AUML Protocols and Code Generation in the Prometheus Design Tool*

Lin Padgham John Thangarajah Michael Winikoff
School of Computer Science and IT
RMIT University
Melbourne, Australia
{linpa,johthan,winikoff}@cs.rmit.edu.au

ABSTRACT

Prometheus is an agent-oriented software engineering methodology. The *Prometheus Design Tool* (PDT) is a software tool that supports a designer who is using the Prometheus methodology. PDT has recently been extended with two significant new features: support for Agent UML interaction protocols, and code generation.

Categories and Subject Descriptors

D.2.2 [**Design Tools and techniques**]: Computer-aided software engineering (CASE); I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents*

General Terms

Design

Keywords

Agent Oriented Software Engineering, Design Tools, Agent UML

1. PROMETHEUS

Prometheus [9] is a software engineering methodology for the design of agent systems. It comprises a set of concepts, a process, and notations for capturing the requirements and design of the system. The process described in the Prometheus book [9] includes both a high level description of the phases of the methodology, and, importantly, detailed steps and techniques for carrying out the various high level steps. For example, one high level step in the overall process is identifying the agent types in the system, and the Prometheus methodology describes how this is done by grouping roles, considering coupling and cohesion, using a data coupling and an agent acquaintance diagram, etc.

Prometheus consists of three phases (depicted in figure 1):

- **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, roles 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 roles, 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.

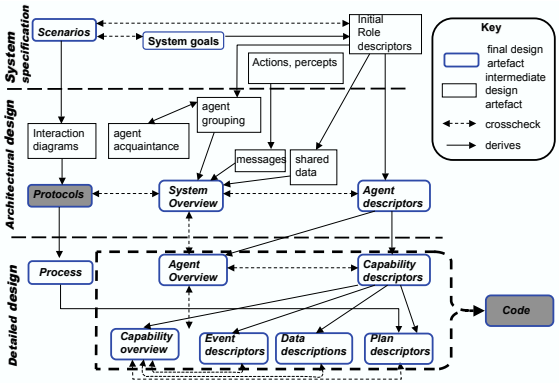

Figure 1: The Prometheus methodology

2. PROMETHEUS DESIGN TOOL

The Prometheus Design Tool (PDT) is a freely available[1] [8] tool, running under Java 1.5, which supports the software designer who is using the Prometheus methodology. PDT provides graphical support for the design phases of the methodology, allowing the designer to enter and edit diagrams and descriptors for entities. PDT also enforces certain constraints (e.g. that an action performed by an agent in the system overview diagram must also appear in the relevant agent overview diagram) and also checks the design for various consistency conditions.

*This work was funded by the Australian Research Council under grant LP0453486, in collaboration with Agent Oriented Software.

[1]From http://www.cs.rmit.edu.au/agents/pdt/

AAMAS'07, May 14–18, 2007, Honolulu, Hawai'i, USA.

978-81-904262-7-5 (RPS) ©2007 IFAAMAS

Two recent additions to PDT (which are highlighted in figure 1) are support for Agent UML interaction protocols, and the ability to generate code.

2.1 Support for AUML in PDT

As can be seen in figure 1, one of the steps in the architectural design is developing interaction protocols that capture the dynamics of the system, in terms of messages between agents. Although a wide range of notations could be used for capturing interaction protocols, we choose to adopt the widely-used Agent UML (AUML[2]) notation [6], specifically its *sequence diagram* notation, which has also been adopted by other methodologies, such as Gaia and Tropos.

Until recently PDT lacked complete support for capturing interaction protocols, and was only able to capture sequences of messages. This has been remedied with the addition of support for Agent UML. The new PDT allows a designer to specify AUML interaction protocols using a textual notation [11]. The traditional AUML graphical rendition is automatically generated, and the tool also propagates information from the protocol to the rest of the design. For example, if a protocol indicates that an agent sends a message, then that message is added to the agent's interface in the system overview and agent overview diagrams.

2.2 Code Generation in PDT

Once a design for an agent system has been developed, the design needs to be implemented. PDT has recently been extended with the ability to generate skeleton code in the JACK agent-oriented programming language [2]. The code generator extension to PDT also maintains synchronisation between the generated code and the design when either of them changes.

3. RELATED WORK

Of the many agent-oriented methodologies that have been proposed, not many have well-developed CASE tools. Two methodologies that do have publicly available and well-developed support tools are MaSE [4], with agentTool[3], and Tropos [1], with TAOM4E[4]. Other existing tools include the REBEL tool[5], supporting the ROADMAP methodology [7], and the PTK[6] tool, supporting the PASSI methodology [3]. Additionally, the JACK Design Environment (JDE)[7] provides design diagrams, but does not provide support for a full methodology.

4. DISCUSSION

The Prometheus methodology has been taught to undergraduates since 2001, and our experiences in teaching this course led to the creation of an initial prototype tool in 2002, with the current Java-based tool being developed in the first half of 2003. Since 2003 it has been actively enhanced, extended, and debugged.

Key areas for future work for PDT include enabling interchange of (relevant parts of) designs with the ISLANDER tool for designing electronic institutions [5]; integrating work on debugging using design models [10]; support for importing and exporting parts of the design, allowing design by a team of developers; and integrating support for testing agent systems, with the focus on *unit testing*.

[2]http://www.auml.org
[3]http://www.cis.ksu.edu/~sdeloach/ai/projects/agentTool/agentool.htm
[4]http://sra.itc.it/tools/taom4e/
[5]http://www.cs.mu.oz.au/agentlab/
[6]http://mozart.csai.unipa.it/passi/ptk.htm
[7]http://www.agent-software.com.au

5. ACKNOWLEDGEMENTS

We would like to thank the many people who have worked on PDT over the years: Christian Andersson, Anna Edberg, Claire Hennekam, Jason Khallouf, Ian Mathieson, Mikhail Perepletchikov, Aman Sahani, Shankar Srikantaiah, and anyone else we may have forgotten.

6. 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] P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical report, Agent Oriented Software Pty. Ltd, Melbourne, Australia, 1998. Available from *http://www.agent-software.com*.

[3] M. Cossentino and C. Potts. A CASE tool supported methodology for the design of multi-agent systems. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP'02)*, Las Vegas, 2002. Available from *http://mozart.csai.unipa.it/passi/*.

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

[5] M. Esteva, D. de la Cruz, and C. Sierra. Islander: an electronic institutions editor. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2002)*, pages 1045–1052, Bologna, Italy, July 15-19 2002.

[6] M.-P. Huget and J. Odell. Representing agent interaction protocols with agent UML. In *Proceedings of the Fifth International Workshop on Agent Oriented Software Engineering (AOSE)*, July 2004.

[7] T. Juan, A. Pearce, and L. Sterling. Roadmap: extending the gaia methodology for complex open systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 3–10, 2002.

[8] L. Padgham, J. Thangarajah, and M. Winikoff. Tool Support for Agent Development using the Prometheus Methodology. In K.-Y. Cai, A. Ohnishi, and M. Lau, editors, *Proceedings of the Fifth International Conference on Quality Software (QSIC 2005)*, pages 383–388, sep 2005. Workshop on Integration of Software Engineering and Agent Technology (ISEAT).

[9] L. Padgham and M. Winikoff. *Developing Intelligent Agent Systems: A Practical Guide*. John Wiley and Sons, 2004. ISBN 0-470-86120-7.

[10] L. Padgham, M. Winikoff, and D. Poutakidis. Adding debugging support to the Prometheus methodology. *Engineering Applications of Artificial Intelligence, special issue on Agent-oriented Software Development,*, 18(2):173–190, 2005.

[11] M. Winikoff. Defining syntax and providing tool support for agent UML using a textual notation. *International Journal of Agent Oriented Software Engineering (IJAOSE)*, Accepted, to appear.