Agent Factory: A Revised Agent Prototyping Environment

R.W. Collier, G.M.P. O'Hare

PRISM Laboratory, Department of Computer Science, University College Dublin (UCD), Belfield, Dublin 4, Ireland
{rem.collier@ucd.ie, gregory.ohare@ucd.ie}

Abstract. Recent years has seen a rise in interest in Intelligent Agents. Much of this interest is focused on how Agent Technologies may be used in modern computer systems. A consequence of this is the need for robust environments that facilitate the efficient construction of Multi-Agent Systems (MAS). One such system is Agent Factory. This paper provides an overview of the Agent Factory System, with particular attention being paid to the Agent Model adopted within the system. This model attempts to embrace recent research in MAS, in particular with respect to Agent Communication Languages, and diverse models of Commitment.

1. Introduction

The Agent Factory System is an environment for the rapid prototyping of intelligent agents that falls squarely in the category of environment that adheres to the Belief-Desire-Intention (BDI) philosophy of rational action [4][19], and promotes inter-agent communication through Agent Communication Languages (ACLs) [5][11].

Detailed descriptions of the Agent Factory System have been presented previously in [6][13][14]. Instead, this paper concentrates on revisions to the Agent Factory System, with particular focus on the generic agent model used and places it within the scope of existing MAS prototyping environments. Section 2 introduces some other MAS prototyping environments, section 3 outlines the Agent Factory System, section 4 introduces the agent model, and section 5 introduces some applications. Finally, section 6 provides some concluding remarks.

2. Multi-Agent Prototyping Environments

Recent trends within the field of Distributed Artificial Intelligence (DAI) and in particular Multi-Agent Systems (MAS) have led to the emergence of environments that facilitate the construction of agent communities. Three such environments, chosen for their adherence to the BDI and/or ACL approach, which have gained particular credence are *AgentBuilder* [2], *JATLite* [3], and *dMars* [1]. *AgentBuilder* offers an integrated tool suite for designing and constructing Intelligent Software Agents. Recent developments include the addition of mental state designed to facilitate the construction of BDI type agents. Inter-agent communication is through the Knowledge Query and Manipulation Language (KQML) ACL [10]. The Java Agent Template, Lite (*JATLite*) is a Java package that facilitates the construction of Software Agents. JATLite uses the KQML ACL for inter-agent communication and provides an open framework for reasoning that has been used successfully with BDI-architectures. *dMars* is an Agent-Oriented Programming Environment [17] developed by Australian Artificial Intelligence Institute (AAII). The system provides a set of tools for the rapid configuration of agents, and insures ease of integration into existing systems. dMars provides no facility for the use of modern ACLs. Many such agent prototyping systems now exist, however, most pay only lip service to modern developments in DAI and MAS such as ACLs and BDI-architectures. A space exists for the development of a agent prototyping environment that encompasses such technologies and provides a powerful degree of configurability. *Agent Factory* achieves this.

3. Agent Factory

The Agent Factory system has been divided into two key areas: a *Run-Time Environment* for the delivery of completed Multi-Agent Systems, and a *Development Environment* for the building and testing of Intelligent Agent designs. Tools are provided in both the run-time and development environments that support the visualization of both individual agents and agent communities.

3.1 The Run-Time Environment

The run-time environment provides the support necessary for the release of completed Multi-Agent System (MAS). This environment is further sub-divided into a *run-time server* and an *agent interpreter*. These components are augmented by a set of tools for visualising and modifying agent instances. A Distributed Control System (section 3.1.2) is interwoven into the run-time server (section 3.1.1) and agent (section 4) designs, allowing both global and local control of the MAS.

3.1.1 The Run-Time Server

The run-time server is an optional component that offers two main services: access to the non-agent components of the system (a *controller* and a *world interface*), and a set of tools for viewing and interacting with the agent community. The controller is part of a *distributed control system* (section 3.1.2) for MAS execution. The world interface provides administrative services for the placement and maintenance of agents in their arenas. Placement involves linking the agent to its "body" (the system – be it physical or virtual – that the agent is to exert control over).

3.1.2 The Distributed Control System

The distributed control system is responsible for the execution of individual agents within the agent community. Controller modules *must* be attached to agents, and *may* be attached to run-time servers. The server controller provides global level of control, whilst the agent controllers provide local agent control. Currently, both asynchronous and synchronous control systems have been developed. The choice of control system depends on the particular problem domain the MAS is to be situated in. Asynchronous controllers offer agents a high degree of autonomy, but can cause problems when attempting synchronized social behaviors. Alternatively, synchronous controllers offer a lower degree of autonomy, but provide a *common time frame* for agents (this is a global clock that *all* agents within a community may access).

3.2 The Development Environment

The development environment extends the Agent Factory Run-time Environment. It adds a *Component Library* and a selection of tools to facilitate the rapid prototyping of agent communities. Currently, the toolset includes an *Interface Customisation Tool*, an *Agent Design Tool*, and a *State Hierarchy Viewer*.

4 The Agent Model

The Agent Model is based on Stronger Notions of Agency [20]. This augments basic features such as *reactivity*, *pro-activity*, *social ability*, and *autonomy* with features such as *rationality*, *benevolence*, and *intentionality*.

4.1 The Agent Architecture

To future proof the agent model, various modules have been identified within an generic *social agent architecture*. In developing an agent design, the developer must identify the appropriate components for these modules, and at run-time, these components are plugged into a generic agent interpreter. This architecture is depicted in figure 1.

The configurable components are: a communications module, a controller (part of the distributed control system – see section 3.1.2), an ACL Module, a Commitment Management System (section

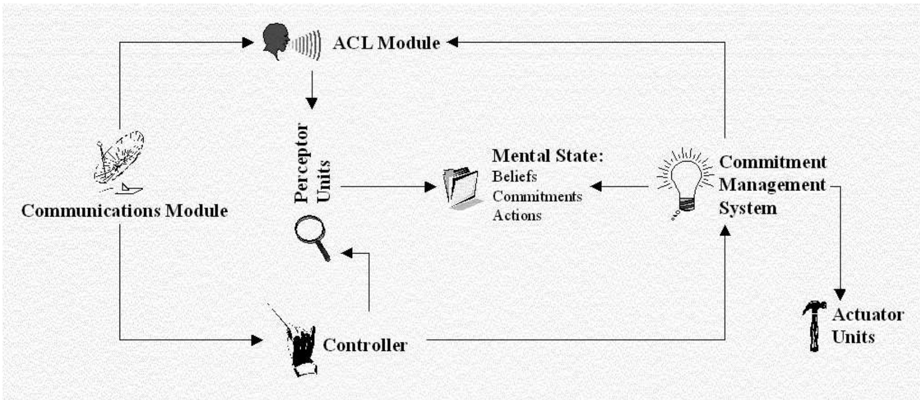

Figure 1 The Agent Architecture

4.2), the Mental State (section 4.2), and the perceptor and actuator units. The communications module permits the agent to be configured for alternate network protocols (for example TCP/IP, POP3, UDP etc). It provides a common interface hiding the protocol dependant machinery from the rest of the system. To keep with modern trends in inter-agent communications, a configurable Agent Communication Language (ACL) Module has been included. This permits the development of KQML [10] and FIPA [9] compliant Agents within Agent Factory. Agent Factory currently makes use of the *Teanga ACL* [16]. Finally, the developer is able to configure the systems perceptor and actuator units for individual agent design. The configuration of the agent architecture occurs through the associated agent design. Upon instantiation, this design is used in conjunction with a generic agent interpreter (section 4.3).

4.2 The Commitment Management System and Mental State

Recent work in MAS has seen the proliferation of commitment models [12]. These theories build commitment-based mental state architectures, and then identify strategies for the update of the commitment attitude within that architecture. For example, Rao and Georgeff [18] propose three basic commitment strategies: *blind commitment*, *single-minded commitment*, and *open-minded commitment*. Choice of such strategies is not always based on which one is the most powerful, but instead depends on the role the agent is to perform and the available reasoning time.

To reflect this the Agent Factory System provides a configurable commitment management Module, called the Commitment Management System. Configuration occurs through the selection of a Commitment Management Strategy. This strategy encompasses three areas: *commitment adoption*, *commitment revision*, and *commitment realisation*. Currently, blind and single-minded Commitment Management Strategies have been developed in Agent Factory.

To facilitate these alternate mental state models, the agent architecture provides for an extensible mental state. To ensure usability, a basis mental state of the form Belief-Commitment-Action-Plan (BDI derivative) is required by the system. This mental state is augmented with a set of Commitment Rules.

4.3 The Agent Interpreter

The agent interpreter is the part of the run-time environment that handles agent execution. In Agent Factory, a generalised algorithm has been developed that may be configured via the agent architecture (see section 4.1). This algorithm is outlined in figure 2 below.

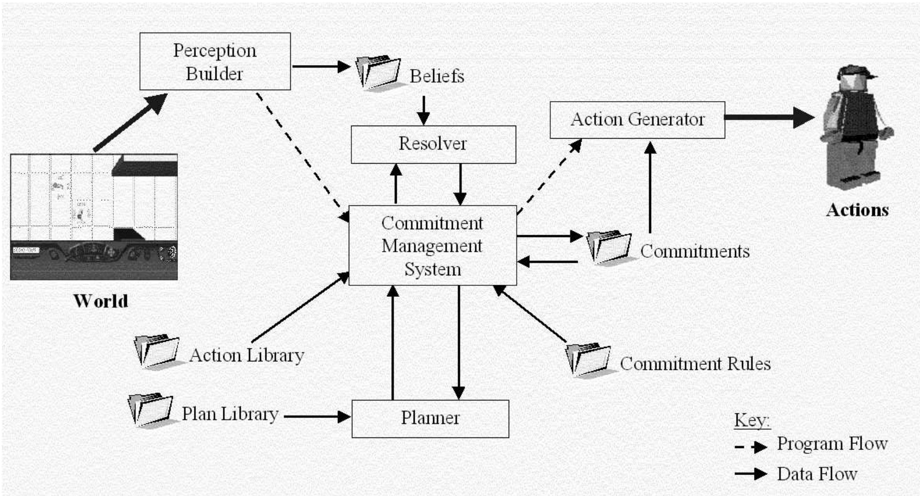

Figure 2 The Agent Interpreter

5. Applications

Agent Factory has been successfully used in a variety of heterogeneous applications, such as Social Robotics where Agent Factory is used within the Social Robotics Architecture [7][8][15], and Personal Digital Assistants (PDAs), where Agent Factory has been used to build diary manager agents that negotiate and manage diary appointments.

6. Conclusion

The Agent Factory System has been designed to reflect modern trends in DAI and MAS research. In particular, emphasis has been placed on the introduction of a generalised Agent Communication Language framework, and on a Commitment Management System module that can be configured for the different Commitment Models/Strategies that have been proposed.

The result is a highly configurable and flexible architecture that can be used in a variety of applications, and with diverse agent designs. This has been shown through the development of various demonstrators and the use of Agent Factory in the Social Robotics Architecture (section 5).

References

[1] Australian Artificial Intelligence Institute, Brief Overview, http://www.aaii.oz.au/proj/dMARS-prod-brief.html

[2] Reticular Systems, AgentBuilder White Paper, *http://www.agentbuilder.com/Documentation/white_paper_r1.3.pdf*

[3] CDR, Stanford University, "JATLite Overview", *http://java.stanford.edu/java_agent/html/*

[4] Cohen, P.R., Levesque, H.J., Intention is Choice With Commitment, *Artificial Intelligence 42, pp 213-261, 1990*

[5] Cohen, P.R., Levesque, H.J., "Communicative Actions for Artificial Agents", *1995*.

[6] Collier, R., The Realisation of Agent Factory: An Environment for the Rapid Prototyping of Intelligent Agents, *UMIST, Manchester, UK (M.Phil. Thesis), 1995*.

[7] Duffy, B.R., Rooney, C.F.B., O'Donoghue, R.P.S., Collier, R.W., O'Hare, G.M.P., Towards Social Robots, *Cognitive Science For The New Millenium, Dublin, 1999*

[8] Duffy, B.R., Collier, R.W., O'Hare, G.M.P., Rooney, C.F.B., O'Donoghue, R.P.S., SOCIAL ROBOTICS: Reality and Virtuality in Agent-Based Robotics, *Bar-Ilan Symposium on the foundations of Artificial Intelligence: Bridging theory and practice (BISFAI), 23-25 June 1999, Ramat Gan, Israel, 1999*

[9] Foundation for Intelligent Physical Agents, "FIPA 97 Specification Part 2: Agent Communication Language", *1997*.

[10] Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McGuire, J., Shapiro, S., Beck, C., "DRAFT Specification of the KQML Agent-Communication Language", *1993*.

[11] Mayfield, J., Labrou, Y., Finin, T., "Desiderata for Agent Communication Languages", *1995*.

[12] O'Hare, G.M.P. and Abbas, S., Commitment Manipulation within Agent Factory, *Proceedings of Decentralised Intelligent and Multi-Agent Systems, DIMAS '95, 22-24 Nov. 1995, Cracow, Poland, 1995*
[13] O'Hare, G.M.P., 'The Agent Factory: An Environment for the Fabrication of Distributed Artificial Systems', In O'Hare, G.M.P. and Jennings, N.R. (Eds.), *Foundations of Distributed Artificial Intelligence, Sixth Generation Computer Series, Wiley Interscience Publishers, New York, pp 449-484, 1996.*
[14] O'Hare, G.M.P., Collier, R., Conlon, J. and Abbas, S., Agent Factory: An Environment for Constructing and Visualising Agent Communities, *9th AICS Conference, Irish Artificial Intelligence and Cognitive Science Conference, UCD, Dublin, 19th-21st Aug, 1998.*
[15] O'Hare G.M.P., Duffy B.R., Collier R.W., Rooney C.F.B., O'Donoghue R.P.S., Agent Factory: Towards Social Robots, *First International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS'99), 30th May-3rd June 1999, St. Petersburg, Russia, 1999*
[16] Rooney, C.F.B., O'Donoghue, R.P.S., Duffy, B.R., O'Hare, G.M.P., Collier, R.W. The Social Robot Architecture: Towards Sociality in a Real World Domain. *Proc. TIMR-99 (Bristol) Technical Report Series, Dept. Computer Science, Manchester University, ISSN 1361-6161. Rep. No. UMCS-99-3-1, 1999.*
[17] Shoham, Y., Agent-Oriented Programming, *Artificial Intelligence (60), pp 51-90, 1993.*
[18] Rao, A.S. and Georgeff, M.P., Modeling Rational Agents within a BDI Architecture, in *Proceedings of Second International Conference on Principles of Knowledge Representation and Reasoning (J.Allen, R.Fikes, and E.Sandwall eds.) p 473-484, Morgan-Kaufmann, San Mateo, CA, 1991.*
[19] Rao A.S., Georgeff, M.P., An Abstract Architecture for Rational Agents, in *Proceedings of Third International Conference on Principles of Knowledge Representation and Reasoning (B.Nebel, C.Rich, W.Swartout eds.) pp 439-449, Morgan-Kaufmann, San Mateo, CA, 1992*
[20] Wooldridge, M. and Jennings, N.R., Intelligent Agents: Theory and Practice, *Knowledge Engineering Review* 10(2), 1995.