Learning to Notice: Adaptive Models of Human Operators

Emma Norling
Computer Science and Software Engineering
The University of Melbourne, Vic 3010
Australia
ejn@cs.mu.oz.au

ABSTRACT

Agent-based technologies have been used for a number of years to model human operators in complex simulated environments. The BDI agent framework has proven to be particularly suited to this sort of modelling, due to its "natural" composition of beliefs, goals and plans. However one of the weaknesses of the BDI agent model, and many other human operator models (agent-based or otherwise), is its inability to support agent learning. Human operators naturally adapt their behaviour over time, particularly to avoid repeating mistakes. This paper introduces an enhancement to the BDI framework which is based on a descriptive psychological model of decision making called "recognition-primed decision making." This enhancement allows the development of agents that adapt their behaviour in real-time, in the same manner as a person would, providing more realistic human operator models.

Keywords

adaptive agents, human operator modelling, recognition-primed decision making, BDI agents

1. INTRODUCTION

Agent-based technologies are increasingly being used to model human operators in complex simulated environments, such as synthetic theatres of war, disaster recovery simulations, and computer games. Agents in these applications are generally provided with extensive information allowing them to interpret data, plan courses of action, and carry out these plans. In the military context, this information is often taken from standard operating procedure (SOP) manuals, or it is elicited from the operators who are being modelled. It would be infeasible to design agents that could learn from scratch how to operate in these environments, just as it would be to expect a person to do so without any background knowledge or training. In these domains we are interested in modelling experienced operators, who at the very least have had training in procedures, if not real-world experience. A characteristic of these operators is their ability to adapt, particularly when things go wrong. If a plan or procedure from the SOPs fails, the operator will first try recover from it, and then he/she will usually try to analyse *why* it failed, so as to avoid repeating the mistake. He/she learns to notice the subtle cues which mean that the plan will be unsuitable. It is this characteristic of human behaviour that is being addressed in the work described here.

One of the difficulties in trying to develop these sorts of agents is that in order to observe these aspects of human behaviour, we need a rich environment, with a complexity of variables approaching the real-world scenarios described above. The current application for this work is agents that can play the deathmatch version of Quake II. This application has been chosen because the game server itself provides a rich simulated environment, and there is also significant public domain knowledge about the workings of the server and network protocol (including the SoarBot project at the University of Michigan AI laboratory [9]). The implementation of these bots is still in preliminary stages, so this paper describes the system that is being developed, discussing some of the issues that have arisen. The paper concludes with some discussion of future directions for this work.

1.1 Background

The BDI (belief-desire-intention) architecture supports goal-directed behaviour, allowing a combination of reactive behaviour and long-term planning. The model stems from the folk psychological principle that rational agents will do what they believe is in their interests. BDI agents have been used by the Defence Science and Technology Organisation (DSTO) in Australia to model human operators in complex environments (e.g. air combat) for about ten years [3]. Initial applications were in simulations for operations analysis, but recently there has been a shift towards interactive simulations, with agents taking the roles of team members or opponents of the human operators. The interactions between real people and these agents have highlighted an issue raised in Pew and Mavor's review of human modelling in the military domain [6]: agents that worked very well in the operations analysis applications are less satisfactory in these interactive environments. Users are quick to discover the unexpected behaviours resulting from the failure to include many influences on human behaviour in the model, such as emotions, adaptability, fatigue, etc. For example, if you were competing against a real person and you discovered a weakness in their tactics, you would expect them to adjust their

tactics, rather than allowing you to repeatedly exploit that weakness. Similarly, if a rescue coordinator used a particular plan to try to rescue a victim from an earthquake-damaged building, only to have the building collapse and kill the victim, you would expect them to try something different at the next building. Unfortunately agents in existing systems do not adapt in this way. The same problem is observed with many of the computer generated characters and bots that are encountered in computer games. When the simulations are to be used for training, this sort of behaviour can be detrimental, leading to false training that does not apply to real world situations. We want agents that can adapt their behaviour in real time, during the course of a simulation, in a way that leads to human-like behaviour patterns.

How then can we design agents that adapt in the way that people do? The field of research known as "naturalistic decision making" (NDM) in psychology gives some direction. Researchers in NDM have developed several descriptive models of decision making, of which one – recognition-primed decision making – is particularly relevant to the sorts of environments and situations being simulated. In this model, the emphasis is on situation recognition: once the situation is recognised, the course of action is (usually) obvious. The decision maker learns to identify subtle differences between similar situations, giving him/her cues as to which course of action to take. By integrating this model of decision making into the BDI framework, we hope to produce agents that behave more like the human operators that they model.

This paper presents a method of implementing this type of adaptability, in a way that is feasible in the complex domains in which we are interested. The complexity of these types of domains, in addition to the fact that we want real-time adaptability, means that many traditional learning algorithms are unsuitable (as discussed by Riley and Veloso [8]). The method proposed here complements the folk psychological approach of BDI, using a set of pre-defined "cues" to be considered when adapting.

2. RECOGNITION-PRIMED DECISION MAKING

Researchers in naturalistic decision making are concerned with the way people make decisions in their natural work environments. They are primarily interested in people with some degree of expertise in their field, and in fields which have complex factors that affect performance, such as ill-structured problems, dynamic environments, time pressure, etc. Studies have shown that experienced operators in these sorts of environments do not apply classical decision making theories (i.e. those based around rational choice theory), even when they are trained to do so ([4], p. 99). Several alternative models of decision making have been proposed by researchers in this field, which are complementary rather than contradictory – different decision making strategies are used in different situations. Klein's recognition-primed decision making (RPD) model is the best known, and also the most applicable – studies of operators in various fields have shown that in some cases over 95% of decisions follow this model ([4], table 7.2), with a strong correlation between the experience of the operators and the degree to which they use RPD.

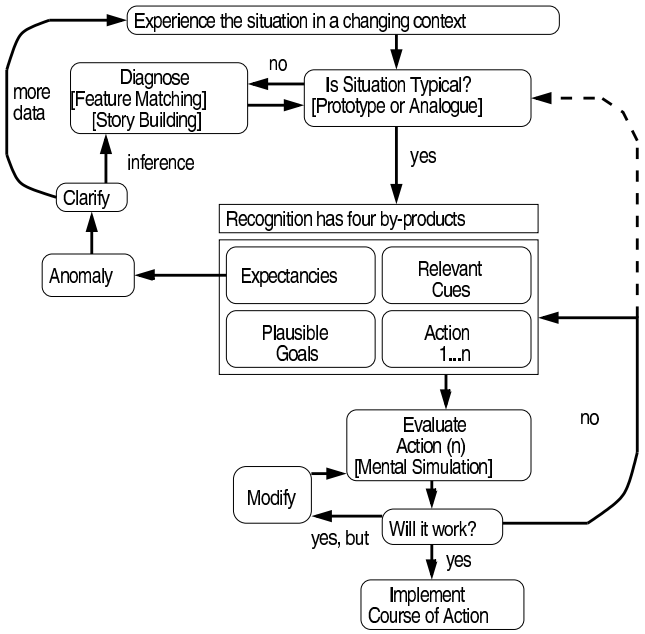

Figure 1: The recognition-primed decision model (Figure 7.1 from "Sources of Power" by G. Klein 1998)

The important thing to note about this model is that the focus is on situation assessment – the operator learns which cues distinguish one situation from another. Once the decision maker has recognised the situation, there are four by-products:

- he/she expects certain things to occur but not others;
- he/she pays attention to certain cues to support the diagnosis;
- there is some understanding of what goals are plausible to achieve; and
- there are certain actions which are likely to succeed.

The decision maker then selects and implements a course of action (i.e. a plan). Once a plan has been selected, the the situation is monitored to make sure it remains as expected, and if not, alternate plans may need to be considered, but otherwise the decision maker does not consider other options. The RPD model postulates that the decision maker will usually select a plan "automatically" once the situation is recognised, and that that plan is likely to be one that was previously successful in that situation.

3. BDI AGENTS FOR MODELLING HUMAN OPERATORS

BDI agents are extremely useful for modelling human operators because they use the same concepts that we naturally use to describe our actions and reasons for actions. The people being modelled can easily express their knowledge in these terms, and any pre-recorded information, such as tactical manuals, standard operating procedures, etc, also

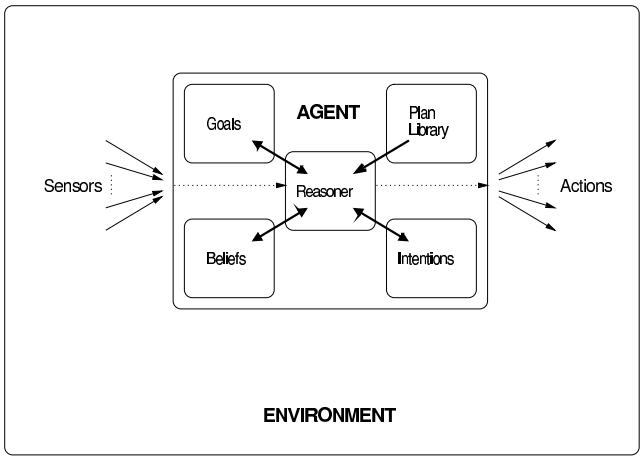

Figure 2: The structure of a BDI agent

tends to be worded in this language. ("Plan X is a way of achieving Y when the world is in state Z. It consists of doing...") This facilitates building models, and makes the completed models relatively easy for a domain expert to understand.

When a goal (desire) is posted, the agent searches the plan library for applicable plans (each plan has a *context* of applicability) and selects one of them, which becomes an *intention*. When there is more than one applicable plan, the agent must use some strategy to decide which one to use. Theoretically, the agent uses rational choice theory to select the "best" one [7]. In practical situations, particularly for the complex environments we are using, calculating the "value" of a plan, in order to make judgements about the "best" plan, can be difficult. Implementations of the BDI architecture, such as JACK [1], use simpler strategies, such as selecting the first applicable plan.

The BDI framework also deals with *plan failure*. If it becomes impossible to continue with an intention, the agent will try to find another plan to achieve the goal. This means that the agent can recover from plan failure (assuming there are suitable plans), but with either of the plan selection strategies mentioned above, it will make the same mistake the next time it is in the same situation.

Plan learning is not supported in the BDI framework. However in the sorts of applications that are being considered, this is not unreasonable. We are trying to model domain experts, who already have learned how to operate in their environment. In extreme circumstances they may improvise new plans to achieve their goals (or save their skins), but generally they draw upon known plans, from experience and SOPs. The operators learn when to choose one particular plan to achieve a goal over another, rather than learning new courses of action.

One of the advantages of using the BDI framework for modelling human operators is that the domain experts naturally express their knowledge in a way that can map to the framework. Consequently a plan library can be built up relatively easily, and domain experts can easily understand (and correct if necessary) the plans in the library. However one problem that is commonly encountered is that people tend to under-specify the context in which a plan is applicable, specifying the broad circumstances, but leaving many particulars implicit. When pressed as to why they chose a particular course of action, many skilled operators will claim that they "just knew," or that it was intuitive. In reality, they notice subtle cues, almost subconsciously, which guide their choice [4]. It can be difficult to define every one of these cues for each plan. The method described here allows the agent to refine plan context, learning the subtle differences through trial and error the way a person would.

4. ADAPTIVE MODELS OF HUMAN OPERATORS

Riley and Veloso [8] propose a method of abstracting useful features in a complex domain – the RoboCup soccer simulator. It makes use of certain attributes of the domain, such as the field, knowledge about players and their interactions with the ball, to build features that can be used to classify opponents (and in turn adapt behaviour). The method is suited to the particular application of the RoboCup soccer simulation league and could probably be easily adapted to other field-based ball games. In contrast to this approach, the work described in this paper aims to allow a similar sort of adaptability, but in a domain-independent manner.

One of the difficulties of dealing with a complex domain is the richness of the feature space – the myriad of things that an agent has beliefs about – that has to be considered in order to decide why a plan has failed. The agent may have hundreds of beliefs, but usually only a small subset of them will be relevant to any particular plan. In addition, these beliefs are usually stored as "raw" data, whereas it would be more natural to refer to abstractions of this data in the plan context. For example, if a Quake player had no armour and only 25% health, he/she might abstract that to "Critical danger!" At the same time, reasonable armour but 5% health and no weapon might lead to the same "Critical danger!" abstraction, where they would form the intention to take evasive action. These sort of abstractions are easy for the player to make, given their background knowledge, but difficult for the agent, unless it was supplied with that sort of knowledge.

What is proposed is that the agent builder should supply the agent with a set of these types of abstractions – predefined cues that could indicate the reason for plan failure. A cue will be a condition that is either true or false. It could be a simple test, such as checking whether a single value is above or below a threshold, or it could be a complex combination of beliefs, such as the "Critical danger!" cue described above. Like beliefs, plans and goals, cues correspond to something a domain expert will naturally refer to – for example, when answering the question "What sort of things could affect the outcome?" With the builder supplying the domain-specific knowledge in this way, the agent framework can then apply a domain independent learning mechanism to learn the reasons for plan failure and adjust the plan contexts accordingly.

When a plan fails, the agent will refer to its set of cues. Rather than considering all possible combinations of its cur-

rent beliefs as causes of the failure, the agent will only need to consider these cues, which will be a significantly smaller search space. At the same time, these cues supply the agent with the abstractions that it needs. With the search space reduced thus to a set of boolean tests, this becomes a fairly standard learning problem. For example, it has similarities to the mistake-driven learning in text categorisation described by Dagan *et al*[2].

An interesting question to consider is the matter of what happens when none of the cues account for plan failure. The hypothesis is that all but the most novel of situations should be accounted for by the cues. If this is the case, then a failure of the cues should mean one of two things:

1. The agent builder did not extract all the cues from the domain expert. In this case, the domain expert should be able to look at the failure (or set of failures) in question and be able to describe why it failed; or
2. A human operator would not have known what to do in this case either. The human operator may have been able to construct a new plan to deal with the situation, but these agents are not designed to do this.

4.1 Implementation

The initial testbed for these ideas is agents that can play the deathmatch version of Quake II. Quake II operates in client-server mode, with a game engine that clients (i.e. players, human or agent) can connect to, both locally and via a network. The server does not send low level graphics to the clients, but labelled information encoded in a binary stream. Clients perform all graphical rendering locally. This data stream, once decoded, provides all the information about the world that an agent needs to "see," and as mentioned previously, much information about this data stream is available in the public domain.

The agents are being implemented using JACK Intelligent Agents [1], a Java-based software package for building multi-agent systems. JACK provides a BDI framework which can be filled in and extended with Java code (or any code which can interface to Java). Unfortunately the time taken to write the interface between the server and agent was vastly underestimated. At the time of writing this paper, the agent can connect to the server, recognise other players and objects, and move around, but it does not have an intelligent set of plans – this is very much work in progress.

The aim is to have an agent, or bot, that can play the deathmatch version of Quake II, and adapt its behaviour in a similar way to a person. The factors that will influence its behaviour are varied, including things such as: the bot's armour and weaponry; its knowledge of the other players' tendencies and power of their weapons; the bot's current health; etc.

5. DISCUSSION

A human operator's ability to learn to notice the cues that make one course of action preferable to another, and adapt their behaviour accordingly, needs to be captured if we wish to build realistic models of these operators. The extension to the BDI architecture described here will provide a domain-independent framework to support this. Clearly there is still much work to be done in this project, and it is likely that further unforeseen issues will arise as it progresses, but already some have come to light. This includes the question of what happens if an agent is unable to determine the cause of plan failure, as discussed previously.

A key issue is how this work will be evaluated: it is not enough to see whether the agent does change its behaviour, or even if it improves its game play. The question that needs to be answered is whether its adaptive behaviour is more human-like. Determining the answer to this question will require domain experts to observe the agent's behaviour and judge its realism. Fortunately there is a plentiful supply of experienced Quake players in any Computer Science department! Laird and Duchi provide some discussion of how this sort of evaluation might be performed [5].

In addition, as briefly mentioned previously, there are other aspects of human performance which are not explicitly accounted for in the BDI architecture, such as emotions. The work here forms part of a larger project which is investigating human modelling in complex simulations, and will be integrated with other results.

6. ACKNOWLEDGEMENTS

Support for this work was provided by a Strategic Partners with Industry Research and Technology (SPIRT) award for the Australian Research Council and Agent Oriented Software Pty Ltd. The author would like to thank Liz Sonenberg and Peter Wallis for their invaluable comments on drafts of this paper.

7. REFERENCES

[1] Agent Oriented Software Pty. Ltd. JACK Intelligent Agents. http://agent-software.com.au/jack.html.

[2] I. Dagan, Y. Kerov, and D. Roth. Mistake-driven learning in text categorization. In C. Cardie and R. Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 55–63. Association for Computational Linguistics, Somerset, New Jersey, 1997.

[3] S. Goss and G. Murray. Artificial intelligence applications in aircraft systems. Technical Report DSTO-RR-0071, Defence Science and Technology Organisation, 1996. http://www.dsto.defence.gov.au/corporate/reports/DSTO-RR-0071.pdf.

[4] G. Klein. *Sources of Power*. The MIT Press, 1998.

[5] J. E. Laird and J. C. Duchi. Creating human-like synthetic characters with multiple skill levels: A case study using the soar quakebot. In M. Freed, editor, *Simulating Human Agents: Papers from the 2000 Fall Symposium*, North Falmouth, MA, November 2000.

[6] R. W. Pew and A. S. Mavor, editors. *Modelling Human and Organizational Behavior: Application to Military Simulations*. National Academy Press, 1998.

[7] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In V. Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*. MIT Press, 1995.

[8] P. Riley and M. Veloso. On behavior acquisition and classification in adversarial environments. In *Proceedings of the Agents-00/ECML-00 Joint Workshop on Learning Agents*, Barcelona, Spain, June 2000.

[9] University of Michigan Artificial Intelligence Laboratory. SoarBot Project Overview. http://ai.eecs.umich.edu/~soarbot/.