Prometheus Design Tool

John Thangarajah
johthan@cs.rmit.edu.au

Lin Padgham
linpa@cs.rmit.edu.au

Michael Winikoff
winikoff@cs.rmit.edu.au

RMIT University
Melbourne, Australia

ABSTRACT

The Prometheus Design Tool is a graphical editor which supports the design tasks specified within the Prometheus methodology for designing agent systems. The tool propagates information where possible and ensures consistency between various parts of the design.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms

Design

Keywords

Agent Software Engineering, Methodologies, Tool Support

1. INTRODUCTION

The Prometheus Design Tool (PDT, see figure 1) is developed to support design and development of multi-agent systems using the Prometheus methodology as described in [4]. It is intended to support software engineers and developers in both developing and documenting the various aspects of specifying and designing a system using an agent oriented approach. The three design phases in Prometheus, which are supported within the tool are:

System specification: in which the goals of the system are identified, the interface between the agents and their environment is captured in terms of actions and percepts, functionalities are described, and detailed scenarios consisting of sequences of steps are developed.

High-level (architectural) design: in which the agent types that will exist in the system are defined by combining functionalities, the overall structure of the system is described using a system overview diagram, and interaction protocols are used to capture the dynamics of the system in terms of legal message sequences.

Detailed design: in which the internals of each agent are developed in terms of capabilities, events, plans and data. Process diagrams are used as a stepping stone between interaction protocols and plans.

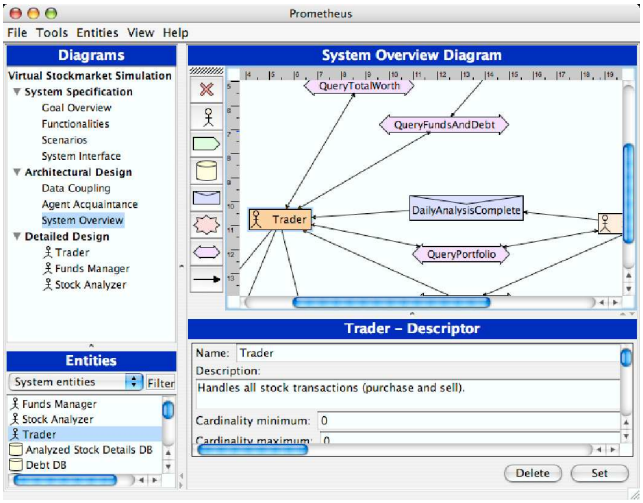

Figure 1: The Prometheus Design Tool (PDT)

Like most modern software engineering methodologies, Prometheus is applied in an iterative manner. An issue that inevitably arises is that when making a change to the design, it is virtually impossible to remember to change all other parts of the design that are effected, so as to ensure that the design remains consistent. Therefore some form of *consistency checking* is necessary. This can be done manually, but this is extremely tedious and error prone, and therefore tool support for consistency checking is highly desirable. This was also confirmed by feedback from students and others who used Prometheus prior to tool support being available.

The tool is written in Java and runs on any platform supporting Java. It has been used extensively under both Windows and Unix and can be downloaded from http://www.cs.rmit.edu.au/agents/pdt.

2. FEATURES OF THE TOOL

The tool supports software development activities from specification through to a very detailed design level which is close to code. The output of the tool can readily be transformed into JACK agent system code, and work is currently underway to automate this. Following is a list of some of the most important features of the PDT:

Graphical interface plus structured textual descriptors: The tool provide a direct manipulation graphical interface for creating the key diagrams of the Prometheus methodology. These include

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

actor diagrams[1], system overview diagrams and agent overview diagrams. There are also textual forms which allow a combination of free text and entries based on menus of items.

Propagation: Wherever possible, information is propagated from one part of the design to another. For example, if goals are associated with a role[2], and the role is associated with an agent, then the goals are also automatically associated with that agent. Similarly, graphical icons representing things that should be included in a particular diagram are automatically placed into that diagram. Whenever information about an application entity is changed, it is propagated to all relevant places in the design.

Consistency checking: The consistency checking performed by the tool has two aspects. One aspect is continuously active: the user interface will prevent certain errors from being made in the first place. The sorts of errors prevented include: (i) *Definition:* it is not possible to have references to non-existent entities, since creating a reference will create the entity if it does not exist, and when an entity is deleted all references to it are deleted as well. (ii) *Naming:* it is not possible for two entities to have the same name, for example a goal and a plan both called *Determine Stocks To Buy*. (iii) *Simple type errors:* for example, it is not possible in PDT to connect an action and another action. (iv) *Scope constraints:* for example, it is not possible to create an incoming percept to a plan without that percept also being (a) shown on the system overview diagram, and (b) shown as incoming to the agent whose plan it is. (v) *Violations of interface declarations:* for example, if an agent is specified as reading a belief set, then it is not possible to create a "write" arrow from one of the agent's plans to the belief set. Similarly, if an agent specifies that it only sends a message, then its plans cannot receive the message, and PDT does not allow the user to violate this constraint.

The other aspect is a consistency check that is performed on demand, generating a list of errors and warnings that can be checked by the developer. Examples of a warning are writing of internal data that is never read, while an example of an error is a mismatch between the interaction protocol specified between two agents and the messages actually sent and received by processes within those agents. Further details are available in [3].

Hierarchical views: The tool allows for each agent to be developed with as many layers of abstraction as needed to keep each layer manageable in size. This is achieved using capabilities and capability overview diagrams. However better support for abstraction is needed at the system level, where an improvement would be to allow a diagram which captures subsystem interaction rather than simply agent interaction.

Report generation: One of the very useful features of the tool is its ability to generate an HTML design document. This document contains both figures and textual information, as well as an index over all the design entities. The tool can also save printable images of the various diagrams (in PNG format).

Integration into Eclipse: PDT can be run as a plug-in to Eclipse, allowing use of the various tools within Eclipse. Further support within the Eclipse environment is currently being developed.

3. RELATED WORK

A number of other agent-oriented software engineering methodologies have tool support including Tropos [1] and MaSE [2]. Tropos tool support[3] consists of a number of separate tools that cover different aspects of the software engineering process. The closest tool to PDT is TAOM, but it does not appear to support cross checking or hierarchical views. The MaSE methodology is supported by agentTool[4]. Unfortunately, the MaSE methodology views agents as "black boxes" and thus does not support the design of plan-based agents. The JACK Design Environment (JDE) is also related to our work, since it provides design diagrams. However, the JDE does not support system specification activities or high level design. Finally, a tool that takes designs produced with PDT and generates a Jadex[5] agent definition file has been independently developed (mentioned in [6]).

4. CONCLUSIONS AND FUTURE WORK

The Prometheus Design Tool provides a number of features that are extremely useful in developing larger projects. However, not all aspects of the Prometheus methodology are currently supported. Future work includes: developing (better) support for protocol specification, developing support for process specification within agents; and integration of separate debugging tools. There are also plans to improve the user interface to facilitate more flexible abstraction and expansion. Also, the tool does not currently support well the exploration of multiple design options, prior to deciding on a particular path. This would greatly improve the usefulness of the software as a sophisticated design tool.

We would like to acknowledge the support of Agent Oriented Software Pty. Ltd. and of the Australian Research Council (ARC) under grant LP0453486.

5. REFERENCES

[1] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi Agent Systems*, 8(3):203–236, May 2004.

[2] S. A. DeLoach. Analysis and design using MaSE and agentTool. In *Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS)*, 2001.

[3] L. Padgham and M. Winikoff. Prometheus: A pragmatic methodology for engineering intelligent agents. In *Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies*, pages 97–108, Seattle, Nov. 2002.

[4] L. Padgham and M. Winikoff. *Developing Intelligent Agent Systems: A practical guide*. Wiley Series in Agent Technology. John Wiley and Sons, 2004.

[5] M. Perepletchikov and L. Padgham. Use case and actor driven requirements engineering: An evaluation of modifications to Prometheus. Technical report, Submitted for publication, 2005.

[6] J. Sudeikat, L. Braubach, A. Pokahr, and W. Lamersdorf. Evaluation of agent-oriented software methodologies: Examination of the gap between modeling and platform. In P. Giorgini, J. Müller, and J. Odell, editors, *Agent Oriented Software Engineering (AOSE)*, July 2004.

[1]This is a recent addition to the methodology and is described in [5].

[2]In a process of integrating our approach with that of some others, what are referred to as *Functionalities* in [4] are now referred to as *Roles*.

[3]http://trinity.dit.unitn.it/~tropos/tools.php, visited 12th April 2005

[4]http://www.cis.ksu.edu/~sdeloach/ai/projects/agentTool/agentool.htm, visited 12th April 2005

[5]http://vsis-www.informatik.uni-hamburg.de/projects/jadex/, visited 12th April 2005