. Similarly, the right part of the figure shows an example with a goal of type *seq*. The beliefs in this goal hierarchy are used to ensure that the actions are executed in the right order, but algorithm

B+1 generates explanations like ‘I collect my equipment because I have not collected my equipment’.

A way to overcome this disadvantage is to select proper explanation algorithms. In both examples, the beliefs above the actions are needed to let the agent perform the right behavior. Therefore, these beliefs should not be removed from the goal hierarchy. Instead, explanation algorithms should be used that select other elements that yield more useful explanations.

Figure 7.2 shows an example of the opposite situation, that is, elements that are useful in an explanation, but that are not needed for the generation of agent behavior. In the goal hierarchy at the left side, information about the mental states of other agents is only implicitly present (Chapter 6, Figure 6.3). The agent can act as if it has a theory of mind, but it cannot explain its behavior in terms of attributed mental concepts. An explanation of this agent would for example be: ‘I switched off electricity in compartment 37 because there is a fire in compartment 37’.

A way to overcome this disadvantage is to add extra information to the goal hierarchy. The right side of the figure shows a goal hierarchy of an agent with a theory of mind ability (Chapter 6, Figure 6.4). This agent is able to reason with mental concepts attributed to other agents, shown by the belief added to its goal hierarchy. This agent cannot only act, but also explain its actions in terms of attributed mental concepts, e.g., ‘I switched off electricity in compartment 37 because there is a fire in compartment 37, and if there is no electricity in compartment 37, the A-Officer can attack the fire with water’. This example shows that if not all information needed in explanations is present in an agent’s goal hierarchy, extra information can be added to it.

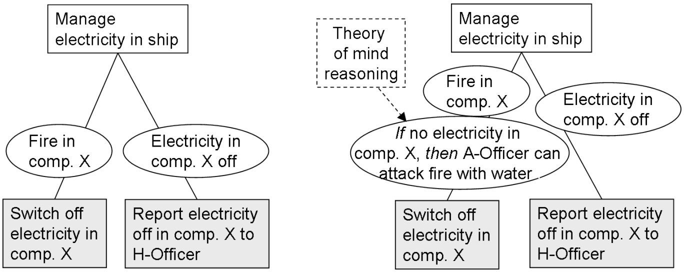

Manage electricity in ship
Fire in comp. X
Electricity in comp. X off
Switch off electricity in comp. X
Report electricity off in comp. X to H-Officer
Theory of mind reasoning
Manage electricity in ship
Fire in comp. X
Electricity in comp. X off
If no electricity in comp. X, *then* A-Officer can attack fire with water
Switch off electricity in comp. X
Report electricity off in comp. X to H-Officer

Figure 7.2: Information needed for explanation, but not for generation of behavior.

To summarize, the advantage of our approach is that developers do not have to be represent agent behavior twice. This saves work. The disadvantage of our approach is that developers may have to overcome problems due to a mismatch between the information needed for explanation and for the generation of behavior. This requires extra work.

In conclusion, there is a trade-off whether the additional effort of developing a behavior model that meets the demands for generating useful explanations is worth the benefits. When it is well possible to represent an agent’s behavior in an explanation-friendly way, our approach saves the endeavor of representing all explanations separately. However, when it requires a great deal of effort to make a behavior representation model explanation-friendly, it might as well be a good option to represent the explanations separately and freely choose an agent's behavior representation model. Whether the benefits of our approach outweigh its costs will depend on the requirements of the application in which explainable agents are needed. In the next section, we discuss under which circumstances and for which application domains our approach is particularly useful.

7.2 Application domains of explainable agents

We have developed our approach for explainable agents in the context of virtual training. In addition, we explored whether explainable agents can enhance coordination in human-agent teams (Section 5.3). In the present section, we will discuss under what circumstances our approach is useful in these two application domains. We also introduce two new application domains in which our approach to developing explainable agents can be useful.

Virtual training. Our approach is especially useful when explainable agents are used for more than one training scenario. When the virtual characters' behavior is not represented by intelligent agents, a complete scenario script needs to be specified in order to create a new scenario, including the behavior of the virtual characters in each possible situation. However, when the virtual characters' behavior is represented by intelligent agents, a new scenario can be created by changing the initial circumstances or the events in the scenario. The intelligent agents should be able to deal with these new circumstances, and a new scenario will unfold. Moreover, when these agents are equipped with explanation capabilities according to our approach, they are able to explain their behavior in these new scenarios without further effort. In other approaches to explainable agents, this would require making new behavior representations in the explanation component.

We provide two examples of when it is useful to use agents in multiple scenarios. First, agents can be reused when the same task is being trained by different training scenarios. Second, agents can be used for training of different tasks in the same domain. For instance, agents playing the role of firefighters may be needed in a training scenario for a leading firefighter, but also in a training scenario for a police officer. In both examples, explainable agents developed according to our approach keep their explanation capabilities in different scenarios. When explainable agents are used in more than one scenario, the extra initial investments in agent development can become worth the benefits later.

Human-agent teams. Also in human-agent teams, our explainable agents approach is most useful when the agents are used in multiple situations. For example, agents can be part of teams with different constellations (humans and agents) and have different roles or tasks in a team. When the agents are modeled according to our approach, they keep their explanation capabilities in these different situations.

Our approach to explainable agents is most beneficial when agents are used in multiple situations. This matches well with the observation that explanations of agent behavior in human-agent teams are especially useful when the human does not know (yet) how the agent(s) in the team behave(s). Explanations help the human to learn the reasons behind the agent's actions. When a human interacts with an agent for some time without receiving explanations, he will learn the rules according to which the agent behaves and start to understand the reasons for its behavior after some time. Once the human has already learned about the agent's behavior, there is less or no need to explain the agent's behavior. Thus, when someone has not had the opportunity yet to learn about an agent's behavior, explanations contribute to increased understanding of the motivations behind the agent's actions. With such understanding, it will be easier to coordinate one's own actions to that of the agent. Therefore, our approach to explainable agents is particularly useful in human-agent teams where explanations are most important.

Debugging. Explanations about agent behavior may be useful for the debugging of agent programs. Debugging agents concerns detecting and solving errors (bugs) in agent programs. Bugs can be divided into syntactic and semantic bugs. When an agent program has a syntactic bug, e.g., a typo, it cannot be compiled and the location of the bug can easily be detected. Semantic errors are usually harder to detect than syntactic errors because they do not hinder the compilation of the agent program. However, when the agent program is run, the agent displays undesired behavior and the cause for this behavior is sometimes hard to find. Our approach to explainable agents can contribute to the detection of semantic bugs in BDI agent programs by providing explanations for why the agent shows certain behavior.

Social simulation. Our approach to explainable agents may be also valuable in agent-based social simulations (Harbers et al., 2010c). Social simulations provide the opportunity to investigate the relation between the behavior of individuals and emerging social phenomena like crowd behavior, cooperation, and reputation. To understand the social phenomena that arise in a simulation, not only the macro processes should be studied, but also the behavior of the single agents. For instance, a crowd can start to move because all agents are running towards a particular place or because they are all following one leader. A second example is cooperation, which may emerge because agents behave in an altruistic or in an self-interested way. Explainable agents can help to obtain such insights into individual agents' behavior in social simulations.

Social simulations are used to investigate social interactions, and often, it is not known beforehand how the agents in a social simulation will behave. Therefore, it is difficult to build an explanation component which has no access to the internal state of an agent. Preferably, explanations about behavior are directly derived from the agents' behavior representations (for an example about animal behavior, see, e.g., Van der Vaart and Verbrugge, 2008).

Some social patterns can be modeled adequately by a few simple if-then rules only, e.g., the flocking behavior of birds (Reynolds, 1987). The behavior of such agents can be explained by their rules and interaction with the environment. The modeling of other social phenomena requires more complex agents that, besides merely reactive behavior, also display proactive behavior. The behavior of such agents is more variable, and harder to predict and understand. In particular in simulations with proactive agents, the similarities or contradictions in the explanations of different agents can help to understand the overall processes. To achieve increased understanding of social phenomena in multi-agent based simulations, the explanations of individual agents need to be aggregated into one, more global explanation. That, however, is beyond the scope of this thesis.

7.3 A more general view on explaining agent behavior

In the first section of this chapter we showed that an agent's behavior representation does not always contain the information that is needed for the explanation of its behavior. Therefore, in our approach the purpose of explaining an agent's behavior needs to be taken into account in its design. Though it is impossible to represent all explaining factors of an agent's actions in its behavior representation model, it is possible to choose a behavior representation in such a fashion that most elements in the representation can be used for explanation as well. In this section, we take a more general perspective on explaining agent behavior based on the literature study in Chapter 2 and our own experiences. We will propose five different ways to explain an agent's action, and we will also distinguish different contexts of explanation. Compared to Malle's framework (Malle, 1999) and Atkinson's argumentation scheme (Atkinson et al., 2006) (Chapter 2) we distinguish more different explanation types. This broader perspective aims to support the design process of explainable agents by making developers aware of different possible explanations for an action.

7.3.1 Five questions

To explain an agent's action in various ways, we introduce five questions that each address a different aspect of the action. The general question asking for an action's explanation, *Why did you perform this action?*, thus contains the following five subquestions.

- *What goal did you try to achieve?*
- *Why do you want to achieve that goal?*
- *Why does this action achieve that goal?*
- *Why did you perform the action at this particular time point?*
- *Why did you perform this action and not another?*

The first question considers the goal behind an action, or in other words, it refers to the desired effects of the action. An explanation is, for instance, 'I called a friend to wish him a happy birthday'. Both Malle's and Atkinson's frameworks distinguish goals as an explanation or argumentation type.

The second question, 'why do you want to achieve this goal?', concerns the reasons behind a goal. For instance, 'I called my friend because I know that he appreciates phone calls for his birthday'. In Malle's framework such explanations are called causal history explanations, and they are similar to values in Atkinson's scheme. In a goal hierarchy, a goal above a goal provides a reason for a goal.

The third question, ‘why does this action achieve that goal?’, can be answered by domain knowledge such as terminology or the function of a tool that is used in the action. The domain knowledge required in our example is rather common, but an explanation of this type would be: ‘I called my friend because calling someone allows one to talk to that person’. This category is not distinguished in the frameworks by Malle and Atkinson.

The fourth question concerns the timing of an action. Possible answers to this question are the event that triggered the action, or the events that made it possible to perform the action. In our example such an explanation could be: ‘I called my friend because today is his birthday’. The timing of an action can be categorized as a reason such as distinguished in Malle’s framework.

The fifth question asks why this particular action was performed and not another. The answer may concern multiple possibilities, e.g., ‘I called my friend because I did not have the time to visit him’, or preferences, e.g., ‘I called my friend because I believe that calling is more personal than sending an email’. Explanations referring to the capabilities of the actor are similar to the enabling factors in Malle’s framework. Malle has no separate category for preferences. Explanations with preferences are more similar to values in Atkinson’s scheme.

7.3.2 Contexts of explanation

We distinguished five questions, but often there are multiple possible answers to these questions. For instance, I leave a note at your desk because I want you to find it, but also because I want to remind you of something. Both explanations in the example contain a goal. To account for different possible explanations of the same type we introduce the notion of an explanation context. Examples of explanation contexts are the following.

- *Physical context*
- *Psychological context*
- *Social context*
- *Organizational context*
- *Etc...*

A physical context of explanation refers to explanations in terms of positions of agents and objects, and physical events in the environment. A psychological context refers to characteristics of the agent such as personality traits, emotions, preferences, and values. A social context refers to aspects such as mental states attributed to other agents, and trust in others. An organizational context refers to an agent’s role, its tasks, its power relation to others, procedures, rules, and norms. The two explanations for putting a note at your desk, ‘so you will find it’ and ‘to remind you of something’, are given in a physical and social context, respectively.

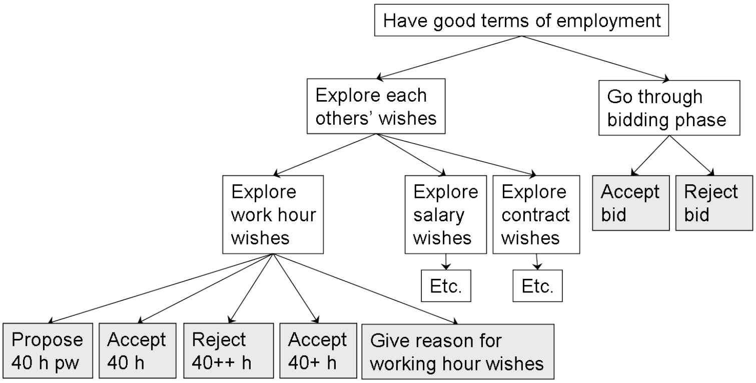


Figure 7.3: The negotiation agent modeled in a procedural context.

7.3.3 Illustration

In this section we will illustrate the presented view with an example of an explainable agent for virtual negotiations training (Section 5.2). The agent has the role of a candidate employee in a negotiation about terms of employment. In the training scenario, the candidate employee and the employer (played by the trainee) explore each other's preferences regarding working hours, contract type, and salary, and then the employer makes a bid. The agent's behavior is represented twice, by two different goal hierarchies. The goal hierarchies represent equal behavior, but are modeled in different explanation contexts and thus yield different explanations. The example clearly shows the relation between behavior representation and explanation.

Figure 7.3 shows the first version of the goal hierarchy of the candidate employee agent. Note that only the agent's goals and actions (in gray) are displayed, and not its beliefs. The actions in the hierarchy can be explained by their underlying goals. For instance, the action to propose 40 hours per week is explained by the goal that you want to explore each other's wishes on working hours because you want to explore all wishes. The acceptance of a bid is explained by the goal that you want to go through the bidding phase. Though the goal hierarchy properly and clearly represents the agent's behavior, these explanations seem quite useless for training. When viewed in the perspective of different explanation contexts, this goal hierarchy can be seen as modeled in a procedural context. To generate useful explanation, however, a more psychological perspective is needed.

Figure 7.4 shows another version of the goal hierarchy that takes the agent's personal preferences and goals into account. The actions (gray) in this model are the same as in the first one. But though both models generate exactly the same behavior, the explanations of the actions are different. In the new model, for instance, the action to propose 40 hours per week is explained by the goal that you want to work with a maximum of 40 hours per week because you want to have enough time to prepare your trip. And the action of

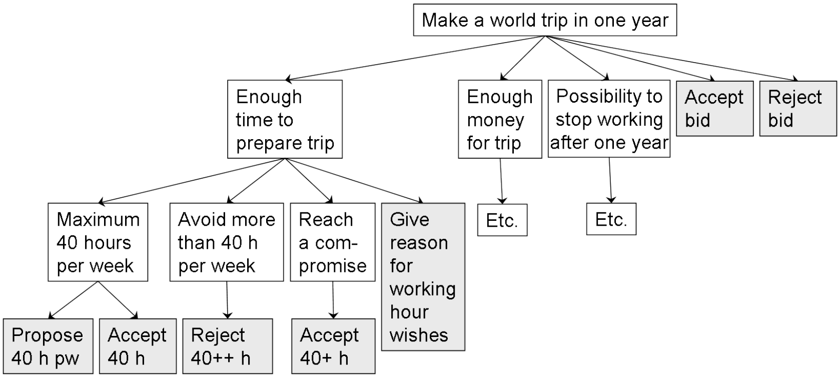

Make a world trip in one year
Enough time to prepare trip
Enough money for trip
Possibility to stop working after one year
Accept bid
Reject bid
Maximum 40 hours per week
Avoid more than 40 h per week
Reach a com- promise
Give reason for working hour wishes
Etc.
Etc.
Propose 40 h pw
Accept 40 h
Reject 40++ h
Accept 40+ h

Figure 7.4: The negotiation agent modeled in a psychological context.

accepting a good bid is explained by the goal that you want to make a world trip in a year. These explanations seem more useful than the previous ones.

7.3.4 Chapter summary

We presented a more general view on the explanation of agent behavior by showing different ways and contexts to explain actions. With an example we showed how a particular behavior representation affects the resulting explanations. Being aware of different explanation types and contexts can help developers in making design choices that yield explanation-friendly representations of agent behavior, e.g., choosing an appropriate context for the objective and adding information to the model when necessary. This should result in agents providing useful explanations in their intended application.

In the presented view, we distinguish more ways to explain actions than Malle's framework (Malle, 1999) and Atkinson's argumentation scheme (Atkinson et al., 2006) (Chapter 2). In future work, user studies could be performed to further develop the perspective into an empirically evaluated framework for explaining agent behavior.

Chapter 8

Conclusion

In this thesis we have presented an approach for developing explainable agents for virtual training. The research aim of this thesis, introduced in the first chapter, was to investigate how to help trainees to learn from virtual training by BDI agents that explain their own behavior. In this final chapter we will examine to what extent the proposed approach contributes to this research aim. For that, we will answer the research questions and present the main contributions of this thesis. Subsequently, we will provide an overview of different directions for future work.

8.1 Main contributions

In the introduction we phrased two questions to guide the research of this thesis (Subsection 1.2.1). In this section we will first provide two concise answers to these questions, and then elaborate on the contributions by providing an overview per chapter.

8.1.1 Answering the research questions

The first research question is as follows.

> *"What explanations about agent behavior can help trainees to learn from virtual training?"*

This question has been answered in three different steps. Literature, described in Chapter 1 and 2, shows that (simulated) human behavior is usually explained and understood from a folk psychological perspective, that is, in terms of beliefs, goals, plans, and other mental concepts. In Chapter 4, we investigated which types of folk psychological explanations potential users of virtual training systems consider most useful. We found that, in general, explanations with a combination of the trigger belief and the goal of an action were preferred. More specific guidelines for providing explanations can be found in Section 4.4. In Chapter 5, we tested whether these explanations help trainees to learn form virtual training. We have not been able to demonstrate significant effects of the explanations on learning, but subjects reported to understand the behavior of agents better when these explained their behavior.

The second research question is as follows.

> *"How can we develop explainable BDI agents that help trainees to learn from virtual training?"*,

This question has mostly been addressed in Chapter 3. In that chapter, we described how explainable agents can be developed by constructing a goal hierarchy, implementing the goal hierarchy as a BDI agent, and adding an explanation module with explanation algorithms to the BDI agent. In Chapter 6, we showed how explainable agents can be extended with a theory of mind, making them able to provide explanations in terms of goals and beliefs attributed to other agents. In Chapter 7, we gave a more general perspective on how to make design choices when developing explainable agents.

8.1.2 Contributions per chapter

Chapter 2 provides an overview of literature that is relevant for this thesis. The overview involves literature on explanation, BDI-based programming, feedback in virtual training, and agent-based virtual training.

In Chapter 3, we presented a way to develop explainable agents based on folk psychology. We first described how to construct a goal hierarchy with the goals, adoption conditions of goals, and actions of a particular agent. We then showed how such a goal hierarchy can be implemented as a BDI agent. Next, we described how BDI agents can be extended by an explanation module that can store an agent's actions and reasoning steps and generate explanations for actions. We proposed five explanation algorithms generating different types of BDI-based explanations, e.g., an action can be explained by its goal, the circumstances that triggered the action, or the state that will be achieved by executing the action.

In Chapter 4, we described three studies investigating which explanation types are considered most useful for learning from virtual training. We used instructors (Section 4.1), novices (Section 4.2) and experts (Section 4.3) as subjects in the domains of onboard firefighting, firefighting and cooking, respectively. We found that, in general, people prefer action explanations that contain a combination of the belief that triggered the action and the goal that is achieved by the action. We also proposed some more specific guidelines for the explanation of agent behavior.

Our next step was to test how explanations generated according to our approach affected performance. Chapter 5 describes two studies that aimed to investigate the effects of our explanations on learning from virtual training, in the domains of onboard firefighting and negotiation, respectively. Since we found no effects of virtual training on learning in general, however, we were neither able to demonstrate an effect of explanation on learning. Still, subjects in the second study indicated that the agent's explanations increased their understanding in the motivations behind its behavior (Section 5.2). In a third study, we investigated the effects of our explanations on coordination in human-agent teams. We found no effect of the explanations on team performance, but the explanations affected user experience in a positive way (Section 5.3).

In Chapter 6, we extended our explainable agents approach by equipping BDI agents with a theory of mind. Theory of mind refers to the ability to attribute mental states such as beliefs and goals to others, and based on that, make predictions about their behavior. We compared two theories about theory of mind, theory-theory and simulation theory, and developed an executable model for BDI agents with a theory of mind based on simulation theory. By providing explainable agents with a theory of mind ability, we made them capable of providing explanations in terms of goals and beliefs attributed to other agents.

The last contribution of this thesis comprised a general discussion on explainable agents, provided in Chapter 7. In this chapter, we examined the advantages and disadvantages of our approach, and discussed possible application domains of explainable agents. Furthermore, we provided a general perspective on explaining agent behavior, including an overview of possible explanation types and contexts. This perspective aims to support the development of explainable agents by guiding design choices when representing an agent's behavior.

8.2 Future work

We already made some suggestions for further research at several places in this thesis. In this section, we give an overview of five main directions for future research. The directions involve the research fields of agent technology, human-computer interaction, learning sciences, and cognitive sciences.

Our first suggestion for future work is to make BDI agents able to involve new abilities and properties in explanations about their behavior. In our work, we showed how a BDI agent can be extended with a theory of mind ability, enabling the agent to explain its behavior by goals and beliefs attributed to other agents (Chapter 6). Likewise, BDI agents can be extended with other abilities and properties, e.g., emotions, personality traits and cultural awareness, so that they can use these properties in explanations about their behavior. Currently, several directions are being explored. Agents are extended, for instance, with emotions (Steunebrink et al., 2010), policies (Bradshaw et al., 2003), norms (Tinnemeier et al., 2009), norm-aware behavior (Doniec et al., 2008), cultural awareness (Mascarenhas et al., 2009), stress (Heuvelink, 2009), and social responsible behavior (Xu et al., 2007). Future research can investigate how these extensions, besides generating more realistic agent behavior, can be used to generate more realistic explanations about agent behavior.

A second direction for further research is that of user-specific explanations. In this thesis we focused on generating useful explanations for trainees in general. In other words, we investigated what explanations should be provided when there is no information available about a specific trainee. In virtual training, however, it is often possible to build a user profile while the user interacts with the system. This profile could be used to improve the usefulness of the explanations for that user. For instance, if a trainee requests an explanation for an action a second time, it is probably more useful to explain that action in another way than providing the same information again. Moreover, users' preferences for specific explanation types may be discovered over time.

A third direction for future research is to explore the use of explainable agents in applications other than virtual training. Section 5.3 of this thesis already describes a study in which our approach to explainable agents was applied to coordination in mixed human-agent teams. Furthermore, in Section 7.2 we mentioned debugging and social simulations as possible applications for explainable agents. Future work could determine requirements on explainable agents specific to these applications, and examine the agents' value in these contexts.

A fourth direction concerns the evaluation of explainable agents. In Chapter 5, we concluded that it is not easy to demonstrate an effect of explainable agents on learning, and more in general, to demonstrate an effect of virtual training on learning. In current research, many virtual training systems have been proposed, but not so many have been evaluated. We believe that the evaluation of virtual training should receive more attention. A possible approach is to use techniques developed for the evaluation of training in general to develop proper evaluation methods for virtual training systems and explainable agents in virtual training.

Our fifth suggestion for future research concerns the development of a theory on the explanation of agent behavior. In Chapter 7, we proposed a general perspective on explanations about agent behavior, where agent behavior is assumed to be similar to human behavior. We distinguished more explanation types than Malle's framework (Malle, 1999) and Atkinson's argumentation scheme (Atkinson et al., 2006) for explaining human behavior (see Chapter 2). The proposed view may be a first step towards a more comprehensive theory on the explanation of agent behavior.

8.3 Closing

This thesis is inspired by and builds upon the research of many others. With the work presented in this thesis we aim to contribute to research in the fields of agent technology, explanation and virtual training, and hope that our work will also be of use for others. Moreover, we hope to support future trainees in their learning from virtual training. We started this thesis with the following fragment of a training scenario.

> *When you are called for a fire in a house, you and your four team members get into a fire engine and drive to the location of the incident as quickly as possible. Once arrived, you see smoke coming out of a house, people are standing dangerously close by and from the distance a siren is approaching. You have to assess the situation quickly, make an attack plan and instruct your team. Subsequently, while discussing with a policeman where to block the road, you see that your first two team members, against your instructions, enter the house through the back instead of the front door. The next moment, a woman tells you in panic that her dog is still inside the house...*

The approach presented in this thesis allows for the development of agents that can explain their own behavior, also in terms of mental states attributed to others. With such agents, it is possible to implement the scenario unfolding as follows.

You wonder whether you should let your other two team members attack the fire because you are not sure what the first two are doing, or let them search for the dog. You believe that there must be a reason for the first two to deviate from their plan, and quickly decide to instruct the second two to go through the front door, check if the dog is at that side of the house, and if it is not, attack the fire. Somewhat later all team members are back with the dog. The fire has been extinguished. When you ask the first two team members why they entered the house through the back door, they answer: "We believed that we heard a dog barking at the back of the house and wanted to save it, and we believed that you would send the other team to the front door to extinguish the fire."

Bibliography

Anderson, J., Conrad, F., and Corbett, A. (1989). Skill acquisition and the LISP tutor. *Cognitive Science*, 13(4):467–506.

Anderson, J. and Lebiere, C. (1998). *The Atomic Components of Thought*. Lawrence Erlbaum Associates, Mahwah, NJ.

Atkinson, K., Bench-Capon, T., and McBurney, P. (2006). Computational representation of practical argument. *Synthese*, 152(2):157–206.

Aylett, R. (1999). Narrative in virtual environments: towards emergent narrative. In *Proceedings of the 1999 AAAI Fall Symposium*, pages 83–86. AAAI Press, Menlo Park, CA.

Aylett, R. and Louchart, S. (2008). If I were you: double appraisal in affective agents. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2008)*, pages 1233–1236, Estoril, Portugal. IFAAMAS.

Baldwin, T. and Ford, K. (1988). Transfer of training: a review and directions for future research. *Personnel Psychology*, 41(1):63–105.

Baron-Cohen, S. (1995). *Mindblindness: an essay on autism and theory of mind*. MIT Press, Cambridge, MA.

Blackmon, M. and Polson, P. (2002). Combining two technologies to improve aviation training design. In *Proceedings of the Conference on Human Computer Interaction (HCI 2002)*, pages 24–29. AAAI Press, Menlo Park, CA.

Boella, G. and Van der Torre, L. (2004). Groups as agents with mental attitudes. In *Proceedings of the Third International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2004)*, pages 964–971. IEEE Computer Society, Washington, DC.

Bordini, R., Dastani, M., Dix, J., and Fallah-Seghrouchni, A., editors (2005). *Multi-agent programming: languages, platforms and applications*. Springer-Verlag, Berlin/Heidelberg, Germany.

Bordini, R., Dastani, M., Dix, J., and Fallah-Seghrouchni, A., editors (2009). *Multi-agent programming: languages, tools and applications*. Springer-Verlag, Berlin/Heidelberg, Germany.

Bordini, R., Hubner, J., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*. Wiley, Hoboken, NJ.

Bosse, T., Memon, Z., and Treur, J. (2011). A recursive BDI-agent model for theory of mind and its applications. *Applied Artificial Intelligence*, 25(1):1–44.

Bradshaw, J., Feltovich, P., and Johnson, M. (2011). Human-agent interaction. In *Handbook of Human-Computer Interaction*. Ashgate publishing, Aldershot, UK.

Bradshaw, J., Uszok, A., Jeffers, R., Suri, N., Hayes, P., Burstein, M., Acquisti, A., Benyo, B., Breedy, M., Carvalho, M., Diller, D., Johnson, M., Kulkarni, S., Lott, J., Sierhuis, M., and Van Hoof, R. (2003). Representation and reasoning about DAML-based policy and domain services in KAoS. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2003)*, pages 835–842. ACM Press, New York, NY.

Bratman, M. (1987). *Intention, Plans and Practical Reason*. Harvard University Press, Cambridge, MA.

Braubach, L., Pokahr, A., and Lamersdorf, W. (2005). Extending the capability concept for flexible BDI agent modularization. In *Proceedings of the Third International Workshop on Programming Multiagent Systems (ProMAS 2005)*, pages 139–155. Springer-Verlag, Berlin/Heidelberg, Germany.

Broekens, J. and De Groot, D. (2006). Formalizing cognitive appraisal: from theory to computation. In *Cybernetics and Systems 2006*, pages 595–600. Austrian Society for Cybernetics Studies.

Broekens, J., Harbers, M., Brinkman, W., Jonker, C., Van den Bosch, K., and Meyer, J.-J. (2011). Towards effective virtual negotiation training. In *Proceedings of the Eleventh International Conference on Intelligent Virtual Agents (IVA 2011)*, to appear.

Broekens, J., Harbers, M., Hindriks, K., Van den Bosch, K., Jonker, C., and Meyer, J.-J. (2010a). Do you get it? User-evaluated explainable BDI agents. In *Proceedings of the Eight German Conference on Multiagent System Technologies (MATES 2010)*, pages 28–39. Springer-Verlag, Berlin/Heidelberg, Germany.

Broekens, J., Jonker, C., and Meyer, J.-J. (2010b). Affective negotiation support systems. *Journal of Ambient Intelligence and Smart Environments*, 2(2):121–144.

Buchanan, B. and Shortliffe, E. (1984). *Rule-based Expert Systems: the MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Longman Publishing, Reading, MA.

Buchanan, G. and Seligman, M. (1995). *Explanatory Style*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Busetta, P., Howden, N., Ronnquist, R., and Hodgson, A. (2000). Structuring BDI agents in functional clusters. In *Intelligent Agents VI: Theories, Architectures and Languages*, pages 277–289.

Busetta, P., Ronnquist, R., Hodgson, A., and Lucas, A. (1999). JACK intelligent agents - components for intelligent agents in Java. *AgentLink News Letter*, 2(1):2–5.

Cannon-Bowers, J., Burns, J., Salas, E., and Pruitt, J. (1998). *Making decisions under stress: implications for individual and team training*, chapter Advanced technology in scenario-based training, pages 365–374. APA, Washington, DC.

Carruthers, P. (1996). *Theories of theories of mind*, chapter Simulation and self-knowledge: a defence of the theory-theory. Cambridge University Press, Cambridge, MA.

Carruthers, P. and Smith, P., editors (1996). *Theories of theories of mind*. Cambridge University Press, Cambridge, MA.

Cassell, J. and Bickmore, T. (2003). Negotiated collusion: modeling social language and its relationship effects in intelligent agents. *User Modeling and User-Adapted Interaction*, 13(1):89–132.

Castelfranchi, C. (1997). To be or not to be an agent. In Muller, J., Wooldridge, M., and Jennings, N., editors, *Intelligent Agents III*, pages 37–41. Springer-Verlag, Berlin/Heidelberg, Germany.

Cavazza, M., Charles, F., and Mead, S. J. (2002.). Character-based interactive storytelling. *IEEE Intelligent Systems*, 17(4):17–24.

Chandrasekaran, B. and Josephson, J. (1999). Cognitive modeling for simulation goals: a research strategy for computer-generated forces. In *Proceedings of the Eighth Computer Generated Forces and Behavioural Representation Conference*, pages 239–250.

Core, M., Lane, H., Van Lent, M., Gomboc, D., Solomon, S., and Rosenberg, M. (2006a). Building explainable artificial intelligence systems. In *Proceedings of the Eighteenth Conference on Innovative Applications of Artificial Intelligence (IAAA 2006)*, pages 1766–1773. AAAI Press, Menlo Park, CA.

Core, M., Traum, T., Lane, H., Swartout, W., Gratch, J., and Van Lent, M. (2006b). Teaching negotiation skills through practice and reflection with virtual humans. *Simulation*, 82(11):685–701.

Dastani, M. (2008). 2APL: a practical agent programming language. *Autonomous Agents and Multi-agent Systems*, 16(3):214–248.

Dastani, M. (2009). *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, chapter Modular rule-based programming in 2APL, pages 25–49. IGI Global, Hershey, PA.

Davies, N. and Mehdi, Q. (2006). BDI for intelligent agents in computer games. In *Proceedings of the Eighth International Conference on Computer Games: AI and Mobile Systems*.

De Giacomo, G., Lesperance, Y., and Levesque, H. (2000). Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1-2):109–169.

Dennett, D. (1987). *The Intentional Stance*. MIT Press, Cambridge, MA.

Dhaliwal, J. and Benbasat, I. (1996). The use and effects of knowledge-based system explanations: theoretical foundations and a framework for empirical evaluation. *Information systems research*, 7(6):243–361.

Dignum, F., Westra, J., Van Doesburg, W., and Harbers, M. (2009). Games and agents: designing intelligent gameplay. *International Journal of Computer Games Technology*.

Doniec, A., Mandiau, R., Piechowiak, S., and Espié, S. (2008). Controlling non-normative behaviors by anticipation for autonomous agents. *Web Intelligence and Agent Systems*, 6(1):29–42.

Ericsson, K., Krampe, R., and Tesch-Rmer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review*, 100(3):363–406.

Flin, R. and Arbuthnot, K., editors (2002). *Incident command: tales from the hot seat*. Ashgate publishing, Aldershot, UK.

Fowlkes, J., Dwyer, D., Oser, R., and Salas, E. (1998). Event-based approach to training (EBAT). *The International Journal of Aviation Psychology*, 8(3):209–222.

Franklin, S. and Graesser, A. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, pages 21–36. Springer-Verlag, Berlin/Heidelberg, Germany.

GATE (2011). URL: http://gate.gameresearch.nl/.

Georgeff, M. and Lansky, A. (1987). Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 677–682. AAAI Press, Menlo Park, CA.

Gmytrasiewicz, P. and Durfee, E. (1995). A rigorous, operational formalization of recursive modeling. In *Proceedings of the First International Conference on Multiagent Systems*, pages 125–132. AAAI Press, Menlo Park, CA.

Goldman, A. (1992). In defence of the simulation theory. *Mind and Language*, 7(1-2):104–119.

Gomboc, D., Solomon, S., Core, M. G., Lane, H. C., and Van Lent, M. (2005). Design recommendations to support automated explanation and tutoring. In *Proceedings of the Fourteenth Conference on Behavior Representation in Modeling and Simulation (BRIMS 2005)*.

Gordon, A., Van Lent, M., Van Velsen, M., Carpenter, P., and Jhala, A. (2004). Branching storylines in virtual reality environments for leadership development. In *Proceedings of the Conference on Innovative Applications of Artifical Intelligence (IAAA 2004)*, pages 844–851. AAAI Press, Menlo Park, CA.

Gordon, R. (1996). *Theories of theories of mind*, chapter 'Radical' simulationism. Cambridge University Press, Cambridge, MA.

Graesser, A. C., Chipman, P., Haynes, B. C., and Olney, A. (2005). Autotutor: an intelligent tutoring system with mixed-initiative dialogue. *Education, IEEE Transactions on*, 48(4):612–618.

Gratch, J., Rickel, J., Andr, E., Cassell, J., Petajan, E., and Badler, N. (2002). Creating interactive virtual humans: some assembly required. *Intelligent Systems*, 17(4):54–63.

Gregor, S. and Benbasat, I. (1999). Explanation from intelligent systems: theoretical foundations and implications for practice. *MIS Quarterly*, 23(4):497–530.

Hall, L., Woods, S., and Aylett, R. (2006). Fearnot! Involving children in the design of a virtual learning environment. *International Journal of Artificial Intelligence in Education*, 16(4):327–351.

Harbers, M., Bradshaw, J., Johnson, M., Feltovich, P., Van den Bosch, K., and Meyer, J.-J. (2011a). Explanation and coordination in human-agent teams: a study in the BW4T testbed. In *Proceedings of the Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems (COIN 2011)*, to appear.

Harbers, M., Broekens, J., Van den Bosch, K., and Meyer, J.-J. (2010a). Guidelines for developing explainable cognitive models. In *Proceedings of the International Conference on Cognitive Modeling (ICCM 2010)*, pages 85–90.

Harbers, M., Van den Bosch, K., Dignum, F., and Meyer, J.-J. (2008). A cognitive model for the generation and explanation of behaviour in virtual training systems. In *Proceedings of the Third International Workshop on Explanation-aware Computing (ExaCt)*, pages 96–107. University of Patras, Patras, Greece.

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2009a). Enhancing training by using agents with a theory of mind. In *Proceedings of the Workshop on Educational Uses of Multi-Agent Systems (EduMAS 2009)*, pages 23–30.

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2009b). A methodology for developing self-explaining agents for virtual training. In *Proceedings of Eighth International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2009)*, pages 1129–1130. IFAAMAS.

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2009c). A methodology for developing self-explaining agents for virtual training. In *Proceedings of the Workshop on LAnguages, methodologies and Development tools for multi-agent systemS (LADS 2009)*, pages 168–182. Springer-Verlag, Berlin/Heidelberg, Germany.

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2009d). Modeling agents with a theory of mind. In *Proceedings of the International Conference on Intelligent Agent Technology (IAT 2009)*, pages 217–224. IEEE Computer Society, Washington, DC.

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2009e). A study into preferred explanations of virtual agent behavior. In *Proceedings of the Ninth International Conference on Intelligent Virtual Agents (IVA 2009)*, pages 132–145. Springer-Verlag, Berlin/Heidelberg, Germany.

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2010b). Design and evaluation of explainable BDI agents. In *Proceedings of the International Conference on Intelligent Agent Technology (IAT 2010)*, pages 125–132. IEEE Computer Society, Washington, DC.

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2010c). Explaining simulations through self explaining agents. *Journal of Artificial Societies and Social Simulation*, 12(1)(4).

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2011b). *Multi-Agent Systems for Education and Interactive Entertainment: design, Use and Experience*, chapter Agents with a Theory of Mind in Virtual Training, pages 172–187. IGI Global, Hershey, PA.

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2011c). A theoretical framework for explaining agent behavior. In *Proceedings of the First International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2011)*, pages 228–231.

Harbers, M., Van den Bosch, K., and Meyer, J.-J. (2012). Modeling agents with a theory of mind: Theory-theory versus simulation theory. *Journal of Web Intelligence and Agent Systems*, 10(3):to appear.

Harmon, S., Hoffmann, D., Gonzalez, A., Knauf, R., and Barr, V. (2002). Validation of human behavior representation. In *Proceedings of Workshop on Foundations for Modeling and Simulation (MS), Verification and Validation (VV) in the 21st Century*. Society of Modeling and Simulation (SCS), San Diego, CA.

Haynes, S., Cohen, M., and Ritter, F. (2009). Designs for explaining intelligent agents. *International Journal of Human-Computer Studies*, 67(1):90–110.

Heal, J. (1996). *Theories of theories of mind*, chapter Simulation, theory, and content. Cambridge University Press, Cambridge, MA.

Heider, F. (1958). *The Psychology of Interpersonal Relations*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Hempel, C. and Oppenheim, P. (1948). studies in the logic of explanation. *Philosophy of Science*, 15(2):135–175.

Herlocker, J. (1999). Position statement - explanations in recommender systems. In *Proceedings of the Workshop on Computer Human Interaction (CHI 1999)*.

Heuvelink, A. (2009). *Cognitive Models for Training Simulations*. PhD thesis, Vrije Universiteit Amsterdam, The Netherlands.

Heuvelink, A., Van den Bosch, K., Van Doesburg, W., and Harbers, M. (2009). Intelligent agent supported training in virtual simulations. In *Proceedings of the NATO HFM-169 Workshop on Human Dimensions in Embedded Virtual Simulation*.

Hindriks, K. (2007). Modules as policy-based intentions: modular agent programming in GOAL. In *Proceedings of the Fifth International Workshop on Programming Multiagent Systems (ProMAS 2007)*, pages 156–171.

Hindriks, K. (2009). *Multi-Agent Programming: languages, Tools and Applications*, chapter Programming Rational Agents in GOAL, pages 119–157. Springer-Verlag, Berlin/Heidelberg, Germany.

Hindriks, K., De Boer, F., Van der Hoek, W., and Meyer, J.-J. (1999). Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401.

Hindriks, K. and Jonker, C. (2008). Creating human-machine synergy in negotiation support systems: towards the pocket negotiator. In *Proceedings of the First International Working Conference on Human Factors and Computational Models in Negotiation (HuCom 2008)*, pages 47–54.

Houtkamp, J. and Bos, F. (2007). Evaluation of a virtual scenario training for leading firefighters. In *Proceedings of the Fourth International Conference on Information Systems for Crisis Response and Management (ISCRAM 2007)*, pages 565–570. VUB Press, Brussels, Belgium.

Hutchins, S., Kemple, W., Porter, G., and Sovereign, M. (1999). Evaluating human performance in command and control environments. In *Proceedings of the Symposium on Command and Control Research and Technology*, pages 50–52.

IEEE (1998). Standard for a software quality metrics methodology. *IEEE Std 1061-1998*.

Jennings, N. and Wooldridge, M. (1998). *Agent Technology: Foundations, Applications and Markets*. Springer-Verlag, Berlin/Heidelberg, Germany.

Jensen, R., Nolan, M., and Chen, D. (2005). Automatic causal explanation analysis for combined arms training AAR. In *Proceedings of the Industry/Interservice, Training, Simulation Education Conference (I/ITSEC 2005)*.

Johnson, L. (1994). Agents that learn to explain themselves. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1257–1263.

Johnson, M., Bradshaw, J., Feltovich, P., Jonker, C., Van Riemsdijk, M., and Sierhuis, M. (2011). Coactive design: why interdependence must shape autonomy. In *Proceedings of the Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems (COIN 2011)*, pages 172–191.

Johnson, M., Jonker, C., Van Riemsdijk, M., Feltovich, P., and Bradshaw, J. (2009). Joint activity testbed: blocks world for teams (BW4T). In *Proceedings of Engineering Societies in the Agents World (ESAW 2009)*, pages 254–256. Springer-Verlag, Berlin/Heidelberg, Germany.

Keil, F. (2006). Explanation and understanding. *Annual Reviews Psychology*, 57(1):227–254.

Kelley, H. (1967). Attribution theory in social psychology. In *Nebraska Symposium on Motivation*, pages 192–240. University of Nebraska Press, Lincoln, NE.

Keysar, B., Lin, S., and Barr, D. (2003). Limits on theory of mind use in adults. *Cognition*, 89(1):25–41.

Kirschner, P., Sweller, J., and Clark, R. (2006). Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, 2(41):75–86.

Klein, G. (1998). *Sources of power: how people make decisions.* MIT Press, Cambridge, MA.

Koedinger, K. and Anderson, J. (1998). Illustrating principled design: the early evolution of a cognitive tutor for algebra symbolization. In *Interactive Learning Environments*, pages 161–180.

Kozhevnikov, M. and Hegarty, M. (2001). Impetus beliefs as default heuristics: dissociation between explicit and implicit knowledge about motion. *Psychonomic Bulletin Review*, 8(3):439–453.

Laird, J. (2001). Using a computer game to develop advanced AI. *Computer*, 34(7):70–75.

Laird, J. and Nielsen (1994). Coordinated behavior of computer generated forces in TacAir-Soar. In *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, pages 325–332.

Lesser, V., Decker, K., Carver, N., Garvey, A., Neiman, D., Nagendra Prasad, M., and Wagner, T. (2004). Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous agents and multi-agent systems*, 9(1-2):87–143.

Livak, T., Heffernan, N., and Moyer, D. (2004). Using cognitive models for computer generated forces and human tutoring. In *Proceedings of the Thirteenth Conference on Behavior Representation in Modeling and Simulation (BRIMS 2004).*

Lockelt, M., Pecourt, M., and Pfleger, N. (2005). Balancing narrative control and autonomy for virtual characters in a game scenario. In *First International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN 2005)*, pages 251–255.

Louchart, S. and Aylett, R. (2005). Managing a non-linear scenario - a narrative evolution. In *Virtual Storytelling*, pages 148–157. Springer-Verlag, Berlin/Heidelberg, Germany.

Malle, B. (1999). How people explain behavior: a new theoretical framework. *Personality and Social Psychology Review*, 3(1):23–48.

March, S. and Smith, G. (1995). Design and natural science research on information technology. *Decision Support Systems*, 15(4):251–266.

Mascarenhas, S., Dias, J. a., Prada, R., and Paiva, A. (2009). One for all or one for one? the influence of cultural dimensions in virtual agents' behaviour. In *Proceedings of the Ninth International Conference on Intelligent Virtual Agents (IVA 2009)*, pages 272–286. Springer-Verlag, Berlin/Heidelberg, Germany.

Mioch, T., Harbers, M., Van Doesburg, W., and Van den Bosch, K. (2008). Enhancing human understanding through intelligent explanations. In *Proceedings of the First International Workshop on Human Aspects in Ambient Intelligence*, pages 327–337. Springer-Verlag, Berlin/Heidelberg, Germany.

Mol, L., Verbrugge, L., and Hendriks, P. (2005). Learning to reason about other people's minds. In *Proceedings of the Joint Symposium on Virtual Social Agents*. University of Hertfordshire Press, Hatfield, UK.

Moulin, B., Irandoust, H., Blanger, M., and Desbordes, G. (2002). Explanation and argumentation capabilities: towards the creation of more persuasive agents. *Artificial Intelligence Review*, 17(3):169–222.

Murray, T. (1999). Authoring intelligent tutoring systems: an analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10(1):98–129.

Nakatsu, R. (2004). Explanatory power of intelligent systems: a research framework. In *International Conference on Decision Support Systems*, pages 568–577.

Nickerson, S. (1999). How we know -and sometimes misjudge- what others know: imputing one's own knowledge to others. *Psychological Bulletin*, 125(6):737–759.

Norling, E. (2003). Capturing the quake player: using a BDI agent to model human behaviour. In Rosenschein, J., Sandholm, T., Wooldridge, M., and Yokoo, M., editors, *Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2003)*, pages 1080–1081. ACM Press, New York, NY.

Norling, E. (2004). Folk psychology for human modelling: extending the BDI paradigm. In *Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS04)*, pages 202–209. ACM Press, New York, NY.

Oser, R. (1999). A structured approach for scenario-based training. In *Proceedings of the Fourtythird Annual Meeting of the Human Factors and Ergonomics Society*, pages 1138–1142.

Patel, P. and Hexmoor, H. (2009). Designing bots with BDI agents. In *Proceedings of the International Symposium on Collaborative Technologies and Systems*, pages 180–186. IEEE Computer Society, Washington, DC.

Peeters, M., Van den Bosch, K., Meyer, J.-J., and Neerincx, M. (2011). Scenario-based training: director's cut. In *Proceedings of the fifteenth International Conference on Artificial Intelligence in Education*.

Perner, J. (1996). *Theories of theories of mind*, chapter simulation as explicitation of predication-implicit knowledge about the mind: arguments for a simulation-theory mix. Cambridge University Press, Cambridge, MA.

Peters, C. (2005). Foundations of an agent theory of mind model for conversation initiation in virtual environments. In *Proceedings of Artificial Intelligence and the Simulation of Behaviour (AISB 2005), Symposium on Virtual Social Agents: Mind-Minding Agents*.

Pokahr, A., Braubach, L., and Lamersdorf, W. (2003). Jadex: implementing a BDI-infrastructure for JADE agents. *In Search of Innovation*, 3(3):76–85.

Polson, M. and Richardson, J. (1988). *Foundations of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates, Mahwah, NJ.

Premack, D. and Woodruff, G. (1978). Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, 1(4):515–526.

Psotka, J., Massey, D., and Mutter, S., editors (1988). *Intelligent Tutoring Systems: lessons Learned*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Pynadath, D. and Marsella, S. (2005). PsychSim: modeling theory of mind with decision-theoretic agents. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1181–1186.

Randolph, J. (2008). Online kappa calculator. Retrieved March 6, 2009 , from http://justus.randolph.name/kappa.

Rao, A. and Georgeff, M. (1991). Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann publishers, San Mateo, CA.

Rao, A. and Georgeff, M. (1995). BDI-agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS 1995)*, pages 312–319.

Reilly, W. S. and Bates, J. (1995). Natural negotiation for believable agents. Technical report, Carnegie Mellon University, Pittsburgh, PA.

Reynolds, C. (1987). Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, 21(4):25–34.

Riedl and Stern (2006). Believable agents and intelligent scenario direction for social and cultural leadership training. In *Proceedings of the fifteenth Conference on Behavior Representation in Modeling and Simulation*.

Rosenbloom, P., Laird, J., and Newell, A. (1993). *The Soar Papers: readings on Integrated Intelligence*. MIT Press, Cambridge, MA.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: a Modern Approach*. Pearson Education, New Jersey, NJ, second edition.

Salmon, W. (1989). *Four decades of scientific explanation*. University of Minneapolis Press, Minneapolis, MN.

Sardina, S., De Silva, L., and Padgham, L. (2006). Hierarchical planning in BDI agent programming languages: a formal approach. In *Proceedings of the Fifth International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2006)*, pages 1001–1008. ACM Press, New York, NY.

Scassellati, B. (2002). Theory of mind for a humanoid robot. *Autonomous Robots*, 12(1):13–24.

Schraagen, J., Chipman, S., and Shalin, V., editors (2000). *Cognitive Task Analysis*. Lawrence Erlbaum Associates, Mahway, NJ.

Shepherd, A. (1998). HTA as a framework for task analysis. *Ergonomics*, 41(11):1537–1552.

Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence archive*, 60(1):51–92.

Simon, H. (1969). *The Sciences of the Artificial*. MIT Press, Cambridge, MA.

Sindlar, M., Dastani, M., and Meyer, J.-J. (2011). Programming mental state abduction. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2011)*, pages 301–308. IFAAMAS.

Srmo, R., Cassens, J., and Aamodt, A. (2005). Explanation in case-based reasoningperspectives and goals. *Artificial Intelligence Review*, 24(2):109–143.

Steunebrink, B., Dastani, M., and Meyer, J.-J. (2010). Emotions to control agent deliberation. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2010)*, pages 973–980. IFAAMAS.

Swartjes, I. and Theune, M. (2009). Iterative authoring using story generation feedback: debugging or co-creation? In *Proceedings of the Second International Conference on Interactive Digital Storytelling (ICIDS 2009)*, pages 62–73. Springer-Verlag, Berlin/Heidelberg, Germany.

Swartout, W. and Moore, J. (1993). *Second-Generation Expert Systems*, chapter Explanation in Second-Generation Expert Systems, pages 543–585. Springer-Verlag, Berlin/Heidelberg, Germany.

Swartout, W., Paris, C., and Moore, J. (1991). Explanations in knowledge systems: design for explainable expert systems. *IEEE Intelligent Systems*, 6(3):58–64.

Taylor, G., Jones, R., Goldstein, M., Frederiksen, R., and Wray, R. (2002). VISTA: a generic toolkit for visualizing agent behavior. In *Proceedings of the Eleventh Conference on Computer Generated Forces and Behavior Representation*, pages 29–40.

Taylor, G., Knudsen, K., and Scott Holt, L. (2006). Explaining agent behavior. In *Proceedings of the Fifteenth Conference on Behavior Representation in Modeling and Simulation (BRIMS)*.

Teichert, M. and Stacy, A. (2002). Promoting understanding of chemical bonding and spontaneity through student explanation and integration of ideas. *Journal of Research in Science Teaching*, 39(6):464–496.

Theune, M., Faas, S., Heylen, D. K. J., and Nijholt, A. (2003). The virtual storyteller: story creation by intelligent agents. In *Proceedings of the Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2003)*, pages 204–215. Fraunhofer IRB Verlag, Darmstadt, Germany.

Tinnemeier, N., Dastani, M., and Meyer, J.-J. (2009). Roles and norms for programming agent organizations. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2009)*, pages 121–128. IFAAMAS.

Traum, D., Marsella, S., Gratch, J., Lee, J., and Hartholt, A. (2008). Multi-party, multi-issue, multi-strategy negotiation for multi-modal virtual agents. In *Proceedings of the Eighth International Conference on Intelligent Virtual Agents (IVA 2008)*, pages 117–130. Springer-Verlag, Berlin/Heidelberg, Germany.

Traum, D., Rickel, J., Gratch, J., and Marsella, S. (2003). Negotiation over tasks in hybrid human-agent teams for simulation-based training. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2003)*, pages 441–448. ACM Press, New York, NY.

Van den Bosch, K. (1999). *Engineering Psychology and Cognitive Ergonomics*, chapter Durable Competence in Procedural Tasks Through Appropriate Instruction and Training, pages 431–438. Ashgate publishing, Aldershot, UK.

Van den Bosch, K., Harbers, M., Heuvelink, A., and Van Doesburg, W. (2009). Intelligent agents for training on-board fire fighting. In *Proceedings of the Second International Conference on Digital Human Modeling*, pages 463–472. Springer-Verlag, Berlin/Heidelberg, Germany.

Van den Bosch, K. and Riemersma, J. (2004). *Scaled Worlds: development, Validation and Applications*, chapter Reflections on scenario-based training in tactical command, pages 1–21. Ashgate publishing, Aldershot, UK.

Van der Vaart, E. and Verbrugge, R. (2008). Agent-based models for animal cognition: a proposal and prototype. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2008)*, pages 1145–1152. IFAAMAS.

Van Doesburg, W. (2007). Quality assessment of human behavior models. In *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, pages 137–144. ACTA Press, Calgary, Canada.

Van Doesburg, W., Heuvelink, A., and Van den Broek, E. (2005). TACOP: a cognitive agent for a naval training simulation environment. In M. Pechoucek, D. Steiner, S. T., editor, *Proceedings of the Industry Track of the International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2005)*, pages 34–41. ACM Press, New York, NY.

Van Doesburg, W. and Van den Bosch, K. (2005). Cognitive model supported tactical training simulation. In *Proceedings of the Fourteenth Conference on Behavior Representation in Modeling and Simulation (BRIMS 2005)*, pages 313–319.

Van Lehn, K. (2006). The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265.

Van Lent, M., Fisher, W., and Mancuso, M. (2004). An explainable artificial intelligence system for small-unit tactical behavior. In *Proceedings of the Sixteenth conference on Innovative Applications of Artifical Intelligence (IAAA 2004)*, pages 900–907. AAAI Press,Menlo Park, CA.

Westra, J., Dignum, F., and Dignum, V. (2011). *Agents for Games and Simulations II*, chapter Guiding User Adaptation in Serious Games, pages 117–131. Springer-Verlag, Berlin/Heidelberg, Germany.

Wick, M. and Tompson, W. (1992). Reconstructive expert system explanation. *Artificial Intelligence*, 54(1-2):33–70.

Wimmer, H. and Perner, J. (1983). Beliefs about beliefs: representation and constraining function of wrong beliefs in young children's understanding of deception. *Cognition*, 13(1):103–128.

Wooldridge, M. (2000). *Reasoning about Rational Agents*. MIT Press, Cambridge, MA.

Wooldridge, M. (2002). *Introduction to Multi-agent Systems*. Wiley, Hoboken, NJ.

Wooldridge, M. and Jennings, N. (1995). Intelligent agents: theory and practice. *Knowledge Engineering Review*, 10(2):115–152.

Xu, M., Padgham, L., Mbala, A., and Harland, J. (2007). Tracking reliability and helpfulness in agent interactions. *Web Intelligence and Agent Systems*, 5(1):31–46.

Ye, R. and Johnson, P. (1995). The impact of explanation facilities on user acceptance of expert systems advice. *MIS Quarterly*, 19(2):157–172.

Yen, J., Fan, X., and Volz, R. (2004). Information needs in agent teamwork. *Web Intelligence and Agent Systems*, 2(3):231–248.

Young, A. and Harper, K. (2005). TRACE: an ontology-based approach to generic traceability tools for human behavior models. In *Proceedings of the Fourteenth Conference on Behavior Representation in Modeling and Simulation (BRIMS 2005).*

Young, M. (2003). Human performance model validation: one size does not fit all. In *Proceedings of the Summer Computer Simulation Conference*, pages 732–736.

Zachary, W., Cannon-Bowers, J., Bilazarian, P., Krecker, D., Lardieri, P., and Burns, J. (1999). The advanced embedded training system (AETS): an intelligent embedded tutoring system for tactical team training. *International Journal of Artificial Intelligence in Education*, 10(3):257–277.

Summary

Computer games are more and more often used for training purposes. These virtual training games are exploited to train competences like leadership, negotiation and social skills. In a virtual training session, a human trainee interacts with one or more virtual characters playing the trainee's team members, colleagues or opponents. To learn from virtual training, it is important that the virtual characters display realistic human behavior. This can be achieved by letting human players, e.g., instructors or co-students, control the virtual game characters. Another way is to let intelligent software agents generate the behavior of virtual characters automatically. Using intelligent agents instead of human players allows trainees to train independently, where and wherever they want.

A potential problem of using intelligent agents in virtual training is that trainees do not always understand the behavior of the agents. For instance, virtual team members (played by intelligent agents) that do not follow the instructions of their leader (played by the trainee) may have misunderstood the instructions, or disobey their leader on purpose. After playing the scenario, the trainee does not know whether he should communicate clearer, be more persuasive, or give better or safer instructions. This problem can be solved by letting virtual agents explain the reasons behind their behavior. When a trainee can ask its virtual team members, colleagues or opponents to explain the motivations for their actions, he is are given the opportunity to understand played scenarios and his own performance better. This thesis is about explaining agent behavior in virtual training, and proposes a way to generate explanations of the behavior of virtual agents in training games automatically.

There already exist several approaches to explaining agent behavior in virtual training. In initial proposals, agent actions were often explained by mere physical properties, e.g., an agent's position or its health. However, psychological research shows that people usually explain and understand human (or human-like) behavior in terms of mental concepts like beliefs, goals and intentions. In more recent approaches to explaining agent behavior in virtual training, the explanations also include an agent's mental properties, such as its goals and intentions.

Existing approaches that provide explanations including an agent's mental properties are all application-independent explanation components. That means that they can be applied to different training systems. Each training system uses its own ways to represent agent behavior, e.g., by a neural network or a collection of if-then rules. It rarely happens that an agent's goals and intentions are explicitly represented in a training system. In order to develop an explanation component that can provide explanations including

an agent's mental properties, developers have to add the goals and motivations for the agent's actions manually. Thus, agent behavior is represented both in the training system and in the explanation component.

In this thesis, we propose an approach that saves developers from representing agent behavior twice. The approach is not application-independent, but instead, integrates the development of agents and their explanation capabilities. By exerting control over the development of agents in virtual training, we can ensure that an agent's behavior representations contain all information needed to explain its actions. We use a BDI-based (Belief Desire Intention) agent programming language to represent the behavior of agents in virtual training. The behavior of a BDI agent is represented by beliefs, goals, plans, and intentions, and its actions are determined by a reasoning process on its mental concepts. Thus, the mental concepts that are responsible for the generation of an action can also be used to explain that action.

The thesis describes how to develop agents that can explain their actions in a BDI programming language. First, a goal hierarchy should be constructed with the agent's main goal at the top, its subgoals below that, and its actions at the bottom of the hierarchy. Adoption conditions in the hierarchy specify when a (sub)goal or action is adopted. Second, the goal hierarchy should be implemented as a BDI agent. Goals and adoption conditions in the hierarchy are represented as goals and beliefs in the agent, respectively. Third, the BDI agents should be extended by an explanation module that stores the agent's actions and reasoning steps, and can generate explanations for these actions. We propose several explanation algorithms that generate different types of explanations. For example, an action can be explained by its goal, the circumstances that triggered the action, or the state that will be achieved by executing the action.

Since our approach to explainable agents can generate different explanation types, we performed three user studies to investigate which explanation types are considered most useful for learning from virtual training. We used instructors, novices and experts as subjects in the domains of onboard firefighting, firefighting and cooking, respectively. We found, among other things, that people generally prefer explanations that contain a combination of (1) the belief that triggered the action and (2) the goal that is achieved by the action. In the thesis we introduce several guidelines for the explanation of agent behavior that were based on the results of the studies.

The thesis describes three studies that validate the proposed approach. In all three studies it was tested how explanations generated according to our approach affected performance. Two of the studies investigated the effects of explanations on learning from virtual training, in the domains of onboard firefighting and negotiation, respectively. Since we found no effects of virtual training on learning in general, we were neither able to demonstrate an effect of explanation on learning. However, subjects in the second study indicated that the agent's explanations increased their understanding in the motivations behind its behavior. In a third study, we investigated the effects of explanations on coordination in human-agent teams. We found no effect of the explanations on team performance, but subjects better understood the agents behavior and preferred the amount of information provided by the agent when the agent explained its behavior.

In one of the first three studies, we found that instructors often prefer explanations containing predictions about the behavior of other agents. In order to provide such ex-

planations, we extended our explainable agent approach by equipping the agents with a theory of mind. Theory of mind refers to the ability to attribute mental states such as beliefs and goals to others, and based on that, make predictions about their behavior. By providing explainable agents with a theory of mind ability, they become capable of providing explanations in terms of goals and beliefs attributed to other agents. We compared two philosophical theories about theory of mind, theory-theory and simulation theory, on their appropriateness for modeling explainable agents with a theory of mind. We found that the theories could generate equally useful explanations, but that agent models based on simulation theory are more efficient to develop. Therefore, we developed an executable model for BDI agents with a theory of mind based on simulation theory.

The thesis ends with a discussion of the advantages and disadvantages of our approach, and provides several suggestions for future research. The main conclusions of this thesis are that (1) BDI agents can be used to generate explanations about their own actions in terms of goals, beliefs, and plans, (2) explanations generated according to our approach can increase understanding of an agent's behavior, and (3) our approach is particularly useful when the explainable agents are reused in different training scenarios.

Samenvatting

Computerspellen worden steeds vaker gebruikt om mensen te trainen. Zulke virtuele trainingsspellen worden bijvoorbeeld ingezet voor het trainen van leiderschap, onderhandelen en sociale vaardigheden. De trainee krijgt in een trainingssessie meestal te maken met één of meerdere virtuele teamleden, collega's of tegenstanders. Om van virtuele training te leren is het belangrijk dat deze virtuele personages realistisch menselijk gedrag vertonen. Er zijn twee manieren om dat te bereiken. Ten eerste kunnen mensen de virtuele personages aansturen, bijvoorbeeld instructeurs of medestudenten van de trainee. Ten tweede kunnen intelligente software agenten het gedrag van de virtuele personages automatisch genereren. Een voordeel van het gebruik van intelligente agenten in plaats van menselijke spelers is dat trainees zo zelfstandig kunnen trainen, waar en wanneer ze maar willen.

Een mogelijk probleem bij het gebruik van intelligente agenten in virtuele training is dat trainees het gedrag van de agenten niet altijd goed begrijpen. Er zijn bijvoorbeeld verschillende redenen waarom virtuele teamleden (gespeeld door intelligente agenten) de instructies van hun leider (gespeeld door de trainee) niet opvolgen. Het kan zijn dat ze zijn instructies niet goed hebben begrepen, maar het kan ook zijn dat ze hem expres niet gehoorzamen. Na het spelen van een scenario weet de trainee dan niet of hij duidelijker moet communiceren, overtuigender moet zijn of veiligere instructies moet geven. Dit probleem kan worden opgelost met virtuele agenten die hun eigen gedrag verklaren. Als een trainee zijn virtuele medespelers kan vragen waarom ze iets doen, krijgt hij de mogelijkheid om het gedrag van anderen en daarmee zijn eigen prestaties beter te begrijpen. Dit proefschrift gaat over het verklaren van agentgedrag in virtuele training en stelt een benadering voor waarin virtuele agenten automatisch verklaringen voor hun gedrag genereren.

Er bestaan reeds verschillende benaderingen voor het verklaren van agentgedrag in virtuele training. In de eerste benaderingen werden de acties van agenten vaak verklaard door fysieke eigenschappen van een agent, bijvoorbeeld zijn positie of gezondheid. Psychologisch onderzoek laat echter zien dat menselijk gedrag meestal wordt begrepen en verklaard in termen van mentale concepten zoals doelen, overtuigingen en intenties. In meer recente benaderingen voor het verklaren van agentgedrag in virtuele training bevatten verklaringen ook mentale eigenschappen van agenten, zoals hun doelen en hun intenties.

De huidige benaderingen voor het verklaren van agentgedrag (acties van agenten worden verklaard door mentale eigenschappen) bestaan allemaal uit onafhankelijke verklaringscomponenten. Dat betekent dat de component die verklaringen genereert op verschillende trainingssystemen toegepast kan worden. Elk trainingssysteem heeft zijn eigen manier om agentgedrag te representeren, bijvoorbeeld met een neuraal netwerk of met een verzameling van als-dan regels. Het komt zelden voor dat de doelen en intenties van een agent expliciet in een trainingssysteem gerepresenteerd zijn. Om toch verklaringen met mentale eigenschappen te genereren, moeten de ontwikkelaars van bestaande benaderingen de doelen en intenties voor de acties van een agent handmatig aan de verklaringscomponent toevoegen. Op die manier wordt agentgedrag dus twee keer gerepresenteerd, in het trainingssysteem en in de verklaringscomponent.

In dit proefschrift stellen we een benadering voor waarbij het gedrag van een agent slechts éénmaal gespecificeerd hoeft te worden. De benadering levert geen verklaringscomponent op die aan verschillende trainingssystemen kan worden gekoppeld. In plaats daarvan worden agenten en hun verklaringscapaciteiten juist gezamenlijk ontwikkeld. Door het gedrag van agenten in virtuele training zelf te representeren kunnen ontwikkelaars ervoor zorgen dat alle informatie die nodig is voor het verklaren van acties al in de gedragsrepresentaties van de agenten aanwezig is. Dit bespaart ontwikkelaars de moeite om agentgedrag nogmaals te representeren in een verklaringscomponent. In onze benadering maken we voor het representeren van agentgedrag gebruik van een BDI programmeertaal (*Belief Desire Intention*). Het gedrag van een BDI agent wordt gerepresenteerd door doelen, overtuigingen, plannen en intenties. De acties van een BDI agent worden bepaald door een redeneerproces met zijn mentale concepten. Op die manier kunnen de mentale concepten die verantwoordelijk zijn voor de generatie van een actie ook worden gebruikt om diezelfde actie te verklaren.

Dit proefschrift beschrijft hoe agenten die hun gedrag kunnen verklaren worden ontwikkeld in een BDI programmeertaal. Eerst moet er een doelhiërarchie worden gemaakt met bovenaan in de hiërarchie het hoofddoel van de agent, daaronder zijn subdoelen en helemaal beneden in de hiërarchie zijn acties. Met adoptiecondities kan worden gespecificeerd onder welke omstandigheden een (sub)doel of actie moet worden aangenomen. Daarna wordt de doelhiërarchie geïmplementeerd als een BDI agent. De doelen en adoptiecondities in de hiërarchie worden respectievelijk gerepresenteerd als doelen en overtuigingen in de agent. Ten slotte wordt de BDI agent uitgebreid met een verklaringsmodule. De verklaringsmodule slaat alle acties en redeneerstappen van de agent op en kan er vervolgens verklaringen voor genereren. We stellen een aantal verklaringsalgoritmes voor die verschillende types verklaringen genereren. Een actie kan bijvoorbeeld worden verklaard door het doel dat met de actie bereikt wordt, de omstandigheden die de aanleiding waren voor de actie, of de staat die bereikt wordt door de actie uit te voeren.

Aangezien onze benadering verschillende soorten verklaringen kan genereren, hebben we drie studies gedaan om uit te zoeken welke soorten als meest nuttig worden beschouwd voor het leren van virtuele training. De deelnemers in de drie studies waren instructeurs, beginners en experts en de domeinen waren brandbestrijding aan boord, brandbestrijding en koken. We hebben onder andere gevonden dat mensen over het algemeen een voorkeur hebben voor verklaringen met (1) een overtuiging over de omstandigheden die de aanleiding waren voor de actie en (2) het doel dat met de actie wordt bereikt. Op basis van de resultaten worden in dit proefschrift een aantal richtlijnen voor het verklaren van agentgedrag voorgesteld.

Het proefschrift beschrijft een tweede reeks van drie studies die de voorgestelde benadering valideren. Dit is gedaan door te onderzoeken hoe de verklaringen prestatie beïnvloeden. Twee van de studies hebben de effecten van verklaringen op het leren van virtuele training onderzocht. De domeinen van de virtuele trainingen in deze studies waren brandbestrijding aan boord en onderhandelen. Aangezien we geen effecten van virtuele training op leren hebben kunnen aantonen in deze studies zijn er ook geen effecten van verklaringen op leren gevonden. De deelnemers van de tweede studie gaven echter aan dat de verklaringen van de agent hun inzicht in de motivaties voor zijn gedrag vergrootte. In een derde studie hebben we de effecten van verklaringen op coördinatie in mens-agent teams onderzocht. We hebben geen effect van de verklaringen op teamprestatie gevonden, maar deelnemers begrepen het gedrag van de agent beter wanneer hij zijn gedrag verklaarde. Verder vonden deelnemers de hoeveelheid informatie die de agent hun verschafte beter wanneer hij zijn gedrag verklaren.

Uit een van de studies uit de eerste reeks bleek dat instructeurs vaak een voorkeur hebben voor verklaringen voor acties die voorspellingen over het gedrag van anderen bevatten. Om zulke verklaringen te kunnen genereren hebben we onze benadering uitgebreid door de agenten te voorzien van een *theory of mind*. Theory of mind is het vermogen om mentale concepten zoals overtuigingen en doelen aan anderen toe te schrijven en op basis daarvan voorspellingen te doen over hun gedrag. Met een theory of mind kunnen agenten verklaringen genereren in termen van doelen en overtuigingen die ze aan anderen toeschrijven. Filosofische literatuur beschrijft twee theorieën over theory of mind: theorie-theorie en simulatietheorie. We hebben deze twee theorieën met elkaar vergeleken wat betreft hun geschiktheid voor het modelleren van agenten met een theory of mind. Uit deze vergelijking bleek dat de verklaringen die op basis van beide theorieën gegenereerd werden even nuttig waren, maar dat het ontwikkelen van agentmodellen efficiënter is wanneer ze gebaseerd zijn op simulatietheorie. We hebben daarom een uitvoerbaar model voor BDI agenten met een theory of mind ontwikkeld dat gebaseerd is op simulatietheorie.

Het proefschrift eindigt met een discussie over de voor- en nadelen van onze benadering en verschaft een aantal suggesties voor verder onderzoek. De belangrijkste conclusies van dit proefschrift zijn dat (1) BDI agenten kunnen worden gebruikt voor het genereren van verklaringen over hun eigen acties in termen van doelen, overtuigingen en plannen, (2) verklaringen gegenereerd volgens onze benadering het begrip in het gedrag van een agent kunnen verhogen, (3) onze benadering met name geschikt is wanneer de agenten gebruikt worden voor verschillende trainingsscenarios.

Dankwoord

Vele mensen hebben mij direct of indirect geholpen bij het tot stand komen van dit proefschrift. Allereerst wil ik daarvan Rineke Verbrugge bedanken. Zij heeft me tijdens mijn studie kunstmatige intelligentie in Groningen (nog meer) enthousiast gemaakt over de wetenschap en haar aanmoedigingen om te gaan promoveren zijn belangrijk geweest bij mijn keuze daarvoor.

Vervolgens wil ik uiteraard graag mijn begeleiders bedanken. Karel is altijd enorm betrokken geweest bij dit project en hield daarbij veel rekening met wat ik er allemaal van vond. Hij heeft mij bij TNO vaak uitgenodigd voor overleggen en projectbijeenkomsten die mogelijk interessant voor me konden zijn. Dat heb ik zeer op prijs gesteld. John-Jules wil ik graag bedanken voor zijn aanstekelijke enthousiasme en brede wetenschappelijke interesse. Wanneer ik zelf soms twijfelde over een nieuw idee wist hij me ervan te overtuigen dat het de moeite waard was om ermee verder te gaan.

Naast Karel en John-Jules heb ik met verschillende anderen samengewerkt. Ik wil hier alle co-auteurs van publicaties in het kader van mijn promotieonderzoek bedanken: Annerieke, Catholijn, Frank, Jeff, Joost Broekens, Joost Westra, Koen, Matt, Mehdi, Michal, Paul, Tina, Willem en Willem-Paul. Met name de discussies met Joost Broekens vond ik zeer inspirerend.

De leescommissie, bestaande uit Jeff Bradshaw, Jaap van den Herik, Mark Neerincx, Rineke Verbrugge en Peter Werkhoven, wil ik bedanken voor het lezen van mijn proefschrift en de nuttige feedback.

Drie maanden van mijn promotieonderzoek heb ik doorgebracht op het Institute for Human and Machine Cognition in Pensacola, Florida. Jeff, Karel en Jurriaan wil ik bedanken voor hun hulp bij de tot standkoming van deze uitwisseling en Jeff en Matt dank ik voor de prettige samenwerking aldaar. ‘The Dutch’ en ‘The Italians’ hebben ervoor gezorgd dat deze tijd ook naast het werk erg leuk was.

Ik heb de afgelopen vier jaar mogen genieten van het hebben van twee werkplekken. Het ene deel van de week bracht ik door bij TNO in Soesterberg, waar ik veel interessante praktijkvoorbeelden en onderzoekstoepassingen heb gezien. Het andere deel van de week bracht ik door op de universiteit Utrecht, waar ik juist veel heb geleerd op theoretisch gebied. Ik heb deze combinatie als een grote luxe ervaren.

Van TNO wil ik alle collega’s uit de afdeling Training Innovations hartelijk bedanken voor de fijne tijd. Daarnaast dank ik de mensen uit de cognitieve modellen groep voor de leuke discussies en de gezelligheid: Jan, Jurriaan, Peter-Paul, Rob, Tijmen, Tina en Willem. Als laatste wil ik Annerieke en Marieke bedanken, met wie ik een kamer heb gedeeld en die ik nog veel beter heb mogen leren kennen.

Bij de universiteit Utrecht heb ik vele koffiepauzes, al dan niet zinnige discussies, raadsels, grappen, borrels, feestjes, conferentiebezoeken en weekendjes weg gedeeld met de mensen uit de IS-groep. Bas, Bob, Eric, Gennaro, Gerard, Henry, Joost Westra, Jan, Joost van Oijen, Liz, Max, Marieke, Mehdi, Michal, Nick, Nieske, Susan, Tom en alle anderen, bedankt hiervoor. In het bijzonder wil ik mijn kamergenoten Hado en Paolo bedanken die altijd wel even wilden kletsen als mijn concentratie het liet afweten.

De afgelopen jaren bestonden niet alleen uit werken. Ik wil mijn vrienden bedanken voor hun interesse en alle welkome afleiding in de vorm van etentjes, spelletjesavonden, wijntjes in de stad, yogalessen, hardlopen, uitgebreide skypesessies, weekendjes weg, vakanties en nog veel meer. Mijn familie, Jan, Marian en Karen, wil ik bedanken voor hun steun en warmte. Tot slot wil ik Chris bedanken voor zijn geduld en liefde bij de laatste loodjes.

SIKS Dissertation Series

1998

1998-01 **Johan van den Akker** (CWI), *DEGAS - An Active, Temporal Database of Autonomous Objects.*

1998-02 **Floris Wiesman** (UM), *Information Retrieval by Graphically Browsing Meta-Information.*

1998-03 **Ans Steuten** (TUD), *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective.*

1998-04 **Dennis Breuker** (UM), *Memory versus Search in Games.*

1998-05 **E.W. Oskamp** (RUL), *Computerondersteuning bij Straftoemeting.*

1999

1999-01 **Mark Sloof** (VU), *Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products.*

1999-02 **Rob Potharst** (EUR), *Classification using decision trees and neural nets.*

1999-03 **Don Beal** (UM), *The Nature of Minimax Search.*

1999-04 **Jacques Penders** (UM), *The practical Art of Moving Physical Objects.*

1999-05 **Aldo de Moor** (KUB), *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems.*

1999-06 **Niek J.E. Wijngaards** (VU), *Re-design of compositional systems.*

1999-07 **David Spelt** (UT), *Verification support for object database design.*

1999-08 **Jacques H.J. Lenting** (UM), *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.*

2000

2000-01 **Frank Niessink** (VU), *Perspectives on Improving Software Maintenance.*

2000-02 **Koen Holtman** (TUE), *Prototyping of CMS Storage Management.*

2000-03 **Carolien M.T. Metselaar** (UVA), *Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.*

2000-04 **Geert de Haan** (VU), *ETAG, A Formal Model of Competence Knowledge for User Interface Design.*

2000-05 **Ruud van der Pol** (UM), *Knowledge-based Query Formulation in Information Retrieval.*

2000-06 **Rogier van Eijk** (UU), *Programming Languages for Agent Communication.*

2000-07 **Niels Peek** (UU), *Decision-theoretic Planning of Clinical Patient Management.*

2000-08 **Veerle Coup** (EUR), *Sensitivity Analyis of Decision-Theoretic Networks.*

2000-09 **Florian Waas** (CWI), *Principles of Probabilistic Query Optimization.*

2000-10 **Niels Nes** (CWI), *Image Database Management System Design Considerations, Algorithms and Architecture.*

2000-11 **Jonas Karlsson** (CWI), *Scalable Distributed Data Structures for Database Management.*

2001

2001-01 **Silja Renooij** (UU), *Qualitative Approaches to Quantifying Probabilistic Networks.*

2001-02 **Koen Hindriks** (UU), *Agent Programming Languages: Programming with Mental Models.*

2001-03 **Maarten van Someren** (UvA), *Learning as problem solving.*

2001-04 **Evgueni Smirnov** (UM), *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets.*

2001-05 **Jacco van Ossenbruggen** (VU), *Processing Structured Hypermedia: A Matter of Style.*

2001-06 **Martijn van Welie** (VU), *Task-based User Interface Design.*

2001-07 **Bastiaan Schonhage** (VU), *Diva: Architectural Perspectives on Information Visualization.*

2001-08 **Pascal van Eck** (VU), *A Compositional Semantic Structure for Multi-Agent Systems Dynamics.*

2001-09 **Pieter Jan 't Hoen** (RUL), *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes.*

2001-10 **Maarten Sierhuis** (UvA), *Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design.*

2001-11 **Tom M. van Engers** (VUA), *Knowledge Management: The Role of Mental Models in Business Systems Design.*

2002

2002-01 **Nico Lassing** (VU), *Architecture-Level Modifiability Analysis.*

2002-02 **Roelof van Zwol** (UT), *Modelling and searching web-based document collections.*

2002-03 **Henk Ernst Blok** (UT), *Database Optimization Aspects for Information Retrieval.*

2002-04 **Juan Roberto Castelo Valdueza** (UU), *The Discrete Acyclic Digraph Markov Model in Data Mining.*

2002-05 **Radu Serban** (VU), *The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents.*

2002-06 **Laurens Mommers** (UL), *Applied legal epistemology; Building a knowledge-based ontology of the legal domain.*

2002-07 **Peter Boncz** (CWI), *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications.*

2002-08 **Jaap Gordijn** (VU), *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas.*

2002-09 **Willem-Jan van den Heuvel** (KUB), *Integrating Modern Business Applications with Objectified Legacy Systems.*

2002-10 **Brian Sheppard** (UM), *Towards Perfect Play of Scrabble.*

2002-11 **Wouter C.A. Wijngaards** (VU), *Agent Based Modelling of Dynamics: Biological and Organisational Applications.*

2002-12 **Albrecht Schmidt** (UVA), *Processing XML in Database Systems.*

2002-13 **Hongjing Wu** (TUE), *A Reference Architecture for Adaptive Hypermedia Applications.*

2002-14 **Wieke de Vries** (UU), *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems.*

2002-15 **Rik Eshuis** (UT), *Semantics and Verification of UML Activity Diagrams for Workflow Modelling.*

2002-16 **Pieter van Langen** (VU), *The Anatomy of Design: Foundations, Models and Applications.*

2002-17 **Stefan Manegold** (UVA), *Understanding, Modeling, and Improving Main-Memory Database Performance*.

2003

2003-01 **Heiner Stuckenschmidt** (VU), *Onotology-Based Information Sharing In Weakly Structured Environments*.

2003-02 **Jan Broersen** (VU), *Modal Action Logics for Reasoning About Reactive Systems*.

2003-03 **Martijn Schuemie** (TUD), *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*.

2003-04 **Milan Petkovic** (UT), *Content-Based Video Retrieval Supported by Database Technology*.

2003-05 **Jos Lehmann** (UVA), *Causation in Artificial Intelligence and Law - A modelling approach*.

2003-06 **Boris van Schooten** (UT), *Development and specification of virtual environments*.

2003-07 **Machiel Jansen** (UvA), *Formal Explorations of Knowledge Intensive Tasks*.

2003-08 **Yongping Ran** (UM), *Repair Based Scheduling*.

2003-09 **Rens Kortmann** (UM), *The resolution of visually guided behaviour*.

2003-10 **Andreas Lincke** (UvT), *Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture*.

2003-11 **Simon Keizer** (UT), *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*.

2003-12 **Roeland Ordelman** (UT), *Dutch speech recognition in multimedia information retrieval*.

2003-13 **Jeroen Donkers** (UM), *Nosce Hostem - Searching with Opponent Models*.

2003-14 **Stijn Hoppenbrouwers** (KUN), *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*.

2003-15 **Mathijs de Weerdt** (TUD), *Plan Merging in Multi-Agent Systems*.

2003-16 **Menzo Windhouwer** (CWI), *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses*.

2003-17 **David Jansen** (UT), *Extensions of Statecharts with Probability, Time, and Stochastic Timing*.

2003-18 **Levente Kocsis** (UM), *Learning Search Decisions*.

2004

2004-01 **Virginia Dignum** (UU), *A Model for Organizational Interaction: Based on Agents, Founded in Logic*.

2004-02 **Lai Xu** (UvT), *Monitoring Multi-party Contracts for E-business*.

2004-03 **Perry Groot** (VU), *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*.

2004-04 **Chris van Aart** (UVA), *Organizational Principles for Multi-Agent Architectures*.

2004-05 **Viara Popova** (EUR), *Knowledge discovery and monotonicity*.

2004-06 **Bart-Jan Hommes** (TUD), *The Evaluation of Business Process Modeling Techniques*.

2004-07 **Elise Boltjes** (UM), *Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes*.

2004-08 **Joop Verbeek** (UM), *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise*.

2004-09 **Martin Caminada** (VU), *For the Sake of the Argument; explorations into argument-based reasoning*.

2004-10 **Suzanne Kabel** (UVA), *Knowledge-rich indexing of learning-objects*.

2004-11 **Michel Klein** (VU), *Change Management for Distributed Ontologies*.

2004-12 **The Duy Bui** (UT), *Creating emotions and facial expressions for embodied agents.*

2004-13 **Wojciech Jamroga** (UT), *Using Multiple Models of Reality: On Agents who Know how to Play.*

2004-14 **Paul Harrenstein** (UU), *Logic in Conflict. Logical Explorations in Strategic Equilibrium.*

2004-15 **Arno Knobbe** (UU), *Multi-Relational Data Mining.*

2004-16 **Federico Divina** (VU), *Hybrid Genetic Relational Search for Inductive Learning.*

2004-17 **Mark Winands** (UM), *Informed Search in Complex Games.*

2004-18 **Vania Bessa Machado** (UvA), *Supporting the Construction of Qualitative Knowledge Models.*

2004-19 **Thijs Westerveld** (UT), *Using generative probabilistic models for multimedia retrieval.*

2004-20 **Madelon Evers** (Nyenrode), *Learning from Design: facilitating multidisciplinary design teams.*

2005

2005-01 **Floor Verdenius** (UVA), *Methodological Aspects of Designing Induction-Based Applications.*

2005-02 **Erik van der Werf** (UM), *AI techniques for the game of Go.*

2005-03 **Franc Grootjen** (RUN), *A Pragmatic Approach to the Conceptualisation of Language.*

2005-04 **Nirvana Meratnia** (UT), *Towards Database Support for Moving Object data.*

2005-05 **Gabriel Infante-Lopez** (UVA), *Two-Level Probabilistic Grammars for Natural Language Parsing.*

2005-06 **Pieter Spronck** (UM), *Adaptive Game AI.*

2005-07 **Flavius Frasincar** (TUE), *Hypermedia Presentation Generation for Semantic Web Information Systems.*

2005-08 **Richard Vdovjak** (TUE), *A Model-driven Approach for Building Distributed Ontology-based Web Applications.*

2005-09 **Jeen Broekstra** (VU), *Storage, Querying and Inferencing for Semantic Web Languages.*

2005-10 **Anders Bouwer** (UVA), *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments.*

2005-11 **Elth Ogston** (VU), *Agent Based Matchmaking and Clustering - A Decentralized Approach to Search.*

2005-12 **Csaba Boer** (EUR), *Distributed Simulation in Industry.*

2005-13 **Fred Hamburg** (UL), *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen.*

2005-14 **Borys Omelayenko** (VU), *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics.*

2005-15 **Tibor Bosse** (VU), *Analysis of the Dynamics of Cognitive Processes.*

2005-16 **Joris Graaumans** (UU), *Usability of XML Query Languages.*

2005-17 **Boris Shishkov** (TUD), *Software Specification Based on Re-usable Business Components.*

2005-18 **Danielle Sent** (UU), *Test-selection strategies for probabilistic networks.*

2005-19 **Michel van Dartel** (UM), *Situated Representation.*

2005-20 **Cristina Coteanu** (UL), *Cyber Consumer Law, State of the Art and Perspectives.*

2005-21 **Wijnand Derks** (UT), *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics.*

2006

2006-01 **Samuil Angelov** (TUE), *Foundations of B2B Electronic Contracting.*

2006-02 **Cristina Chisalita** (VU), *Contextual issues in the design and use of information technology in organizations.*

2006-03 **Noor Christoph** (UVA), *The role of metacognitive skills in learning to solve problems.*

2006-04 **Marta Sabou** (VU), *Building Web Service Ontologies*.

2006-05 **Cees Pierik** (UU), *Validation Techniques for Object-Oriented Proof Outlines*.

2006-06 **Ziv Baida** (VU), *Software-aided Service Bundling – Intelligent Methods & Tools for Graphical Service Modeling*.

2006-07 **Marko Smiljanic** (UT), *XML schema matching – balancing efficiency and effectiveness by means of clustering*.

2006-08 **Eelco Herder** (UT), *Forward, Back and Home Again – Analyzing User Behavior on the Web*.

2006-09 **Mohamed Wahdan** (UM), *Automatic Formulation of the Auditor's Opinion*.

2006-10 **Ronny Siebes** (VU), *Semantic Routing in Peer-to-Peer Systems*.

2006-11 **Joeri van Ruth** (UT), *Flattening Queries over Nested Data Types*.

2006-12 **Bert Bongers** (VU), *Interactivation – Towards an e-cology of people, our technological environment, and the arts*.

2006-13 **Henk-Jan Lebbink** (UU), *Dialogue and Decision Games for Information Exchanging Agents*.

2006-14 **Johan Hoorn** (VU), *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*.

2006-15 **Rainer Malik** (UU), *CONAN: Text Mining in the Biomedical Domain*.

2006-16 **Carsten Riggelsen** (UU), *Approximation Methods for Efficient Learning of Bayesian Networks*.

2006-17 **Stacey Nagata** (UU), *User Assistance for Multitasking with Interruptions on a Mobile Device*.

2006-18 **Valentin Zhizhkun** (UVA), *Graph transformation for Natural Language Processing*.

2006-19 **Birna van Riemsdijk** (UU), *Cognitive Agent Programming: A Semantic Approach*.

2006-20 **Marina Velikova** (UvT), *Monotone models for prediction in data mining*.

2006-21 **Bas van Gils** (RUN), *Aptness on the Web*.

2006-22 **Paul de Vrieze** (RUN), *Fundaments of Adaptive Personalisation*.

2006-23 **Ion Juvina** (UU), *Development of Cognitive Model for Navigating on the Web*.

2006-24 **Laura Hollink** (VU), *Semantic Annotation for Retrieval of Visual Resources*.

2006-25 **Madalina Drugan** (UU), *Conditional log-likelihood MDL and Evolutionary MCMC*.

2006-26 **Vojkan Mihajlovic** (UT), *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*.

2006-27 **Stefano Bocconi** (CWI), *Vox Populi: generating video documentaries from semantically annotated media repositories*.

2006-28 **Borkur Sigurbjornsson** (UVA), *Focused Information Access using XML Element Retrieval*.

2007

2007-01 **Kees Leune** (UvT), *Access Control and Service-Oriented Architectures*.

2007-02 **Wouter Teepe** (RUG), *Reconciling Information Exchange and Confidentiality: A Formal Approach*.

2007-03 **Peter Mika** (VU), *Social Networks and the Semantic Web*.

2007-04 **Jurriaan van Diggelen** (UU), *Achieving Semantic Interoperability in Multi-agent Systems: A Dialogue-based Approach*.

2007-05 **Bart Schermer** (UL), *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*.

2007-06 **Gilad Mishne** (UVA), *Applied Text Analytics for Blogs*.

2007-07 **Natasa Jovanovic** (UT), *To Who It May Concern - Addressee Identification in Face-to-Face Meetings*.

2007-08 **Mark Hoogendoorn** (VU), *Modeling of Change in Multi-Agent Organizations*.

2007-09 **David Mobach** (VU), *Agent-Based Mediated Service Negotiation.*

2007-10 **Huib Aldewereld** (UU), *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols.*

2007-11 **Natalia Stash** (TUE), *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System.*

2007-12 **Marcel van Gerven** (RUN), *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty.*

2007-13 **Rutger Rienks** (UT), *Meetings in Smart Environments; Implications of Progressing Technology.*

2007-14 **Niek Bergboer** (UM), *Context-Based Image Analysis.*

2007-15 **Joyca Lacroix** (UM), *NIM: a Situated Computational Memory Model.*

2007-16 **Davide Grossi** (UU), *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems.*

2007-17 **Theodore Charitos** (UU), *Reasoning with Dynamic Networks in Practice.*

2007-18 **Bart Orriens** (UvT), *On the development an management of adaptive business collaborations.*

2007-19 **David Levy** (UM), *Intimate relationships with artificial partners.*

2007-20 **Slinger Jansen** (UU), *Customer Configuration Updating in a Software Supply Network.*

2007-21 **Karianne Vermaas** (UU), *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005.*

2007-22 **Zlatko Zlatev** (UT), *Goal-oriented design of value and process models from patterns.*

2007-23 **Peter Barna** (TUE), *Specification of Application Logic in Web Information Systems.*

2007-24 **Georgina Ramrez Camps** (CWI), *Structural Features in XML Retrieval.*

2007-25 **Joost Schalken** (VU), *Empirical Investigations in Software Process Improvement.*

2008

2008-01 **Katalin Boer-Sorbn** (EUR), *Agent-Based Simulation of Financial Markets: A modular,continuous-time approach.*

2008-02 **Alexei Sharpanskykh** (VU), *On Computer-Aided Methods for Modeling and Analysis of Organizations.*

2008-03 **Vera Hollink** (UVA), *Optimizing hierarchical menus: a usage-based approach.*

2008-04 **Ander de Keijzer** (UT), *Management of Uncertain Data - towards unattended integration.*

2008-05 **Bela Mutschler** (UT), *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective.*

2008-06 **Arjen Hommersom** (RUN), *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective.*

2008-07 **Peter van Rosmalen** (OU), *Supporting the tutor in the design and support of adaptive e-learning.*

2008-08 **Janneke Bolt** (UU), *Bayesian Networks: Aspects of Approximate Inference.*

2008-09 **Christof van Nimwegen** (UU), *The paradox of the guided user: assistance can be counter-effective.*

2008-10 **Wauter Bosma** (UT), *Discourse oriented summarization.*

2008-11 **Vera Kartseva** (VU), *Designing Controls for Network Organizations: A Value-Based Approach.*

2008-12 **Jozsef Farkas** (RUN), *A Semiotically Oriented Cognitive Model of Knowledge Representation.*

2008-13 **Caterina Carraciolo** (UVA), *Topic Driven Access to Scientific Handbooks.*

2008-14 **Arthur van Bunningen** (UT), *Context-Aware Querying; Better Answers with Less Effort.*

2008-15 **Martijn van Otterlo** (UT), *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains.*

2008-16 **Henriette van Vugt** (VU), *Embodied agents from a user's perspective*.

2008-17 **Martin Op 't Land** (TUD), *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*.

2008-18 **Guido de Croon** (UM), *Adaptive Active Vision*.

2008-19 **Henning Rode** (UT), *From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search*.

2008-20 **Rex Arendsen** (UVA), *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven*.

2008-21 **Krisztian Balog** (UVA), *People Search in the Enterprise*.

2008-22 **Henk Koning** (UU), *Communication of IT-Architecture*.

2008-23 **Stefan Visscher** (UU), *Bayesian network models for the management of ventilator-associated pneumonia*.

2008-24 **Zharko Aleksovski** (VU), *Using background knowledge in ontology matching*.

2008-25 **Geert Jonker** (UU), *Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency*.

2008-26 **Marijn Huijbregts** (UT), *Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled*.

2008-27 **Hubert Vogten** (OU), *Design and Implementation Strategies for IMS Learning Design*.

2008-28 **Ildiko Flesch** (RUN), *On the Use of Independence Relations in Bayesian Networks*.

2008-29 **Dennis Reidsma** (UT), *Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans*.

2008-30 **Wouter van Atteveldt** (VU), *Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content*.

2008-31 **Loes Braun** (UM), *Pro-Active Medical Information Retrieval*.

2008-32 **Trung H. Bui** (UT), *Toward Affective Dialogue Management using Partially Observable Markov Decision Processes*.

2008-33 **Frank Terpstra** (UVA), *Scientific Workflow Design; theoretical and practical issues*.

2008-34 **Jeroen de Knijf** (UU), *Studies in Frequent Tree Mining*.

2008-35 **Ben Torben Nielsen** (UvT), *Dendritic morphologies: function shapes structure*.

2009

2009-01 **Rasa Jurgelenaite** (RUN), *Symmetric Causal Independence Models*.

2009-02 **Willem Robert van Hage** (VU), *Evaluating Ontology-Alignment Techniques*.

2009-03 **Hans Stol** (UvT), *A Framework for Evidence-based Policy Making Using IT*.

2009-04 **Josephine Nabukenya** (RUN), *Improving the Quality of Organisational Policy Making using Collaboration Engineering*.

2009-05 **Sietse Overbeek** (RUN), *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality*.

2009-06 **Muhammad Subianto** (UU), *Understanding Classification*.

2009-07 **Ronald Poppe** (UT), *Discriminative Vision-Based Recovery and Recognition of Human Motion*.

2009-08 **Volker Nannen** (VU), *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*.

2009-09 **Benjamin Kanagwa** (RUN), *Design, Discovery and Construction of Service-oriented Systems*.

2009-10 **Jan Wielemaker** (UVA), *Logic programming for knowledge-intensive interactive applications*.

2009-11 **Alexander Boer** (UVA), *Legal Theory, Sources of Law & the Semantic Web*.

2009-12 **Peter Massuthe** (TUE, Humboldt-Universitaet zu Berlin), *Perating Guidelines for Services.*

2009-13 **Steven de Jong** (UM), *Fairness in Multi-Agent Systems.*

2009-14 **Maksym Korotkiy** (VU), *From ontology-enabled services to service-enabled ontologies.* making ontologies work in e-science with ONTO-SOA

2009-15 **Rinke Hoekstra** (UVA), *Ontology Representation - Design Patterns and Ontologies that Make Sense.*

2009-16 **Fritz Reul** (UvT), *New Architectures in Computer Chess.*

2009-17 **Laurens van der Maaten** (UvT), *Feature Extraction from Visual Data.*

2009-18 **Fabian Groffen** (CWI), *Armada, An Evolving Database System.*

2009-19 **Valentin Robu** (CWI), *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets.*

2009-20 **Bob van der Vecht** (UU), *Adjustable Autonomy: Controling Influences on Decision Making.*

2009-21 **Stijn Vanderlooy** (UM), *Ranking and Reliable Classification.*

2009-22 **Pavel Serdyukov** (UT), *Search For Expertise: Going beyond direct evidence.*

2009-23 **Peter Hofgesang** (VU), *Modelling Web Usage in a Changing Environment.*

2009-24 **Annerieke Heuvelink** (VUA), *Cognitive Models for Training Simulations.*

2009-25 **Alex van Ballegooij** (CWI), *"RAM: Array Database Management through Relational Mapping".*

2009-26 **Fernando Koch** (UU), *An Agent-Based Model for the Development of Intelligent Mobile Services.*

2009-27 **Christian Glahn** (OU), *Contextual Support of Social Engagement and Reflection on the Web.*

2009-28 **Sander Evers** (UT), *Sensor Data Management with Probabilistic Models.*

2009-29 **Stanislav Pokraev** (UT), *Model-Driven Semantic Integration of Service-Oriented Applications.*

2009-30 **Marcin Zukowski** (CWI), *Balancing vectorized query execution with bandwidth-optimized storage.*

2009-31 **Sofiya Katrenko** (UVA), *A Closer Look at Learning Relations from Text.*

2009-32 **Rik Farenhorst and Remco de Boer** (VU), *Architectural Knowledge Management: Supporting Architects and Auditors.*

2009-33 **Khiet Truong** (UT), *How Does Real Affect Affect Affect Recognition In Speech?.*

2009-34 **Inge van de Weerd** (UU), *Advancing in Software Product Management: An Incremental Method Engineering Approach.*

2009-35 **Wouter Koelewijn** (UL), *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling.*

2009-36 **Marco Kalz** (OUN), *Placement Support for Learners in Learning Networks.*

2009-37 **Hendrik Drachsler** (OUN), *Navigation Support for Learners in Informal Learning Networks.*

2009-38 **Riina Vuorikari** (OU), *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context.*

2009-39 **Christian Stahl** (TUE, Humboldt-Universitaet zu Berlin), *Service Substitution – A Behavioral Approach Based on Petri Nets.*

2009-40 **Stephan Raaijmakers** (UvT), *Multinomial Language Learning: Investigations into the Geometry of Language.*

2009-41 **Igor Berezhnyy** (UvT), *Digital Analysis of Paintings.*

2009-42 **Toine Bogers** (UvT), *Recommender Systems for Social Bookmarking.*

2009-43 **Virginia Nunes Leal Franqueira** (UT), *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients.*

2009-44 **Roberto Santana Tapia** (UT), *Assessing Business-IT Alignment in Networked Organizations.*

2009-45 **Jilles Vreeken** (UU), *Making Pattern Mining Useful.*

2009-46 **Loredana Afanasiev** (UvA), *Querying XML: Benchmarks and Recursion.*

2010

2010-01 **Matthijs van Leeuwen** (UU), *Patterns that Matter.*

2010-02 **Ingo Wassink** (UT), *Work flows in Life Science.*

2010-03 **Joost Geurts** (CWI), *A Document Engineering Model and Processing Framework for Multimedia documents.*

2010-04 **Olga Kulyk** (UT), *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments.*

2010-05 **Claudia Hauff** (UT), *Predicting the Effectiveness of Queries and Retrieval Systems.*

2010-06 **Sander Bakkes** (UvT), *Rapid Adaptation of Video Game AI.*

2010-07 **Wim Fikkert** (UT), *A Gesture interaction at a Distance.*

2010-08 **Krzysztof Siewicz** (UL), *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments.*

2010-09 **Hugo Kielman** (UL), *Politiële gegevensverwerking en Privacy, Naar een effectieve waarborging.*

2010-10 **Rebecca Ong** (UL), *Mobile Communication and Protection of Children.*

2010-11 **Adriaan Ter Mors** (TUD), *The world according to MARP: Multi-Agent Route Planning.*

2010-12 **Susan van den Braak** (UU), *Sensemaking software for crime analysis.*

2010-13 **Gianluigi Folino** (RUN), *High Performance Data Mining using Bio-inspired techniques.*

2010-14 **Sander van Splunter** (VU), *Automated Web Service Reconfiguration.*

2010-15 **Lianne Bodenstaff** (UT), *Managing Dependency Relations in Inter-Organizational Models.*

2010-16 **Sicco Verwer** (TUD), *Efficient Identification of Timed Automata, theory and practice.*

2010-17 **Spyros Kotoulas** (VU), *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications.*

2010-18 **Charlotte Gerritsen** (VU), *Caught in the Act: Investigating Crime by Agent-Based Simulation.*

2010-19 **Henriette Cramer** (UvA), *People's Responses to Autonomous and Adaptive Systems.*

2010-20 **Ivo Swartjes** (UT), *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative.*

2010-21 **Harold van Heerde** (UT), *Privacy-aware data management by means of data degradation.*

2010-22 **Michiel Hildebrand** (CWI), *End-user Support for Access to Heterogeneous Linked Data.*

2010-23 **Bas Steunebrink** (UU), *The Logical Structure of Emotions.*

2010-24 **Dmytro Tykhonov** (), *Designing Generic and Efficient Negotiation Strategies.*

2010-25 **Zulfiqar Ali Memon** (VU), *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective.*

2010-26 **Ying Zhang** (CWI), *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines.*

2010-27 **Marten Voulon** (UL), *Automatisch contracteren.*

2010-28 **Arne Koopman** (UU), *Characteristic Relational Patterns.*

2010-29 **Stratos Idreos** (CWI), *Database Cracking: Towards Auto-tuning Database Kernels.*

2010-30 **Marieke van Erp** (UvT), *Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval.*

2010-31 **Victor de Boer** (UVA), *Ontology Enrichment from Heterogeneous Sources on the Web.*

2010-32 **Marcel Hiel** (UvT), *An Adaptive Service Oriented Architecture: Automatically solving Interoperabil-*

ity Problems.

2010-33 **Robin Aly** (UT), *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval.*

2010-34 **Teduh Dirgahayu** (UT), *Interaction Design in Service Compositions.*

2010-35 **Dolf Trieschnigg** (UT), *Proof of Concept: Concept-based Biomedical Information Retrieval.*

2010-36 **Jose Janssen** (OU), *Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification.*

2010-37 **Niels Lohmann** (TUE), *Correctness of services and their composition.*

2010-38 **Dirk Fahland** (TUE), *From Scenarios to components.*

2010-39 **Ghazanfar Farooq Siddiqui** (VU), *Integrative modeling of emotions in virtual agents.*

2010-40 **Mark van Assem** (VU), *Converting and Integrating Vocabularies for the Semantic Web.*

2010-41 **Guillaume Chaslot** (UM), *Monte-Carlo Tree Search.*

2010-42 **Sybren de Kinderen** (VU), *Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach.*

2010-43 **Peter van Kranenburg** (UU), *A Computational Approach to Content-Based Retrieval of Folk Song Melodies.*

2010-44 **Pieter Bellekens** (TUE), *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain.*

2010-45 **Vasilios Andrikopoulos** (UvT), *A theory and model for the evolution of software services.*

2010-46 **Vincent Pijpers** (VU), *e3alignment: Exploring Inter-Organizational Business-ICT Alignment.*

2010-47 **Chen Li** (UT), *Mining Process Model Variants: Challenges, Techniques, Examples.*

2010-48 **Milan Lovric** (EUR), *Behavioral Finance and Agent-Based Artificial Markets.*

2010-49 **Jahn-Takeshi Saito** (UM), *Solving difficult game positions.*

2010-50 **Bouke Huurnink** (UVA), *Search in Audiovisual Broadcast Archives.*

2010-51 **Alia Khairia Amin** (CWI), *Understanding and supporting information seeking tasks in multiple sources.*

2010-52 **Peter-Paul van Maanen** (VU), *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention.*

2010-53 **Edgar Meij** (UVA), *Combining Concepts and Language Models for Information Access.*

2011

2011-01 **Botond Cseke** (RUN), *Variational Algorithms for Bayesian Inference in Latent Gaussian Models.*

2011-02 **Nick Tinnemeier** (UU), *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language.*

2011-03 **Jan Martijn van der Werf** (TUE), *Compositional Design and Verification of Component-Based Information Systems.*

2011-04 **Hado van Hasselt** (UU), *Insights in Reinforcement Learning - Formal analysis and empirical evaluation of temporal-difference learning algorithms.*

2011-05 **Base van der Raadt** (VU), *Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.*

2011-06 **Yiwen Wang** (TUE), *Semantically-Enhanced Recommendations in Cultural Heritage.*

2011-07 **Yujia Cao** (UT), *Multimodal Information Presentation for High Load Human Computer Interaction.*

2011-08 **Nieske Vergunst** (UU), *BDI-based Generation of Robust Task-Oriented Dialogues.*

2011-09 **Tim de Jong** (OU), *Contextualised Mobile Media for Learning.*

2011-10 **Bart Bogaert** (UvT), *Cloud Content Contention.*

2011-11 **Dhaval Vyas** (UT), *Designing for Awareness: An Experience-focused HCI Perspective.*

2011-12 **Carmen Bratosin** (TUE), *Grid Architecture for Distributed Process Mining.*

2011-13 **Xiaoyu Mao** (UvT), *Airport under Control; Multiagent Scheduling for Airport Ground Handling.*

2011-14 **Milan Lovric** (EUR), *Behavioral Finance and Agent-Based Artificial Markets.*

2011-15 **Marijn Koolen** (UVA), *The Meaning of Structure: the Value of Link Evidence for Information Retrieval.*

2011-16 **Maarten Schadd** (UM), *Selective Search in Games of Different Complexity.*

2011-17 **Jiyin He** (UVA), *Exploring Topic Structure: Coherence, Diversity and Relatedness.*

2011-18 **Mark Ponsen** (UM), *Strategic Decision-Making in complex games.*

2011-19 **Ellen Rusman** (OU), *The Mind's Eye on Personal Profiles.*

2011-20 **Qing Gu** (VU), *Guiding service-oriented software engineering - A view-based approach.*

2011-21 **Linda Terlouw** (TUD), *Modularization and Specification of Service-Oriented Systems.*

2011-22 **Junte Zhang** (UVA), *System Evaluation of Archival Description and Access.*

2011-23 **Wouter Weerkamp** (UVA), *Finding People and their Utterances in Social Media.*

2011-24 **Herwin van Welbergen** (UT), *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior.*

2011-25 **Syed Waqar ul Qounain Jaffry** (VU), *Analysis and Validation of Models for Trust Dynamics.*

2011-26 **Matthijs Aart Pontier** (VU), *Virtual Agents for Human Communication.*

2011-27 **Aniel Bhulai** (VU), *Dynamic website optimization through autonomous management of design patterns.*

2011-28 **Rianne Kaptein** (UVA), *Effective Focused Retrieval by Exploiting Query Context and Document Structure.*

2011-29 **Faisal Kamiran** (TUE), *Discrimination-aware Classification.*

2011-30 **Egon van den Broek** (UT), *Affective Signal Processing (ASP): Unraveling the mystery of emotions.*

2011-31 **Ludo Waltman** (EUR), *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality.*

2011-32 **Nees Jan van Eck** (EUR), *Methodological Advances in Bibliometric Mapping of Science.*

2011-33 **Tom van der Weide** (UU), *Arguing to Motivate Decisions.*

2011-34 **Paolo Turrini** (UU), *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations.*

2011-35 **Maaike Harbers** (UU), *Explaining Agent Behavior in Virtual Training.*