User Understanding of Cognitive Processes in Simulation: A Tool For Exploring and Modifying

David Scerri, Sarah Hickmott, Lin Padgham

RMIT University, Melbourne, Australia
Outline

1 Motivation
2 Belief Desire Intention (BDI) Agents
3 Tool for Dynamic Exploration
4 Refining the Model
5 Discussion and Conclusion
Outline

1 Motivation

2 Belief Desire Intention (BDI) Agents

3 Tool for Dynamic Exploration

4 Refining the Model

5 Discussion and Conclusion
Representation of Agent Decisions and Behaviours

- Simple reactive rules work well for things like flocking, traffic movement, or crowd behaviour.

- But social science simulations often need more complex human behaviours.

- e.g. modelling behaviours and decisions in bushfire evacuation, or in a flood situation, will involve more complex decisions.

- Can be coded in java/python/etc. or represented as a FSM - but this is not easy for non-programmers to understand.

- We want something more intuitive for non-programmers.
Modelling of Human Behaviour

- Humans are reactive - but not entirely.

- They typically have goals and plans that extend over a period of time.

- They make and adjust decisions based on the unfolding situation.

- They know what they have been doing and why - this is part of what they do next.

- The BDI agent paradigm captures these aspects well and has been used extensively for developing intelligent agent systems.
Motivation

Belief Desire Intention (BDI) Agents

Tool for Dynamic Exploration

Refining the Model

Discussion and Conclusion

Outline
BDI Agent Oriented Programming

• BDI Agent-Oriented Programming provides abstraction at the level of **mental attitudes** to explain the operation of a system. Beliefs, Desires, Intentions.

• The **modularity of plans** makes it easy to develop complexity incrementally.

• The **goal oriented** approach makes it suitable for use in dynamic environments.

• Many **efficient and powerful** development environments available. JACK, Jadex, Jason, PRS, 2APL, ...

• BDI agent programs are **fast to develop**. A 2006 study showed:
  • Gain compared to Java programming **500%**.
Belief-Desire-Intention (BDI) Agent Architecture

Percepts in, actions out. Internally, beliefs, goals and plans.
Example Plan Structure
A plan is a sequence of steps
A step can be a goal, an action, a message to another agent, or some computation.
A goal may have different plans, for achieving it in different situations.
A goal may have **different plans**, for achieving it in **different situations**.
A goal may have different plans, for achieving it in different situations.
A goal may have different plans, for achieving it in different situations.
For a goal to succeed one of the plans must succeed. If one fails try another.
For a plan to succeed, all steps must succeed.
Example Plan Structure

If things fail, recovery happens as locally as possible
Plan selection **responsive** to changing environment.
Outline

1 Motivation

2 Belief Desire Intention (BDI) Agents

3 Tool for Dynamic Exploration

4 Refining the Model

5 Discussion and Conclusion
Need to Understand at Runtime

- For **community use**: individuals want to identify with an agent, understand and maybe control it.

- For **participatory modelling**: experts need to understand what is happening, verify or change the model.

- **General debugging.**
Tool Overview

Three main ways the tool can be used:

- To stop at a specific point, and explore what the agent has decided and why.

- To control the choices an agent makes at particular points, in order to observe the effect of those choices.

- To modify and refine the model - useful for participatory modelling.
Specify Stop Points for an Agent

Select which agent to track
Specify Stop Points for an Agent

Select which agent to track

Select goals/decision points to investigate.
The Goal-Plan hierarchy
The Goal description
Tool Information at Stop Point

The Applicable Plans
Tool Information at Stop Point

The Parameter Bindings for the selected plan
Tool Information at Stop Point

The Plan Description
Exploring Non-Applicable Plans

Can also select and get information about non-applicable plans
Exploring Non-Applicable Plans

No parameter bindings. Also context failed (red).
In this example the first clause could be satisfied (green), but not the second (red)
The **Context clauses** used for any of the plans
Further Information

The numbers indicate plan priority.
Any instances which have been tried and **failed** for this goal instance are annotated
Interactive Plan Selection

Selected PlanWalk
Interactive Plan Selection

User Selection
Select Now or Always

When the user makes a different selection than the system would have made, this can be for two different reasons:

- To explore what would happen with a different selection. This does not require a permanent change to the program.
- To refine the program to make this choice in future. Requires more information and change to the program.

E.g. Select “PlanWalk”; need to know whether this choice should be incorporated into the program, and if so under what conditions.
Modifying Selection for Future

Possibilities are:

- The selected plan should **always** be chosen if applicable. Modify priorities to reflect this.
- In some situations (including current one), this plan should be preferred. Describe in comments pane the situational factors that indicate that this should be the preferred plan.
- A specific instance of a plan type should be chosen. Priority is on the type, so same for all instances of a type. Describe in comments pane what factors about an instance are used to make this choice.
Possibilities are:

- **The selected plan should always be chosen** if applicable. Modify priorities to reflect this.
- **In some situations** (including current one), this plan should be preferred. Describe in comments pane the situational factors that indicate that this should be the preferred plan.
- A specific instance of a plan type should be chosen. Priority is on the type, so same for all instances of a type. Describe in comments pane what factors about an instance are used to make this choice.
Modifying Selection for Future

Possibilities are:

- The selected plan should **always** be chosen if applicable. Modify priorities to reflect this.

- In **some situations** (including current one), this plan should be preferred. Describe in comments pane the situational factors that indicate that this should be the preferred plan.

- A **specific instance of a plan type** should be chosen. Priority is on the type, so same for all instances of a type. Describe in comments pane what factors about an instance are used to make this choice.
Modifying the Model

Ways to change the model:

1. change plan priorities (changing ranking)

2. change context condition of one or more plans (changing applicability)
   - may require new information from environment

3. provide extra selection information

4. provide additional plans
Modifying the Model

Ways to change the model:

1. change plan priorities (changing ranking)

2. change context condition of one or more plans (changing applicability)
   - may require new information from environment

3. provide extra selection information

4. provide additional plans

These can be done directly as long as no new information is required.
Modifying the Model

Ways to change the model:

1. change plan priorities (changing ranking)

2. change context condition of one or more plans (changing applicability)  
   - may require new information from environment

3. provide extra selection information

4. provide additional plans

These must be programmed. Comments can be provided for developer.
Modifying Context: some issues

- Modifying context conditions must be done with care!

- If a clause that binds a variable is removed, this may cause a problem if variable is used in the plan.
  - E.g. Neighbour($N) in context;
    Ring $N in plan.

- Could reason about this and provide warnings or disallow. Not currently done.
Outline

1 Motivation

2 Belief Desire Intention (BDI) Agents

3 Tool for Dynamic Exploration

4 Refining the Model

5 Discussion and Conclusion
Need to Consider All Plans for that Goal

- To understand a plan selection, need to understand what was not applicable.

- e.g. choose walk because nothing else available, not because of context for walk plan...

- Decisions leading to current point may also be relevant - this is evident in the hierarchy.
Need to Generalise Current State to Express Context

- Ideally allow user to indicate plan selection, determine (preference) context from current state.

- Not that simple.
  - some aspects are irrelevant;
  - some aspects must be generalised.

- Could potentially use learning
  - but simpler to just ask...
Conclusion

• A tool to understand the internals of a single agent.

• Useful for
  • community awareness tools: user identification,
  • debugging and development,
  • participatory modelling,
  • domain expert input.

• Initial development in JACK - now planning development in Jadex (open source).

• Many potential areas of development, but want to trial and evaluate first.
Questions