Agent Programming Languages and Logics in Agent-Based Simulation

John Bruntse Larsen

DTU Compute, Technical University of Denmark, 2800 Kongens Lyngby, Denmark
jobla@dtu.dk

Abstract. Research in multi-agent systems has resulted in agent programming languages and logics that are used as a foundation for engineering multi-agent systems. Research includes reusable agent programming platforms for engineering agent systems with environments, agent behavior, communication protocols and social behavior, and work on verification. Agent-based simulation is an approach for simulation that also uses the notion of agents. Although agent programming languages and logics are much less used in agent-based simulation, there are successful examples with agents designed according to the BDI paradigm, and work that combines agent-based simulation platforms with agent programming platforms. This paper analyzes and evaluates benefits of using agent programming languages and logics for agent-based simulation. In particular, the paper considers the use of agent programming languages and logics in a case study of simulating emergency care units.

Keywords: multi-agent systems, logic, simulation

1 Introduction

Agent-Oriented Programming (AOP) is a programming paradigm where programs are composed of agents. Similar to objects in Object-Oriented Programming (OOP), agents maintain a mental state and react to input by performing actions and changing their mental state. Some agents are also assumed to be intelligent agents, meaning that they pursue goals and exhibit social behavior by communicating with other agents. Agent programming languages are programming languages that are designed for development of multi-agent systems with AOP. Examples of platforms that use agent programming languages include Agent-0 [1], 3APL [2], 2APL [3], Jason [4], JACK [5, 6] and GOAL [7]. The notions of belief, desire and intention (BDI) are key components in these languages, as they respectively denote what the agent believes, what the agent would like to achieve, and what the agent is currently working towards achieving. Formalizations of a BDI model in modal logics provide syntax and semantics for the model. Thus logic provides a theoretic framework for specification and verification of agent programs.

The BDI paradigm has also been used in agent-based simulation (ABS). The purpose of ABS compared to multi-agent systems is to gain insight into how

global properties emerge from a system of local interacting processes. Examples of ABS platforms include Mason [8], Repast [9] and GAMA [10]. ABS platforms generally do not use above mentioned agent programming languages but some of them provides a framework for making models with the BDI paradigm [11]. A BDI model allows agents to exhibit more complex behavior than purely reactive models but without the computational overhead of cognitive architectures. It is generally also easier for domain experts to specify their knowledge in terms of a BDI model compared to an equations-based model, and a BDI model supports explainable behavior. Adam and Gaudou [12] present an extensive analysis and evaluation of approaches to integrating BDI models in ABS. They highlight the previously mentioned benefits of BDI models as a way to implement *descriptive agents* which use richer and more complex models than reactive agents.

This paper presents an analysis and evaluation of using recent advances in agent programming languages and logics, in particular frameworks for implementing social behavior, in ABS. The paper first presents a summary of AOP, ABS platforms and work on integrating BDI models in simulation platforms based on the work of Adam and Gaudou [12]. It then describes research in frameworks and meta-models for implementing virtual environments and social behavior in agent programming languages. It evaluates potential benefits of using a framework for implementing agent organizations in ABS, and finally discusses further use of MAOP in ABS. The criteria used in the evaluation are in terms of:

1. How the framework supports descriptive agents.
2. How reusable the framework or meta-model is.
3. How useful the framework or meta-model is for analysis.

The evaluation is based on previous work on using the agent organization framework AORTA [13] to create a simulation model for an emergency care unit [14].

2 AOP, Logic and Agent-Based Simulation

AOP was originally proposed by Shoham [1] as a specialization of OOP. Shoham motivated AOP with cases in which multiple entities interacted with each other in order to manufacture cars and reserve plane tickets. In AOP, each entity (now called an agent) maintains a mental state of beliefs, capabilities and decisions that have dedicated terms with a formal syntax. Communication with other agents occurs through speech-act inspired messages. Some of the approaches to programming languages designed for AOP include:

- AgentSpeak(L) [15] and Agent-0 [15] in which an agent has a database of plans or rules for choosing actions that match its current mental state. The agent programming platform Jason [4] implements AgentSpeak(L).
- Languages based on logic programming such as 3APL [2], 2APL [3], and GOAL [7].
- Jack [5, 6] which extends Java with agent programming keywords.

– A combination of XML and Java. This approach is used in the agent programming platform Jadex [16].

These programming languages use BDI as a common paradigm for a mental model but as it can be seen, they have very different approaches to implementing it. The BDI paradigm comes from philosophy and the mental model can be given formal syntax and semantics with epistemic logics. Other logics such as first-order logic and temporal logics can be used to specify world models of concepts and dynamics. Given a specification it is then possible to use logic reasoning to verify properties of the specification. Thus logic provides a theoretical framework for specifying and verifying properties of agent programs. In the programming languages AgentSpeak(L), 3APL, 2APL and GOAL, the agents also use logic to do reasoning in their decision making.

ABS is an approach to simulation that takes the perspective of the individuals that inhabit the simulated system. ABS is useful in cases where it is easier to describe a system in terms of interacting agents rather than as a global process [17]. A critical part of ABS is a scheduling mechanism which ensures that all agents are synchronized in a finite sequence of time steps. ABS platforms provide frameworks for ABS and typically features tools for visualizing the simulation, data extraction tools and analysis tools. Commonly used ABS platforms include:

– Mason [8] which is a Java based discrete-event simulation platform that has been extended with ABS.
– Repast [9] which is a suite of tools in multiple programming languages for implementing ABS.
– GAMA [10] which features an XML based language GAML for implementing agents. GAMA also features tools for using GIS data in the simulation.

The ABS platforms typically have tools for implementing reactive agents but little support for implementing proactive behavior. This works well for many cases but as argued by Adam and Gaudou [12], there are also cases, often those involving human agents, where more descriptive agent models are useful for gaining insight into the decision making. The BDI paradigm provides a framework for implementing descriptive agents that are still fairly efficient. There have been three general approaches to implementing BDI in ABS:

– Extending agent programming platforms with ABS features. Bordini and Hübner does this with Jason [18].
– Extending ABS platforms with BDI modeling features. Caballero et al. does this with Mason [19].
– Combining ABS platforms with agent programming platforms. Padgham et al. [20] does this with Repast and JACK, and Singh et al. [21] designs a framework for integrating any two platforms with each other.

The benefit of the last approach is that it leverages features from both platforms but with a cost of computational power in keeping agents synchronized between the platforms. Besides mental models for the individual agents, there is also work on implementing meta-models for the environment and social behavior such as

the MASQ meta-model by Dignum et al. [22]. Meta-models for environments and social behavior are covered further in the following section.

3 From AOP to MAOP

Much of the early research in agent programming languages has been focused on the internal agent architectures with different approaches to programming languages based on the BDI paradigm and speech-act communication. Both the environment that the agents inhabit and the social skills of the agents have been designed and programmed for specific domains. Recent research has gone into making more reusable frameworks and meta-models for creating environments and agent societies [23, 24]. Notable examples include:

- CArtAgO (Common Artifact Infrastructure for Agent Open environment) [25] which is a Java-based framework for developing and running virtual environments based on the Agents & Artifacts meta-model. In this meta-model, the agents use artifacts to communicate with other rather than only by speech-acts. The artifacts provide an interface for the communication that allows for also non-BDI agents to communicate with BDI agents. The framework has been integrated with Jason, 2APL and JADEX [26, 27].
- EIS (Environment Interface Standard) which is a Java-based framework for connecting agent programming platforms with environments. It is not a meta-model for environments but it acts as an interface for agent programming platforms to environment platforms such as CArtAgO-based platforms.
- OperA [28] which is a meta-model for agent organizations. In agent organizations, the agents are assigned roles that puts a structure on how the agents can use their abilities to communicate and carry out actions. The Eclipse plugin Operetta [29] is a tool for design, verification and simulation of OperA models.
- $\mathcal{M}$OISE$^+$ [30] which is also a meta-model for implementing agent organizations. $\mathcal{M}$OISE$^+$ is integrated with CArtAgO and Jason in the JaCaMo platform [31].
- AORTA [13, 32] which is a meta-model that enables individual agents to reason about organizations described in OperA. It is designed for adding organizational reasoning capabilities to BDI agents and has been integrated with Jason [33].

Table 1 summarizes the main characteristics of these frameworks and meta-models. A common feature of the examples is that they support more open heterogeneous systems of agents: agents can enter and exit the system freely even though they use different internal mechanisms for decision making. The frameworks and meta-models put an emphasis on Multi-Agent Oriented Programming (MAOP) with system level frameworks rather than traditional AOP with agent-level mental models and speech-act communication. Use of MAOP is not common in ABS literature. A possible reason for this might be that openness is less important in simulation where the purpose is to gain insight in a

Framework/meta-model	Main characteristic
CArtAgO	Virtual environments with Agents & Artifacts meta-model.
EIS	Interface between AOP platform and environment.
OperA	Agent organization meta-model.
$\mathcal{M}$OISE$^+$	Agent organization meta-model.
AORTA	Meta-model for enabling reasoning with OperA in agents.

Table 1. Summary of main characteristics of MAOP frameworks and meta-models.

given system. A potential benefit of MAOP though is that it can offer reusable tools for implementing environments and social behavior. Using MAOP with a foundation in logic would also allow for specification and validation of simulation models similar to the work presented by Jensen [34] on verification of organization-aware agents in AORTA.

4 Case Study: Emergency Care Units

In this section, the use of agent programming languages and logic in simulation is analyzed and evaluated in a case study with emergency care units based on previous work [14]. Emergency care units are responsible of providing care for acute patients. Hospitals often have an entire department dedicated to emergency care and the number of incoming patients has been increasing in recent years. Due to limited funding, it is necessary for the department to work efficiently by establishing clear guidelines including work procedures, roles and responsibilities. Such guidelines are not complete though and the doctors and nurses are flexible in taking actions that comply with the guidelines. In some cases they might even go against the guidelines. For example if there are no nurses available and a secretary has no other currently urgent tasks, the secretary might help a patient with taking a urine sample even though that is the job of a nurse. The dynamic nature of the decision making makes it difficult to keep track of what consequences the decisions can have and what can be done to avoid potential issues such as long waiting lists or staff overwork hours. Simulation could provide insight into these things given that the simulation model describes the interaction in the emergency department with sufficient accuracy.

The first thing to notice in this case study is that it involves quite different kinds of actors. There are individual human actors such as doctors, nurses and patients that show goal achieving behavior. The doctors and nurses are responsible for providing emergency care and the patients arrive at the department with the goal of receiving emergency care. The human actors also make use of various tools and IT-systems. Finally secretaries, nurses and doctors communicate with institutions such as other departments in the hospital or rescue teams. A BDI model offers flexibility to implement these different actors. The human actors can be implemented as agents that show proactive behavior, the tools and IT-systems can be implemented as reactive agents, and the other departments and institutions can be implemented as agents that have goals and can send

requests to the emergency department. Such a model can be implemented in an agent programming platform with simulation features such as Jason or an ABS platform with a BDI model extension such as Mason.

The second thing to notice in this case study is the significance of established work procedures, roles and responsibilities, which serve as guidelines rather than complete plans. The staff are expected to follow the guidelines but also to fill out with details not covered by the guidelines. The staff are assumed to be intelligent and be able to act independently a manner that fits the situation, deviating from the guidelines in extraordinary situations. In other words, the guidelines describe norms and goals that the agents are obliged but not enforced to adhere to. The BDI paradigm on its own does not offer a formalism for modeling such behavior. While BDI agents are flexible and act toward solving goals, they are enforced to follow the plans they are given. Instead it makes sense to apply meta-models for agent organizations that have frameworks for encoding organizational knowledge explicitely in the model. As mentioned earlier, some of these meta-models are already integrated with agent programming platforms. Previous work presented an AORTA model of the acute patient treatment process [14]. In the AORTA meta-model, we can encode organizational knowledge in terms of roles, objectives and sub-objectives, role dependencies and conditions. Each agent then maintains two knowledge bases: one with personal knowledge and one with organizational knowledge. The organizational knowledge base describes the stages that the patient goes through, which staff members are involved in each stage and a selection conventions that the agents are expected to follow. When deliberating which action to perform, an agent can then reason about if an action complies or violates any obligations of the agent. Updating the knowledge base is done accordingly to general rules of the meta-model when the agents perform actions. The agents can also perform organizational actions, such as enacting roles, which will update their knowledge base accordingly. In addition, the explicit representation of organizational knowledge supports specification and verification of the organizational agent model. To summarize the evaluation in accordance to the criteria proposed in the introduction, using an AORTA model in ABS would give:

1. Descriptive agents that have a mechanism to include organizational reasoning in their decision making. They are descriptive in the sense that they implement complex social behavior and support explainable behavior. An agent can use AORTA to reason about what other agents expect of the agent, and what it can expect of the other agents.
2. A reusable meta-model that can be integrated in any agent programming platform.
3. Formal syntax and semantics in logic that can used for specification and verification of the organizational agent model. Logic reasoning can provide insight into social relations which are otherwise hard to identify or reason about.

5 Discussion

The BDI paradigm on its own only provides generalized methods for implementing internal agent reasoning. It does not provide generalized methods for implementing important aspects of multi-agent systems such as organizations and environments. The previous section analyzed potential benefits that the AORTA meta-models can provide for ABS in the emergency care unit scenario. In this section we recap that analysis and discuss potential benefits of applying the other frameworks and meta-models listed in table 1 for ABS.

CArtAgO provides a framework for implementing agent environments in Java, which is commonly used in ABS platforms, using the Agents & Artifacts meta-model. In domains where people interact through physical objects such as whiteboards or telephones, CArtAgO would provide a generalized framework for encoding these objects. In the case study with emergency care units, the physical location and availability of information communication technologies can have a major influence on the workflow. CArtAgO has been implemented in Jason and has been used to an increasing extent in MAS. As it is Java based, it could potentially also be implemented for dedicated ABS platforms that support BDI models. The Agents & Artifacts meta-model also provides theoretical foundation for specification and verification of agent environments.

EIS provides a Java framework for integrating agent programming platforms with environments. This is useful for implementing systems where the internal agent reasoning logic and the environment logic are separated from each other. The separation allows for more openness, as agents can then be integrated in the environment no matter how their internal reasoning works like. As mentioned earlier, openness is less of a concern in ABS than MAS so, although the framework is reusable, we do not see an immediate benefit of using EIS in ABS.

OperA provides a meta-model for designing and analyzing agent organizations. As the evaluation in the previous sections shows, there are clear benefits of applying organization meta-models to domains with human organizations. Making a model of the organization in OperA would provide a basis for implementing ABS with AORTA agents that perform organizational reasoning. $\mathcal{M}$OISE$^+$ provides an alternative meta-model for agent organizations. Its integration with CArtAgO and Jason in JaCaMo could provide a framework for implementing ABS with both environment and organization models.

The AORTA meta-model, which was evaluated in the previous section, provides a basis for implementing organizationally aware agents in ABS platforms. Doing so would give ABS that supports descriptive agents that replicate organizational behavior in terms of roles and norms. In domains with human organizations, such as in the hospital case, simulation with organizationally aware agents should provide more accurate outcomes than with only the BDI paradigm. There are already implementations of AORTA in Jason, which to some degree supports ABS, and since AORTA has well defined semantics and operational rules, it can be implemented in dedicated ABS platforms that support BDI models. The formal syntax and semantics in logic also supports specification and verification of the organizational agent model.

In ABS of social systems, there is also a growing interest in frameworks and meta-models for social values. A social value represents a concept that an agent cares about and it will generally perform actions that promotes its social values. Simulation with social value models have gained interest as a way to implement social behavior that agents do exhibit without explicitely reasoning about them. Although there is work on meta-models for social values, there still remains much to be done in terms of formalization and implementation in ABS platforms.

6 Conclusion and Future Work

There is active research into providing better frameworks for implementing BDI models in ABS. They generally use one of the methods:

- Implementing simulation features in agent programming platforms [18].
- Implementing BDI models in ABS platforms [19].
- Combining ABS platforms with agent programming platforms [20, 21].

The third method has the advantage that it can make use of advances in tools for both ABS and AOP platforms. As argued by Adam and Gaudou [12], the cost of high computational power might also become negligible as computers get more powerful. Research in agent programming languages and logics has given frameworks and meta-models for implementing environments and social behavior. These are designed to be reusable and their logical foundation can be used for specification and verification of ABS models. The paper has given an analysis and evaluation of using agent programming languages and logics in a case study based on emergency care with the AORTA meta-model. The meta-model gives descriptive agents that can include organizational reasoning in their decision making, is reusable, and has a formal syntax and semantics in logic that can be used for specification and verification. We also discussed potential benefits of using some of the other MAOP frameworks shown in table 1 for ABS, as well as .

To the author's knowledge, there are still few reusable frameworks and meta-models for implementing social behavior in ABS. Future work include implementing an ABS of the emergency care unit scenario with agents that exhibit social behavior, and using logic for specifying and verifying properties of meta-models for social behavior.

Acknowledgements

This work is part of the Industrial PhD project *Hospital Staff Planning with Multi-Agent Goals* between PDC A/S and Technical University of Denmark. I am grateful to Innovation Fund Denmark for funding and the governmental institute Region H, which manages the hospitals in the Danish capital region, for being a collaborator on the project. I would also like to thank Jørgen Villadsen, Rijk Mercuur and Virginia Dignum for comments on the ideas described in this paper and comments on a draft.

References

1. Shoham, Y.: Agent-oriented programming. Artificial Intelligence **60**(1) (1993) 51–92
2. Hindriks, K.V., De Boer, F.S., Van Der Hoek, W., Meyer, J.J.C.: Agent programming in 3APL. Autonomous Agents and Multi-agent Systems **2**(4) (1999) 357–401
3. Dastani, M.: 2APL: a practical agent programming language. Autonomous Agents and Multi-agent Systems **16**(3) (2008) 214–248
4. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. (2007) 1–273
5. Winikoff, M.: Jack intelligent agents: An industrial strength platform. In Bordini, R.H., Dastani, M., Dix, J., El allah Seghrouchni, A., eds.: Multi-Agent Programming: Languages, Platforms and Applications. Springer (2005) 175–193
6. Busetta, P., Ronnquist, R., Hodgson, A., Lucas, A.: JACK intelligent agents - components for intelligent agents in Java. AgentLink News Letter **2** (1999) 2–5
7. Hindriks, K.V.: Programming rational agents in goal. In El Fallah Seghrouchni, A., Dix, J., Dastani, M., Bordini, R.H., eds.: Multi-Agent Programming: Languages, Tools and Applications. Springer (2009) 119–157
8. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: a multiagent simulation environment. Simulation-transactions of the Society for Modeling and Simulation International **81**(7) (2005) 517–527
9. North, M.J., Collier, N.T., Ozik, J., Tatara, E.R., Macal, C.M., Bragen, M., Sydelko, P.: Complex adaptive systems modeling with Repast Simphony. Complex Adaptive Systems Modeling **1**(1) (2013) 3
10. Amouroux, E., Chu, T.Q., Boucher, A., Drogoul, A.: GAMA: an environment for implementing and running spatially explicit multi-agent simulations. Lecture Notes in Computer Science **5044** (2009) 359–371
11. Kravari, K., Bassiliades, N.: A survey of agent platforms. Jasss-the Journal of Artificial Societies and Social Simulation **18**(1) (2015) 11
12. Adam, C., Gaudou, B.: BDI agents in social simulations: a survey. Knowledge Engineering Review **31**(3) (2016) 207–238
13. Jensen, A.S., Dignum, V.: AORTA: adding organizational reasoning to agents. Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014) **2**(3) (2014) 1493–1494
14. Larsen, J.B., Villadsen, J.: An approach for hospital planning with multi-agent organizations. In: Rough Sets: International Joint Conference, IJCRS 2017, Part II, Springer (2017) 454–465
15. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In Van de Velde, W., Perram, J.W., eds.: Agents Breaking Away: 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW '96 Eindhoven, The Netherlands, January 22–25, 1996 Proceedings. Springer (1996) 42–55
16. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: A BDI reasoning engine. In Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A., eds.: Multi-Agent Programming: Languages, Platforms and Applications. Springer (2005) 149–174
17. Siebers, P.O., Macal, C.M., Garnett, J., Buxton, D., Pidd, M.: Discrete-event simulation is dead, long live agent-based simulation! Journal of Simulation **4**(3) (2010) 204–210
18. Bordini, R.H., Hübner, J.F.: Agent-based simulation using BDI programming in Jason. In: Multi-Agent Systems: Simulation and Applications, CRC Press (2009) 451–476

19. Caballero, A., Botia, J., Gomez-Skarmeta, A.: Using cognitive agents in social simulations. Engineering Applications of Artificial Intelligence **24**(7) (2011) 1098–1109
20. Padgham, L., Scerri, D., Jayatilleke, G., Hickmott, S.: Integrating BDI reasoning into agent based modeling and simulation. Proceedings of the Winter Simulation Conference (2011) 345–356
21. Singh, D., Padgham, L., Logan, B.: Integrating BDI agents with agent-based simulation platforms. Autonomous Agents and Multi-agent Systems **30**(6) (2016) 1050–1071
22. Dignum, V., Tranier, J., Dignum, F.: Simulation of intermediation using rich cognitive agents. Simulation Modelling Practice and Theory **18**(10) (2010) 1526–1536
23. Birna Van Riemsdijk, M.: 20 years of agent-oriented programming in distributed AI: History and outlook. Splash 2012: Agere 2012 - Proceedings of the 2012 Acm Workshop on Programming Systems, Languages and Applications Based on Actors, Agents, and Decentralized Control Abstractions (2012) 7–10
24. Weiss, G.: Multiagent Systems – 2nd Edition. MIT Press (2013)
25. Ricci, A., Viroli, M., Omicini, A.: CArtAgO: a framework for prototyping artifact-based environments in MAS. Environments for Multi-agent Systems Iii. Third International Workshop, E4mas 2006. Selected Revised and Invited Papers (Lecture Notes in Artificial Intelligence Vol. 4389) (2006) 67–86
26. Piunti, M., Ricci, A., Braubach, L., Pokahr, A.: Goal-directed interactions in artifact-based MAS: Jadex agents playing in CARTAGO environments. 2008 International Conference on Intelligent Agent Technology **2** (2008) 207–213
27. Ricci, A., Bordini, R.H., Piunti, M., Hbner, J.F., Acay, L.D., Dastani, M.: Integrating heterogeneous agent programming platforms within artifact-based environments. Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems **1** (2008) 222–229
28. Dignum, V.: A Model for Organizational Interaction: based on Agents, founded in Logic. SIKS Dissertation Series 2004-1. Utrecht University (2004) PhD Thesis.
29. Aldewereld, H., Dignum, V.: OperettA: organization-oriented development environment. Lecture Notes in Computer Science **6822** (2011) 1–18
30. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the MOISE+ model: Programming issues at the system and agent levels. Int. J. Agent-Oriented Softw. Eng. **1**(3/4) (December 2007) 370–395
31. Hübner, J.F., Boissier, O., Kitio, R., Ricci, A.: Instrumenting multi-agent organisations with organisational artifacts and agents. Autonomous Agents and Multi-Agent Systems **20**(3) (2010) 369–400
32. Jensen, A.S., Dignum, V., Villadsen, J.: A framework for organization-aware agents. Autonomous Agents and Multi-Agent Systems **31**(3) (2017) 387–422
33. Jensen, A.S., Dignum, V., Villadsen, J.: The AORTA architecture: Integrating organizational reasoning in Jason. Lecture Notes in Computer Science **8758**(3) (2014) 127–145
34. Jensen, A.S.: Model checking AORTA: verification of organization-aware agents. CoRR **abs/1503.05317** (2015)