A Methodology for Developing Self-Explaining Agents for Virtual Training*

Maaike Harbers
Utrecht University
P.O. Box 80.089, Utrecht
The Netherlands
maaike@cs.uu.nl

Karel van den Bosch
TNO Defence, Security & Safety
P.O. Box 23, Soesterberg
The Netherlands
karel.vandenbosch@tno.nl

John-Jules Meyer
Utrecht University
P.O. Box 80.089, Utrecht
The Netherlands
jj@cs.uu.nl

1. INTRODUCTION

Virtual training systems are used to train people for complex, dynamic tasks in which fast decision making is required, e.g. those in command in a crisis, military mission or fire incident. During a training session, trainees have to fulfill a task or mission for which they have to interact with other players, such as team-members, opponents, or colleagues from other domains. By using intelligent agents to generate the behavior of these virtual players, trainees can train at any place and time, reducing costs.

Typical mistakes of trainees involve situations in which a trainee makes false assumptions about other agents' knowledge or intentions. In cognitive science, incorrect attribution knowledge and intentions to others is a well described phenomenon, e.g. the tendency to ascribe one's own knowledge to others [6], or limits on the use of theory of mind, i.e. ascribing mental states to others, in practice [4]. Self-explaining agents can make trainees aware of their (possibly) false assumptions about others, namely by explaining the reasons for their actions after a training session is over. As humans usually explain their own and others' behavior in terms of beliefs, desires and other mental contents [3], the agents' explanations should use similar concepts. With the explanations, trainees obtain better insight in the events in the training session, and typical mistakes can be prevented in the future.

In this paper, we propose a methodology for developing self-explaining agents in virtual training systems. The methodology involves four steps: studying the actions the agent must be able to execute, constructing its task hierarchy, implementating the agent in a BDI-based agent programming language, and adding explanation facilities to the implementation. Although explanation facilities are added to the agent only in the end, the methods used in the previous steps contribute to easily do so. Due to space limitations we will only give a general outline of the proposed methodology.

*This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

Cite as: A Methodology for Developing Self-Explaining Agents for Virtual Training, Maaike Harbers, Karel van den Bosch and John-Jules Meyer, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

2. FROM ANALYSIS TO AGENT

The **first** step in developing self-explaining agents is to collect information about the required behavior and capacities of the agents from domain experts, e.g. the responsibilities of a fire-fighter or an operator. The information gathering is guided by the training scenario(s) in which the agents will act as the scenarios determine the scope of situations in which the agents might arrive, and thus which actions they might need to execute.

Second, with required capacities of an agent in a scenario, a task hierarchy can be constructed. Though writing a training scenario lays down the possible *observable* actions of an agent, it does not specify the *unobservable* processes leading to select those particular actions. We believe the agents' generation of behavior should be connected to behavior explanation. The deliberation steps that are taken to generate an action can also be used best to explain that action, and when these deliberation steps are understandable, the explanations should be as well. So while designing an agent with explanation capabilities, the unobservable internal processes should be meaningful. In cognitive psychology, cognitive task analysis is a well established technique [8] to connect observable behavior to internal cognitive processes by the systematical decomposition of goals or tasks.

The result of a cognitive task analysis can be represented as a hierarchical task network (HTN). We define a task hierarchy H in state S as a number of tasks which are related to each other by task-subtask relations. A task is defined as *T(Rel,[(T1,C1),...,(Tn,Cn)])*, where task T is decomposed into n subtasks $[T1,..,Tn]$, *Rel* denotes the relation of T with its subtasks, and *C1..n* denote the conditions under which a subtask is adopted. A relation denotes for example whether one, some or all subtasks have to be achieved in order to achieve task T. Subtasks can in turn be decomposed into subtasks, etc. Subtasks that are not decomposed are primitive tasks and can be achieved by executing a single action in the environment.

The **third** step involves the implementation of the agent. The agent programming language to be used should fulfill four requirements: the agent's reasoning elements (beliefs and goals) should be represented explicitly, the operationalization of these elements should be available as well, the agent should be able to introspect, and the agent needs to have a memory. We have chosen to use the BDI-based agent programming language 2APL [1] for the implementation of self-explaining agents. 2APL connects declarative aspects like beliefs and goals to plans and actions, following from

the interaction between beliefs and goals. Introspection is also possible in 2APL: the agent can perform checks on its belief and goal bases. A 2APL agent typically has no memory of its past states, processes and actions; these are only implicitly present in the interpreter. However, the agent's belief base can be considered as a memory, and a log about past states, processes and actions can be created in its belief base.

To implement a self-explaining agent, its task hierarchy representation needs to be translated to a BDI-based agent program. The most important similarity between task hierarchies and BDI models is that both reduce high-level entities into lower-level ones (see also [7]). The state, main task, subtasks and primitive tasks in a task hierarchy respond to beliefs, a main goal, sub-goals and actions in a BDI agent, respectively. Each task *T(Rel,[(T1,C1),...,(Tn,Cn)])* can be translated to the following general 2APL code.

```
Task and not Task1 <- C1 | adopt(Task1)
...
Task and not Taskn <- Cn | adopt(Taskn)
```

The first part of both rules *Task and not Taski* is a check on the agent's goal base, the second part *Ci* is a check on the agent's belief base, and the last part *adopt(Taski)* is an action. In this example the actions involve the adoption of subgoals, but in case of primitive actions, the actions are executed in the agent's environment. For each relation *Rel* of a task with its subtasks the specific code slightly varies.

The **fourth** and last step is to add explanation capabilities to the agent. In 2APL, the information required for explanations is present in the program and the interpreter, but not available to the agent for explanation at a later moment in time. However, in order to ensure that the agent's goals are dropped at the right time, updates about executed actions are made. Besides these updates about executed actions, the agents requires beliefs about the task hierarchy to know for which goals actions were performed. Therefore, for each task the agent has a belief of the type **Task(T,[T1,..,Tn])**, where T1 to Tn are T's subtasks. With these beliefs, the agents has knowledge about the complete task structure, and an explanation of the action can be created.

An extensive explanation could provide all tasks/goals. However, the complete trace of tasks/goals responsible for one action might provide too much and irrelevant information; especially in bigger agent models it is crucial to provide a selection of goals in the explanation. Such a selection consists of tasks either with a higher or a lower position in the hierarchy, yielding more abstract or specific explanations, respectively. More advanced explanation facilities could be interactive. For instance, the self-explaining agent starts with providing an abstract explanation, but if the trainee asks for extra information, more specific goals are provided.

3. DISCUSSION

In this paper, we have introduced a methodology for developing self-explaining agents in virtual training systems, which explain their behavior in terms of mental concepts. Current approaches of explanation in artificial intelligence do not provide such explanations. Expert system explanations usually provide traces of the steps behind a diagnose or advice, and often also their justifications [10]. There are a few accounts of self-explaining agents in virtual training systems [5, 9, 2], but these do not provide information about the actual goals behind an agent's actions. Unlike most automated explanation facilities in training systems, e.g. intelligent tutoring systems, explanations of self-explaining agents do not provide direct feedback on the trainees' performances. Instead, explanations serve to increase the trainees' awareness of other agents' perspectives, i.e. what they think and why they behave as they do.

Single phenomena and processes can be explained in many different ways, but providing complete explanations is neither possible, nor desired [3]. By choosing for a BDI-based approach, the scope of possible explanations is strongly restricted. Actions are only explained in terms of beliefs and goals. However, task hierarchies as we proposed can infinitely be extended and thus the implementations as well. Though the scenario in which an agent acts restricts the scope of its possible actions, some selection on the information provided in an explanation might increase its effectiveness. For example, abstract explanations are given by just providing goals higher in the task hierarchy, and more specific explanations only consist of elements lower in the hierarchy. In future research can be investigated whether there is a general desired abstraction level of explanations, and experimented with interactive explanation facilities.

4. REFERENCES

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

[2] D. Gomboc, S. Solomon, M. G. Core, H. C. Lane, and M. van Lent. Design recommendations to support automated explanation and tutoring. In *Proc. of the 14th Conf. on Behavior Representation in Modeling and Simulation*, Universal City, CA., 2005.

[3] F. Keil. Explanation and understanding. *Annual Reviews Psychology*, 57:227–254, 2006.

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

[5] W. Lewis Johnson. Agents that learn to explain themselves. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence*, pages 1257–1263, 1994.

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

[7] S. Sardina, L. De Silva, and L. Padgham. Hierarchical planning in bdi agent programming languages: A formal approach. In *Proc. of AAMAS 2006*. ACM Press, 2006.

[8] J. Schraagen, S. Chipman, and V. Shalin, editors. *Cognitive Task Analysis*. Lawrence Erlbaum Associates, Mahway, New Jersey, 2000.

[9] M. Van Lent, W. Fisher, and M. Mancuso. An explainable artificial intelligence system for small-unit tactical behavior. In *Proc. of IAAA 2004*, Menlo Park, CA, 2004. AAAI Press.

[10] R. Ye and P. Johnson. The impact of explanation facilities on user acceptance of expert systems advice. *Mis Quarterly*, 19(2):157–172, 1995.