20 Years of Agent-Oriented Programming in Distributed AI: History and Outlook

M. Birna van Riemsdijk

Interactive Intelligence
Delft University of Technology
Mekelweg 4, 2628 CD, Delft
The Netherlands
m.b.vanriemsdijk@tudelft.nl

Abstract

This extended abstract summarizes the invited talk with the corresponding title at the AGERE! workshop at SPLASH'12. It describes a history of 20 years of research in agent programming, viewed from the perspective of the author. This perspective is naturally influenced by the author's own research on cognitive agent programming languages. We do not aim for a comprehensive description or claim that this is provided by this extended abstract or the talk. Rather the aim is to highlight several important developments in the field, providing a general idea of the types of issues investigated in agent-oriented programming, targeted at researchers from other fields of computer science.

Categories and Subject Descriptors I.2.5 [*Artificial Intelligence*]: Programming Languages and Software; I.2.11 [*Artificial Intelligence*]: Distributed Artificial Intelligence—Intelligent agents, languages and structures, Multiagent systems

General Terms Languages

Keywords Agent programming languages

1. Introduction

The field of agent-oriented programming addresses the development of programming languages, platforms and development environments for programming autonomous agents and multi-agent systems. An agent can be defined as "an encapsulated computer system that is situated in some environment and that is capable of flexible autonomous action in that environment in order to meet its design objectives" [21].

AGERE! 2012, October 21–22, 2012, Tucson, Arizona, USA.

Agent-oriented programming lies in the intersection of artificial intelligence (AI) and programming and software engineering research. The main research questions addressed in the field are similar to those in mainstream programming research, but focused on programming multi-agent systems: Which concepts can or should be used for programming autonomous agents and multi-agent systems? How to translate these concepts to language constructs and abstractions? And finally how to use these languages in practice?

2. The First Decade

Agent-oriented programming has part of its origins in Bratman's Belief-Desire-Intention (BDI) philosophy [7]. BDI philoshopy was in turn inspired by Dennet's intentional stance [12]. The intentional stance has been proposed for predicting and explaining the behavior of rational agents. The basic idea is to ascribe beliefs and desires to an object treated as a rational agent, and predicting what it will do using practical reasoning applied to these beliefs and desires. Bratman argues that another notion is key in predicting and explaining the behavior of rational agents, namely intention. Intentions can be defined as chosen desires that the agent commits to achieving. Intentions, unlike desires, induce an agent to determine how to achieve them and to act accordingly. BDI notions have been further investigated through defining corresponding logics, resulting in highly influential papers [8, 26].

The term agent-oriented programming was coined by Shoham in 1993 [29], and the basic idea is that cognitive notions like beliefs, desires and intentions could be used not only to predict and explain the behavior of rational agents as in BDI philosophy, but also to *program* these agents. Shoham proposes AGENT-0, an agent-oriented programming language that incorporates these notions as first class citizens in the language. The basic construct is an **if** ⟨condition⟩ **then** ⟨action⟩ rule, where the condition refers to the "mental state" of the agent consisting of cognitive notions like beliefs and goals.

In the decade following the introduction of AGENT-0, several other agent-oriented programming languages are proposed that each take their own approach to translating cognitive notions to constructs and abstractions in an agent programming language. Examples of influential languages that have been proposed are GOLOG [23], AgentSpeak(L) [25] and 3APL [19]. One could say that research in this first decade focused on agent programming as programming with mental models (see also [17]), resulting in languages and architectures that could translate perceptual input through the use of mental notions to output in the form of actions that modify the agent's external environment (see also the architecture of [14]).

3. The Second Decade

Around 2001, a new spur in activity emerged in the area, giving rise to many new research directions. The field was broadened by expanding the view of agent programming beyond the use of cognitive notions, and investigating various new themes related to programming agent decision making. An overview of some of that research can be found in books on agent programming [4, 6]. Workshops that have stimulated research in the field in the second decade were established in 2003 as part of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS): Programming Multi-Agent Systems (ProMAS) and Declarative Agent Languages and Technologies (DALT).

Looking back, one could say that much of the research in agent programming in the second decade was driven by the following questions:

- Which knowledge representation language to use?
  Most of the languages proposed in the first decade were in one way or another based on logic programming. Researchers realized that in order to increase chances of adoption of agent programming as an established programming paradigm, it is important to also investigate the use of more mainstream languages for representing cognitive notions. The Jack language for example is an industrial strength language that is an extension of Java with agent concepts [33]. Also, the Jadex platform combines the use of XML and Java for programming agent reasoning [24].
- How flexible are these agents, really?
  Agents were argued to be capable of flexible action in highly dynamic environments, by combining reactive and proactive behavior. While the first agent programming languages indeed addressed this challenge to some extent, many challenges still needed to be addressed to make agents more flexible in their decision making. This gave rise for example to research in reasoning about goals, addressing among other things the issue of goal conflicts (see, e.g., [30, 31]).
- Where is the environment?
  In the first decade, the environment in which agents function was taken as given. An interface to the environment had to be established such that abstracted percepts could be processed easily by agents, but no further role was attributed to the environment. In the second decade the idea was proposed (see, e.g., [28]) to use the environment as a first class abstraction for multi-agent systems engineering, encapsulating functionalities and services to support agent activities.
- How to coordinate a collection of autonomous agents?
  In the first decade, the issue of how to coordinate a collection of autonomous agents was (in the context of agent programming) addressed mainly by investigating agent communication languages. These were typically incorporated in agent programming languages through asynchronous message passing mechanisms that allow agents for example to give other agents information about the state of the environment or request achievement of goals. At the beginning of the second decade, a different approach was proposed for coordination, namely imposing an organization on the multi-agent system that guides agent behavior by defining norms, organizational structure and tasks that should be performed by agents participating in the organization (see, e.g., [13, 16]). Several organizational modelling languages and normative frameworks have resulted from these investigations, as presented in for example the International Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN).
- How to specify and verify multi-agent systems?
  As in other areas of programming and software engineering research, specification and verification is important also in agent programming for ensuring correctness of the developed multi-agent system. Approaches typically investigate how to apply and adapt techniques from mainstream formal methods research to the context of agent programming. For example, techniques for model checking AgentSpeak [3] and GOAL [22] have been developed, and [11] proposes a verification framework for GOAL. An overview of state-of-the-art can be found in [10].
- How to use and support use of the languages in practice?
  While the origins of agent-oriented programming lie in philosophy and logics and a large amount of work has been done on the theoretical foundations of the languages, many researchers have also taken a more practical perspective towards the languages. In the second decade this has resulted in the development of several development platforms for agent-oriented languages such as 2APL [9], GOAL [18] and Jason/AgentSpeak [5], as well as platforms for modelling environments (e.g.,

CArtAgO [27]) and organizations (e.g., MOISE [20] and AMELI [15]).

4. Outlook

There are many directions in which research on agent-oriented programming could develop, as also exemplified by the discussions that emerged at the Dagstuhl Seminar 12342 on Engineering Multi-Agent Systems which was held in August 2012 (organized by Jürgen Dix, Koen Hindriks, Brian Logan and Wayne Wobcke). Below I highlight a few of these directions.

It will be interesting to investigate the use of agent-oriented programming as a paradigm for development of concurrent and distributed systems. As research on agent programming has different origins than more mainstream work on concurrent systems, the concepts and ideas that have been developed for agents may be inspiring for research in these areas. While several platforms and develoment environments have been developed by now, tool support for engineering multi-agent systems can be improved considerably to bring them to a level comparable to those for mainstream languages. In particular it will be challenging to provide support for debugging, as this is notoriously hard in multi-agent systems due to their concurrent nature and the dynamism of the environments in which they operate. We suggest to increasingly use empirical methods to evaluate and develop tools, languages and techniques, thereby complementing theoretical studies in the field (see position paper by the author in this volume). Through empirical methods, different kinds of questions can be answered than through analytical approaches, such as how programmers use agent programming languages and which problems they encounter. Moreover, research can be done on the integration of agents, environment and organization, such as in recent work on the JaCaMo platform [2]. Developments in the use of environments and organizations pose new challenges for (programming) agent reasoning, as agents whould be able to determine how to adapt their behavior to function effectively in these contexts (see, e.g., [1, 32]). Furthermore, there are challenges in the integration of multi-agent systems with non-agent software. It is unlikely that software systems of the future will be solely agent-based, and so to make effective use of the techniques developed in this area, it should be investigated how to integrate with other (parts of) software systems.

Acknowledgments

I would like to thank the agent programming community for its openness and generosity in including me as one of its members.

References

[1] N. Alechina, M. Dastani, and B. Logan. Programming norm-aware agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*, pages 1057–1064. IFAAMAS, 2012.

[2] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi. Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 2011.

[3] R. H. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking agentspeak. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 409–416. ACM, 2003.

[4] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[5] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming Multi-agent Systems in AgentSpeak using Jason*. Wiley, 2007.

[6] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni. *Multi-Agent Programming: Languages, Tools and Applications*. Springer, Berlin, 2009.

[7] M. E. Bratman. *Intention, plans, and practical reason*. Harvard University Press, Massachusetts, 1987.

[8] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[9] M. Dastani. 2APL: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, 2008.

[10] M. Dastani, K. V. Hindriks, and J.-J. Ch. Meyer, editors. *Specification and Verification of Multi-agent Systems*. Springer, 2010.

[11] F. de Boer, K. Hindriks, W. van der Hoek, and J.-J. Meyer. A Verification Framework for Agent Programming with Declarative Goals. *Journal of Applied Logic*, 5:277–302, 2007.

[12] D. Dennett. *The Intentional Stance*. MIT Press, Cambridge MA., 1987.

[13] V. Dignum. *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. PhD thesis, 2004.

[14] M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dMARS. In *ATAL '97: Proceedings of the 4th International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Languages*, pages 155–176, London, UK, 1998. Springer-Verlag.

[15] M. Esteva, B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos. AMELI: An agent-based middleware for electronic institutions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 236–243. IEEE Computer Society, 2004.

[16] J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: An organizational view of multi-agent systems. In *Proceedings of 4th International Workshop on Agent-Oriented Software Engineering (AOSE'03)*, volume 2935 of *LNCS*, pages 214–230. Springer, 2003.

[17] K. V. Hindriks. *Agent programming languages - programming with mental models*. PhD thesis, 2001.

[18] K. V. Hindriks. Programming rational agents in GOAL. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Tools and Applications*. Springer, Berlin, 2009.

[19] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3APL. *Int. J. of Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[20] J. F. Hübner, J. S. Sichman, and O. Boissier. Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3/4):370–395, 2007.

[21] N. Jennings. Agent-oriented software engineering. In *Proceedings of the 12th International Conference on Industrial and Engineering Applications of AI*, pages 4–10. 1999. Invited paper.

[22] S.-S. Jongmans, K. V. Hindriks, and M. van Riemsdijk. Model checking agent programs by using the program interpreter. In *Computational Logic in Multi-Agent Systems*, volume 6245 of *LNCS*, pages 219–237. Springer, 2010. URL `http://dx.doi.org/10.1007/978-3-642-14977-1_17`.

[23] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. Golog: A logic programming language for dynamic domains. *J. Log. Program.*, 31(1-3):59–83, 1997.

[24] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: a BDI reasoning engine. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[25] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. van der Velde and J. Perram, editors, *Agents Breaking Away (LNAI 1038)*, pages 42–55. Springer-Verlag, 1996.

[26] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann, 1991.

[27] A. Ricci, M. Piunti, M. Viroli, and A. Omicini. Environment programming in cartago. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Tools and Applications*, pages 259–288. Springer, Berlin, 2009.

[28] A. Ricci, M. Piunti, and M. Viroli. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23(2):158–192, 2011.

[29] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[30] J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and avoiding interference between goals in intelligent agents. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003.

[31] M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Goals in conflict: Semantic foundations of goals in agent programming. *Autonomous Agents and Multi-Agent Systems*, 18(3): 471–500, 2009.

[32] M. B. van Riemsdijk, K. V. Hindriks, and C. M. Jonker. Programming organization-aware agents: A research agenda. In *Proceedings of the Tenth International Workshop on Engineering Societies in the Agents' World (ESAW'09)*, volume 5881 of *LNAI*, pages 98–112. Springer, 2009.

[33] M. Winikoff. JACK™ intelligent agents: an industrial strength platform. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.