Agents and the UML: A Unified Notation for Agents and Multi-agent Systems?

Bernhard Bauer[1], Federico Bergenti[2], Philippe Massonet[3], and James J. Odell[4]

[1] Siemens, CT IC 6
D-81730 München, Germany
Bernhard.Bauer@mchp.siemens.de
[2] Università degli Studi di Parma, Parco Area delle Scienze 181A
43100 Parma, Italy
Bergenti@CE.UniPR.IT
[3] CEDITI, Av. Georges Lemaître, 21
B-6041 Charleroi, Belgium
Philippe.Massonet@cediti.be
[4] James Odell Associates, 3646 W. Huron River Drive
Ann Arbor, MI 48103-9489 USA
jodell@compuserve.com

Over the last few years, agent-oriented software engineering has promoted the adoption of agents as a first-class paradigm for software engineering in research and industrial development. Agents have been used in research development for more than twenty years, while they still do not find complete acceptance in industrial settings. We believe that basically three characteristics of industrial development prevent the adoption of agents: *(i)* The scope of industrial projects is much larger than typical research efforts, *(ii)* The skills of developers are focused on established technologies, *(iii)* The use of advanced technologies is not part of the success criteria of a project. In order to establish a solid ground for giving agent technologies these characteristics, we recognize that accepted methods for industrial development depend on standard representations of artifacts supporting all phases of the software lifecycle. Standard representations are needed by tool developers to provide commercial quality tools that mainstream software engineering departments need for industrial agent systems development.

Nowadays, many agent-oriented methodologies and tools are available, and the agent community is facing the problem of identifying a common vocabulary to support them. The idea of using UML as a common ground for building such a vocabulary has led to a remarkable work that is summarized in papers presented at various workshops and conferences. Just to mention papers that appeared in 2001, we find a great interest in this topic in AOSE-2001 [1, 2, 3, 4, 5, 6, 7, 8] and OAS-2001 [13, 14] workshops and in Autonomous Agents conference [9, 10, 11, 12]. Two practical reasons for using UML as a common ground are that many agent systems can be implemented in terms of distributed object-oriented technologies, and many mainstream software engineering departments already know and use UML.

The panel discussion at AOSE-2001 workshop focused on the idea of using the UML to model agents and multi-agent systems and it showed some agreement of the audience on this topic. The critiques to this idea were only technical and they concerned basically the first attempts to use the UML to model agent concepts, i.e., AUML interaction-protocol diagrams. The question that the audience raised periodically was: *"Do we really need to extend the UML or shall we go with what we have now?"* One of the original proposers suggested that the current UML language should provide a base from which we can reuse notations found useful to model agents; in this way we could minimize inventing yet-another notation, while extending and adding notation where appropriate. Two of the original proposers of such diagrams agreed that the extension of the UML was not the main point of their proposal. They decided to extend the notation only to model concepts, such as interaction protocols and roles within interaction protocols, that were not expressible with the current UML at the time of writing. Moreover, the UML community accepted some of their ideas and integrated them in next the release of the notation.

Some members of the audience expressed the fear that using UML as a basis for an agent notation would not emphasize the fact that the agent paradigm is radically different from the object-oriented paradigm. They referred to the transition from the structured to the object-oriented paradigm in the last decade and the difficulties that programmers had when making the transition from C to C++ (basically writing C++ programs in the structured programming style as if they were C programs).

Panelists agree that the discussion showed the interest of the community in finding a common vocabulary for agent technologies and, besides some technical issues, they find the idea of defining the AUML a good starting point to achieve this purpose. For the current status of the AUML please refer to the official website: `http://www.auml.org`.

Papers in Proceedings of AOSE-2001

1. Bauer, B.: "UML Class Diagrams: Revisited in the Context of Agent-Based Systems," pp.1-8.
2. Parunak, V., Odell, J.: "Representing Social Structures in UML," pp. 17-31.
3. Gervais, M.P., Muscutariu, F.: "Towards an ADL for Designing Agent-Based Systems," pp.49-56.
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: "Modeling early requirements in Tropos: A Transformation-based Approach," pp. 67-75.
5. Sparkman, C.H., DeLoach, S.A., Athie, L.: "Self Automated Derivation of Complex Agent Architectures from Analysis Specifications," pp.77-84.
6. Flores, R.A., Kremer, R.C.: "Bringing Coherence to Agent Conversations," pp. 85-92.
7. Koning, J.L., Huget, M.P., Wei, J., Wang, X.: "Extended Modeling Languages for Interaction Protocol Design," pp. 93-100.
8. Caire G., Leal, F., Chainho, P., Evans, R., Garijo, F., Gomez, J., Pavon, J., Kearney, P., Stark, J., Massonet, P.: "Agent Oriented Analysis using MESSAGE/UML," pp. 101-108.

Papers in Proceedings of Autonomous Agents 2001

9. Karacapilidis, N., Pavlos, M.: “Intelligent Agents for an Artificial Market System,” pp. 592-599.
10. Bergenti, F., Poggi, A.: “A Development Toolkit to Realize Autonomous and Interoperable Agents,” pp. 632-639.
11. Depke, R., Heckel, R., Küster, J.M.: “Improving the Agent-Oriented Modeling Process by Roles,” pp. 640-647.
12. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: “A Knowledge Level Software Engineering Methodology for Agent Oriented Programming,” pp. 648-655.

Papers in Proceedings of OAS-2001

13. Cranefield, S., Hausteiny, S., Purvis, M.: “UML-Based Ontology Modelling for Software Agents,” pp. 21-28.
14. Cranefield, S., Purvis, M.: “Generating Ontology-Specific Content Languages,” pp. 29-35.