Expectation and Complex Event Handling in BDI-based Intelligent Virtual Agents (Demonstration)

Surangika Ranathunga
Department of Information Science
University of Otago
Dunedin 9054, New Zealand
surangika@infoscience.otago.ac.nz

Stephen Cranefield
Department of Information Science
University of Otago
Dunedin 9054, New Zealand
scranefield@infoscience.otago.ac.nz

ABSTRACT

When operating in virtual communities, intelligent agents should maintain a high-level awareness of the physical and social environment around them in order to be more believable and capable. However, due to the inherent differences between virtual worlds and agent systems such as BDI, such a high-level of awareness has not been achieved for IVAs. In this paper we present a system that enables IVAs to maintain a high-level awareness of their environment by identifying complex events taking place in their environment, as well as by being able to monitor for the fulfilment and violation of their expectations.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents, Multiagent systems

General Terms

Design

Keywords

Intelligent Virtual Agents, Expectations, BDI, Complex Event Processing

1. INTRODUCTION

Intelligent Virtual Agents (IVAs) are present in many virtual communities alongside human participants. While interacting with human participants in virtual communities these IVAs are expected to exhibit an acceptable level of awareness of the environment they are operating in.

The task of dynamically perceiving and comprehending what is happening in an agent's surrounding environment is non-trivial, given the inherent differences between virtual worlds and agent systems. First and foremost, there is an information representation gap between agent systems and virtual worlds. Virtual worlds operate with low-level primitive data while agent systems such as those based on the BDI (Belief-Desire-Intention) architecture are declarative, operating at higher abstraction levels. Solutions implemented for this problem have mainly focused only on creating static abstractions of virtual environments. Moreover, virtual worlds operate at a much higher frequency and generate large amounts of low-level sensor data, when compared with agent systems that perceive their environment at a lower frequency. This results in a cognitive overload for the IVA.

In this paper, we present our solution that enables an IVA to dynamically comprehend the abstract events unfolding in its surrounding environment and make use of this knowledge to identify the fulfilments and violations of its expectations. In achieving this, the implemented framework has two main components: The first component is a data processing module that processes the low-level sensor data received from a virtual world to identify domain-specific abstract information that is of interest to an IVA. This process is based on a virtual environment formalism we have developed in previous research [4]. This high-level information is used by the expectation monitor component to identify fulfilments and violations of agent expectations. The agent can also use this high-level information as percepts in its deliberation process.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. SOLUTION OVERVIEW

The first step in dynamically comprehending the surrounding environment is to create a coherent snapshot of the virtual environment, based on the primitive sensor data. This step is important because a piece of sensor data received from a virtual world at a given time instant may not contain the state of all the entities in that environment. However, for the successful implementation of the subsequent steps, it is important that we have a complete view of the environment observable by the IVA at the time instant of the received piece of sensor data. The snapshot generated in the first step accomplishes this requirement. The second step is to identify the static relationships (e.g. spatial or structural) among the entities included in an individual snapshot. This provides the first level of abstraction over the sensor data. In the third step, snapshots enriched with the entity relationship information are subject to complex event recognition techniques to identify the dynamic (i.e. temporal) relationships between entities, thus further abstracting the low-level sensor data.

We have presented an interface that can be implemented by an agent platform to enable its agents to start and stop monitoring for their expectations [3]. Through this interface, monitoring for agent expectations can be delegated to a monitoring service provided by the local agent platform. This enables agents to monitor for the fulfilment and violation of their expectations without relying on a centralised monitoring mechanism. This way, it is possible for agents to have plans that respond to identified fulfilments and violations of their expectations, while being able to make use of well established expectation monitoring techniques.

3. IMPLEMENTATION

In Figure 1, the *Data Processing Module* is the central processing component that identifies high-level domain-specific complex events. The monitoring service contains the logic for identifying fulfilments and violations of expectations delegated by agents.

The *VW Connection Manager* provides an interface to connect agent systems with the given virtual world and our data processing module over a TCP/IP connection. It can accommodate multiple concurrent agents to be deployed in the virtual world[1]. Figure 1 shows how this interface is used to deploy Jason [1] agents in the popular multi-purpose virtual world Second Life[2]. The SL Client module contains logic specific to extracting sensor data from Second Life.

3.1 Data Processing Module

The data processing module has three main processing levels. First, data inference and data amalgamation mechanisms are employed on the received dynamic low-level sensor data of entities (objects and avatars), and snapshots of the virtual environment are created. A snapshot provides a complete view of the environment observable by the agent at a given time instant. Based on our virtual environment formalism, a snapshot at this level contains low-level dynamic property values of entities (e.g. their positions, velocities and the currently played animations), messages exchanged in the public chat channels, and primitive events generated by the changes of dynamic property values of entities.

In the second step, each snapshot is analysed to identify the non-temporal relations between entities. Such relations can include the location of entities with respect to given land marks in the environment, and entities close to a given entity.

The *Data Pre-Processor* is responsible for both these processing steps. It makes use of a *static relation identifier* that contains logic needed to identify relations for a given virtual simulation. This is implemented as an external rule-based dynamic script. Thus this logic is readily customisable for the specific needs of a simulation. It also utilises an external database to store the static information needed to identify these relations.

The third step is to identify the temporal relations included in these snapshots. The *Complex Event Detector (CED)* achieves this task. Currently, we employ the Esper complex event processing engine[3] to identify the high-level temporal relations between the snapshots generated by the *Data Pre-Processor*.

The *Data preparation sub module* processes the snapshots received from the *Data Pre-Processor* into a format suitable for the *CED* module. This way, a new complex event recognition mechanism can be easily employed. Finally, the *Data Post-Processor* amalgamates the identified high-level temporal relations with the original snapshot. It then converts the snapshot to a string representation to be sent over the TCP/IP connection. At this level, the snapshot contains three levels of abstractions of the virtual world sensor data. It is possible to eliminate the inclusion of low-level sensor data in the snapshot, thus reducing the number of percepts sent to the agent. If the snapshot is communicated to the agent only when an interesting temporal relation occurred, this further reduces the amount of information sent to the agent.

3.2 Monitoring for Expectations

Snapshot strings generated by the data processing module are used by the Jason Environment class to prepare percepts for Jason agents. Using a TCP/IP connection in the *EM Connection Manager*, the Environment class also forwards these snapshot strings to a separate monitoring service. The monitoring service implements instances of an expectation monitor developed in previous research [2]. This expectation monitor is implemented in Python, and is integrated with the C#-based monitoring service using IronPython[4]. A single expectation monitor is responsible for monitoring for a specific agent expectation defined as a rule in temporal logic. We have introduced two new Jason internal actions that enable agents to start and stop monitoring (i.e. starting and stopping of expectation monitors) for their expectations [3]. Fulfilments and violations identified by an expectation monitor are communicated back to the corresponding agent as events to be handled by plans.

[1]Only one connection is shown in the figure for clarity.
[2]`http://secondlife.com/`
[3]`http://esper.codehaus.org/`

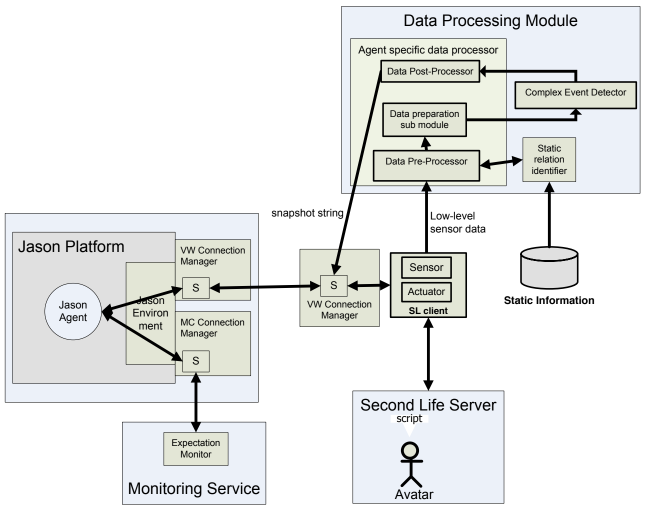

Figure 1: Framework

4. REFERENCES

[1] R. H. Bordini, J. F. Hubner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons Ltd, England, 2007.

[2] S. Cranefield and M. Winikoff. Verifying social expectations by model checking truncated paths. *Journal of Logic and Computation*, 21(6):1217–1256, 2011.

[3] S. Ranathunga, S. Cranefield, and M. Purvis. Integrating Expectation Handling into Jason. In *International Workshop on Programming Multi-Agent Systems (ProMAS 2011)*, pages 105 – 120, 2011. `http://www.cs.huji.ac.il/~jeff/aamas11/workshops/ProMAS2011.pdf`.

[4] S. Ranathunga, S. Cranefield, and M. Purvis. Identifying events taking place in Second Life virtual environments. *Applied Artificial Intelligence*, 26:137–181, 2012.

[4]`http://ironpython.net/`