The Shaping of the Agent-Oriented Mindset
Twenty Years of Engineering MAS

Koen V. Hindriks

Delft University of Technology, EEMCS, The Netherlands

Abstract. In the past twenty years we have seen an enormous growth and development of new techniques, technologies, and tools that support the engineering of Multi-Agent Systems (MAS). The 1990s perhaps are best characterized as the period in which the foundations were laid and the more theoretical underpinnings of the MAS field were explored. Besides a continuation of this foundational work, since 2000 the agent-based community has also been increasingly able to demonstrate the great potential for applying the MAS technology that has been developed in a very broad and diverse range of application domains.

In this paper, I will trace the shaping of the agent-oriented mindset from the mid 90s on as it evolved in the work presented in the international workshops ProMAS, AOSE, and DALT that recently merged into the EMAS workshop. For this reason the focus of this overview will be in particular on *cognitive agents* as it seems fair to say that most work reported in ProMAS, AOSE, and DALT has taken its inspiration from Belief-Desire-Intention (BDI) agents.

1 Introduction

In a recent survey of applications of MAS technology [44], mature applications are reported in such diverse areas as Logistics and Manufacturing, Telecommunication, Aerospace, E-commerce, and Defense. The authors conclude that "dedicated agent platforms actually can make a difference regarding business success". They also write that "more recent platforms [...] may take some more time to mature". It was found, for example, that quite a few mature applications were built using one of the older and well-known agent platforms JADE [6]. In order to continue these successes, it is important to identify what is needed to advance more recently developed technologies for engineering MAS to a level that they can be used to engineer mature applications.

In this paper, we will focus in particular on *cognitive agent technology* as it seems fair to say that most work reported in the international workshops ProMAS, AOSE, and DALT that recently merged into the EMAS workshop has taken its inspiration from so-called Belief-Desire-Intention (BDI) agents. Arguably, the step to mature applications for technologies that support the engineering of cognitive agents and MAS is bigger than that of more general purpose frameworks for engineering agents such as JADE. One reason, moreover, why a broader uptake and the application of cognitive agent technologies has been

F. Dalpiaz et al. (Eds.): EMAS 2014, LNAI 8758, pp. 1–14, 2014.

somewhat slow perhaps, may be that this work originally has had a strong conceptual focus, aiming, for example, to relate agent frameworks to formal theories of rational agents.

To move forward it is important to learn from past successes and failures and to take stock of where we are today. To this end, the aim is to trace and to provide an overview of the agent-oriented mind-set by revisiting some of the results discussed and proposed in the past 20 years on Engineering MAS (EMAS). I will only be able here to provide a high-level overview of the past twenty years of developments related to engineering MAS and this overview thus will necessarily be far from complete and will only include some of what I consider to be its highlights. In the remainder, some of the core concepts, research goals, and achievements of twenty years of EMAS will be presented, followed by a brief perspective on future research of engineering MAS.

2 The Agent-Oriented Mindset

One perspective on what we as a research community are trying to achieve is that we are *shaping the agent-oriented mind-set*. This mind-set, among others, consists of key concepts that we use to design a multi-agent system. A lot of research has gone into clarifying and refining concepts associated with agent-based systems. In addition, to support the design and engineering of MAS using this mind-set, we have developed corresponding agent-oriented tools, techniques, and methodologies.

The agent-oriented mindset is well-established by now and it is not hard to answer the question what is part of that mind-set. A short interaction with the audience at EMAS yielded the concepts that are common and familiar by now to most MAS developers, including:

- *autonomy*,
- *environment*, *event*, *reactive*,
- *rational*, *goal-directedness*, *intentional stance*
- *decentralization*, *interaction*, and *social*.

To summarize and paraphrase a well-known definition [71], apart from being *autonomous*, an agent is *reactive*, *proactive*, and *interactive* (also known as a *weak* notion of agency).

Another defining notion in our field of research has been the notion of a *Belief-Desire-Intention (BDI) agent* [56]. The notion of a BDI agent is about the internal, cognitive structure of an agent that consists, among others, of an agent's *beliefs* and can be viewed as a refinement of a *pro-active* agent to an agent that has a *motivational state* that consists of, e.g., *desires*, *goals*, and/or *intentions*. The idea is that an agent aims at achieving something it wants and [56] therefore emphasises the *rationality* of agents instead of their autonomy. The cognitive state of an agent should, moreover, satisfy basic *rationality constraints*, e.g., goals should be compatible with the agent's beliefs, intentions should be compatible with goals, and agents should not procrastinate with respect to their intentions.

That is, agent should be *committed* to achieving their goals but should not do so blindly. Another very influential paper [55] proposed an agent programming language called AgentSpeak(L) derived from the notion of a BDI agent but which also added the concept of a *plan*. An agent in AgentSpeak(L) has a *plan library* that should provide an agent with the means to achieve its goals (see also Figure 1). Mental states have been identified by some as the essential ingredient of Agent-Oriented Programming (AOP) [13].

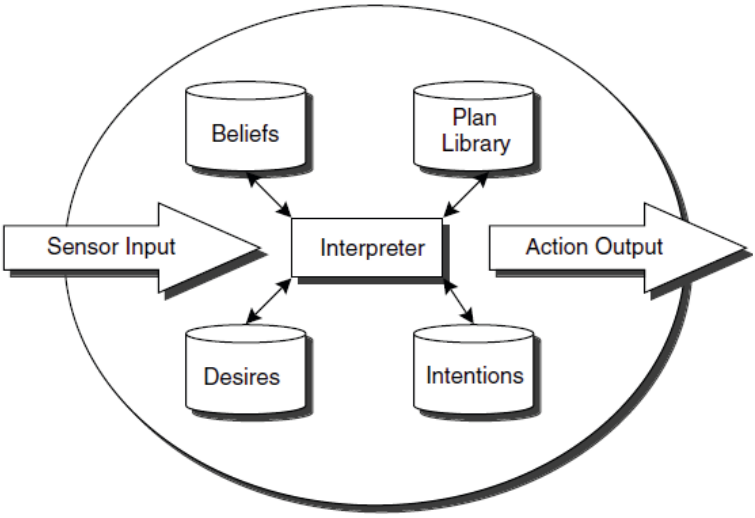

Fig. 1. Interpreter for BDI Agent

Models that formalized the notion of an agent have typically been based on some form of logic and throughout "the declarative paradigm" has been promoted within the community (and less so game theoretic models). Especially the work reported in the DALT workshop has contributed to implementation models and refinements or extensions of the notion of a BDI agent. Just to mention two examples, [2] introduced a cooperative BDI model *Coo-BDI*, and [1] presents an efficient (linear time) *belief contraction operation*.

Agents are distinctly different from other software entities such as objects because they are intrinsically motivated: agents are pro-active and aim to achieve their goals in order to meet their design objectives. It thus is not surprising to see that quite some work has focussed on the notion of a *goal*. For example, [66] studies the *dynamics of declarative goals*, [17] introduced a mechanism for *goal generation*, [38] presents an account of *goal change* that is able to handle prioritized goals and subgoals and their dynamics, whereas [70] has investigated the *interactions between goals* and provides a framework for reasoning about such interactions. Various goal types were distinguished in this work, including most importantly *achievement*, *maintenance*, and *perform* goals (see also [67]).

Important results were also obtained on the *life cycle of goals*: [12,61,67] discuss various states in different life cycle models which include, for example, the *suspension* and *abortion* of goals.

Right from the start it was recognized that agents that are part of a MAS should be somehow *organized*. An important aspect of this organization concerns the modelling of agent interaction. New models for interaction based on the notion of *commitment* rather than that of a speech act have been introduced with corresponding methods for verification based on the notion of compliance [3,7,15]. Another means to regulate the behaviour of agents is to introduce *norms* that agents should obey or comply with. Various works have looked at the notion of an *institution* with associated norms, including, for example, [68] which proposes a definition of norms for electronic institutions for synthesising *norm-aware agents*, [28] which introduces a social layer for MAS in which normative positions are explicitly represented and managed, and [27] which presents a model of norms for specifying, amongst others, *sanctions*.

Summary. The concept within the agent-oriented mindset that has been refined most over the years has been that of a *goal* whereas the notion of a *norm-aware* agent has been the most significant extension of the notion of a cognitive agent.

3 The Design of MAS

In Agent-Oriented Software Engineering (AOSE), agent interaction, not the agent's environment, was emphasized, at least initially, as a key characteristic of complex software that calls for new methods. The agent metaphor defines a new software engineering paradigm and agent metaphors and technologies are adopted to harness and govern the *complexity* of software systems. The basic idea was that the growing complexity of systems calls for new models and technologies that promote system predictability and MAS can provide a solution to this problem.

Although other methodologies were also proposed at the time (e.g., [51]), the multi-agent software engineering methodology MaSE is an early representative of work on design methodologies that is still being further developed [41]. The MaSE methodology is based on several key concepts that have remained important in AOSE, including *requirements*, *goal hierarchy*, *use cases or scenarios*, *roles*, *agents* and their *conversations*. MaSE has evolved into O-MaSE [40]. Another early well-known methodology for agent-oriented software engineering methodologies is Gaia [72]. The Gaia methodology proposed several design artifacts that the methodology required from a design of a MAS. The methodology supports the analysis and design life cycle phases but did not provide any tooling to support the design process. MASDK is an extension of Gaia [30].

The main life cycle phases that have been distinguished in the design process of a MAS include the *requirements* phase, *analysis* phase, *design* phase (sometimes a distinction is made between the architectural and detailed design phase), the *implementation* phase, and the *testing* phase. State of the art methodologies such

as O-MaSE [40], Prometheus [52], and Tropos [29] cover and provide support for all of these phases by means of design tools. These methodologies are compared with each other using a conference management case study in [53]. See [60] for a recent overview of agent-oriented methodologies.

An important contribution of work on AOSE has been the introduction of graphical notations for design specifications of agent systems. UML [8] has been taken as a starting point because it is easier to develop an agent-based extension based on the object-oriented notation, and it is relatively easy to provide high-quality tools by extending existing object-oriented tools [4,48]. Typically, however, each methodology has introduced its own notation. Some effort has been done to unify notations again [54]. It is also worthwhile to mention some of the work on *design patterns* in this context (see, e.g., [50,20]).

Several methodologies also provide dedicated support for organizational modelling. A well-known model is the AGR model [26] which stands for Agent-Group-Role. The notion of a role refers to the *constraints* (obligations, requirements, skills) that an agent needs to have to obtain a role, the *benefits* (abilities, authorization, profits) that an agent will receive in playing the role, and the *responsibilities* associated to the role. A basic assumption of the AGR approach is that the organizational model does not make any assumptions about the cognitive capabilities of the agents within the organization. The notion of a *group* is used to partition agents into units in which they can freely interact whereas different groups are assumed to be opaque to each other. Several other organizational meta-models have been proposed, including MOISE+ [34], TEAMS [36], ISLANDER [24], and OperA [49].

A topic that has gained more attention recently is *testing*. Some initial work on providing a testing framework for MAS development, including SUNIT [63] and a framework integrated with Tropos [46]. [74] provides a technique for unit testing of plan based agent systems, with a focus on the automated generation and execution of test cases.

Summary. Much has been achieved with respect to design methodologies for MAS that provide useful graphical notation for the specification of a MAS and cover all design life cycle phases, where in particular the testing phase has gained more attention only recently. In particular the concept within the agent-oriented mindset that has been refined most over the years has been that of an *organization*.

4 Programming Languages for Cognitive Agents

Various programming languages have been proposed that facilitate the implementation of MAS based on cognitive agents. We have already mentioned the AgentSpeak(L) language [55] above. Programming languages are also needed for bridging the gap between analysis and design, which yields an agent-oriented system design, and implementation of a MAS. Agent programming languages aim to provide support for a rather direct implementation of the core concepts that are part of the agent-oriented mind-set.

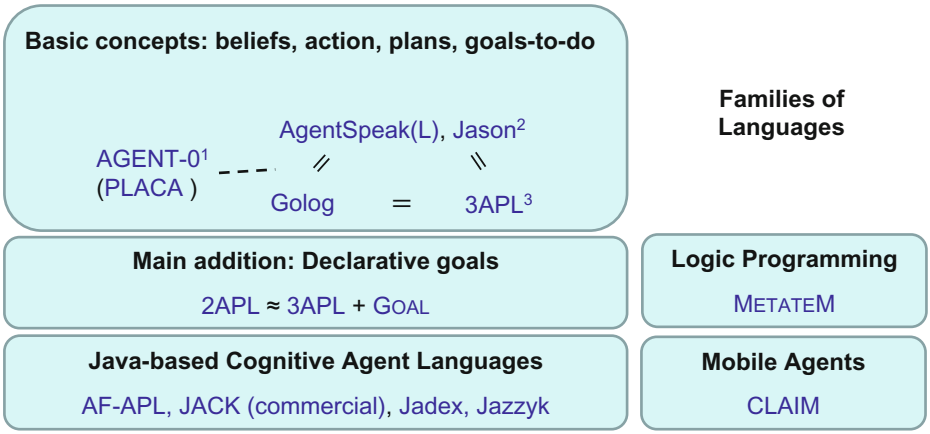

Fig. 2. Families of Agent Programming Languages

The community has been particularly productive in the area of programming frameworks for agent systems. Figure 2 provides an overview of the landscape of languages and highlights the distinction between *Java-based* and *logic-based* languages. Java-based languages stay closer to the well-known and familiar object-oriented paradigm whereas logic-based languages provide more powerful reasoning engines for reasoning about the beliefs and goals of an agent.

Early work introduced the JACK^{TM} language as an implementation of the Belief/Desire/Intention model of rational agency in Java with extensions to support the design and execution of agent systems [25] and the CLAIM language that supports the design of *mobile agents* [23]. Three other frameworks that were introduced and built on top of Java are Jadex [12], which was motivated by extending JADE with BDI agents, AF-APL [58], which was motivated by the need for a practical programming language for agent systems, and JIAC [37,42], which has been motivated by the desire to be able to meet requirements imposed by modern industrial projects. Finally, [47] presents the language Jazzyk which is motivated by the need for a clean separation between the knowledge representational and the behavioural level of an agent program. The work [19] incorporates the notion of a *declarative goal* into the agent programming language 3APL [33].

An important contribution of work on agent programming languages has been the introduction of *modules* that support modular design of agent programs. In [11] a module concept is presented that is based on the capability concept for structuring BDI agents in functional clusters introduced before [14] that supports a higher degree of *reusability*. In [31] and [43], respectively, the logic-based agent languages GOAL [32] and *Jason* [9] are extended with modules. Another approach for adding structure to a MAS program based on the notion of an *organization* is introduced in [62].

Substantial work has also been done in the area of *debugging MAS*. The Tracing method proposed in [39] assists a programmer in debugging agents by explaining the actual agent behaviour in the implemented system. The method logs

actual agent behaviour from which it derives interpretations in terms of, e.g., the beliefs, goals, and intentions of an agent. [10] proposes the use of data mining to assist during the debugging of MAS. [16] describes how debugging has been supported for the Agent Factory Agent Programming Language (AF-APL). [18] proposes an assertion language for specifying the cognitive and temporal behaviour of an agent program as support for debugging.

The integration of sophisticated AI techniques into agent systems has mainly been looked at in the context of agent-oriented programming. A *planner* is integrated into Jadex for providing dynamic plans at runtime [69]. The integration approach used is one where the cognitive agent takes responsibility for plan monitoring and re-planning and only the responsibility for the creation of plans is delegated to the planner. Recently also work on integrating *learning* into the agent programming language GOAL has been reported in [59]. The focus in this paper is on improving action selection in rule-based agent programming languages using a reinforcement learning mechanism under the hood.

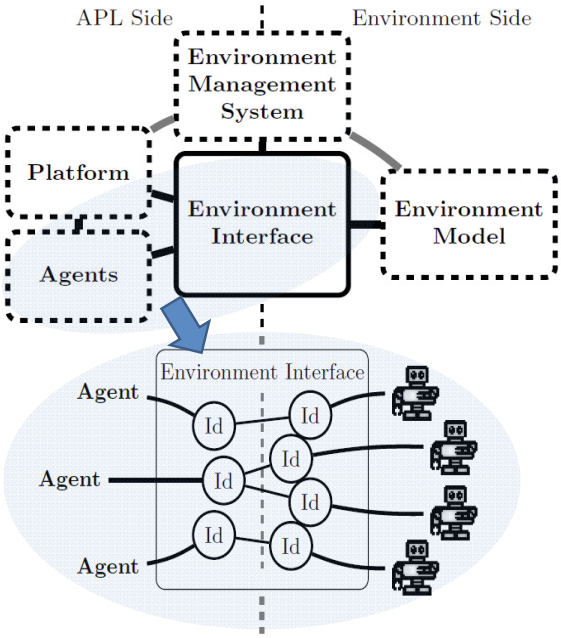

Fig. 3. Environment Interface for Agent Interaction with Environments

One important feature of agent systems has not yet been discussed: agent systems are embedded in and agent systems *interact with an environment*. Various models that support the interaction between agents and their environments have been proposed. The Agents and Artifacts (A&A) model of environments is based on the idea that an environment is composed of different sorts of artifacts that are shared and used by agents to support their activities[57].

The PRESAGE model introduced in [45] proposes the use of environments as a rapid prototyping tool for agent societies. The CIGA middleware proposed in [64] aims at facilitating the coupling between a MAS and a game engine. Finally, the Environment Interface Standard (EIS) introduced in [5] provides support for connecting agent platforms to environments such as games (see also [22] for a range of environment implementations that have been made available). The EIS interface provides generic functionality for executing actions and for perceiving changes in an agent's environment and also provides support for managing an environment, e.g., for starting, pausing and terminating it (see also Figure 3).

Summary. Various programming language that support the agent-oriented paradigm have been proposed. Several extensions such as the notion of modular programming have made these languages more useful in practice. Work on debugging agent programs has also contributed to this end. An important contribution has also been the development of several models that support the interaction of an agent with its environment.

5 Conclusion

Cognitive agent technology offers a powerful solution for developing the next generation autonomous decision-making systems. To make this happen it is important to continue to promote and contribute to the agent-oriented mindset. It also continues to be important to justify the need for a paradigm shift from existing paradigms such as the object- or service-oriented paradigms to the agent-oriented paradigm ([60]; see also [35]). In particular, it would be useful to be able to perform quantitative assessments and comparisons of the agent-based paradigm with other paradigms ([73]; see also [21]).

We also want to suggest that it is time to start paying more attention to the kind of support that a MAS developer needs to facilitate him or her when engineering future MAS applications (see also [21,65]). It is important to identify the needs of a developer and make sure that a developer is provided with the right tools for engineering MAS. For the same reason we should focus more on issues related to *ease of use*, *scalability and performance*, and *testing*. As we have seen, work on techniques and tools that support the testing phase has only quite recently produced more concrete results (see also [60]).

There are also promises of the agent-oriented paradigm that are still to be realized. As argued in [35], "agents are the right abstraction to (re-)integrate various AI sub-disciplines together again". Robots should come to mind here. Can we provide tools and techniques that facilitate the integration of sophisticated AI techniques into agents? As a community, we can provide an important contribution by focusing on understanding how to provide programmers with easy access to such techniques. We have seen that already some proposals have been made to re-integrated planning and learning. Similarly, it remains to be shown that agent-orientation can solve key concurrency and distributed computing issues. If agents are advocated as the next generation model for engineering complex,

distributed systems, we should be able to demonstrate the added value of agent systems.

Finally, it seems particularly worthwhile to put more effort into integrating agent-based methodologies and programming languages. There are several areas of clear overlap where both can reinforce and improve their respective results, e.g., in the area of testing and the area of organizational modelling. In any case, to stimulate the adoption of cognitive agent technology and MAS, we need to provide methods and tools that jointly support the agent-oriented mindset.

References

1. Alechina, N., Jago, M., Logan, B.: Resource-bounded belief revision and contraction. In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) DALT 2005. LNCS (LNAI), vol. 3904, pp. 141–154. Springer, Heidelberg (2006)
2. Ancona, D., Mascardi, V.: Coo-BDI: Extending the BDI model with cooperativity. In: Leite, J., Omicini, A., Sterling, L., Torroni, P. (eds.) DALT 2003. LNCS (LNAI), vol. 2990, pp. 109–134. Springer, Heidelberg (2004)
3. Baldoni, M., Baroglio, C., Marengo, E.: Commitment-Based Protocols with Behavioral Rules and Correctness Properties of MAS. In: Omicini, A., Sardina, S., Vasconcelos, W. (eds.) DALT 2010. LNCS (LNAI), vol. 6619, pp. 60–77. Springer, Heidelberg (2011)
4. Bauer, B., Müller, J.P., Odell, J.J.: Agent UML: A Formalism for Specifying Multiagent Software Systems. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 91–103. Springer, Heidelberg (2001)
5. Behrens, T.M., Hindriks, K.V., Dix, J.: Towards an environment interface standard for agent platforms. Annals of Mathematics and Artificial Intelligence 61(4), 261–295 (2011)
6. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley (2007)
7. Bentahar, J., Moulin, B., Meyer, J.-J.C.: A tableau method for verifying dialogue game protocols for agent communication. In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) DALT 2005. LNCS (LNAI), vol. 3904, pp. 223–244. Springer, Heidelberg (2006)
8. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison Wesley Longman Publishing Co., Inc., Redwood City (1999)
9. Bordini, R.H., Hübner, J.F., Wooldridge, M.J.: Programming multi-agent systems in AgentSpeak using Jason, vol. 8. John Wiley & Sons (2007)
10. Botía, J.A., Hernansáez, J.M., Gómez-Skarmeta, A.F.: On the application of clustering techniques to support debugging large-scale multi-agent systems. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2006. LNCS (LNAI), vol. 4411, pp. 217–227. Springer, Heidelberg (2007)
11. Braubach, L., Pokahr, A., Lamersdorf, W.: Extending the capability concept for flexible bdi agent modularization. In: Bordini, R.H., Dastani, M. M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2005. LNCS (LNAI), vol. 3862, pp. 139–155. Springer, Heidelberg (2006)
12. Braubach, L., Pokahr, A., Moldt, D., Lamersdorf, W.: Goal Representation for BDI Agent Systems. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2004. LNCS (LNAI), vol. 3346, pp. 44–65. Springer, Heidelberg (2005)

13. Burkhard, H.-D.: Agent-oriented programming for open systems. In: Wooldridge, M.J., Jennings, N.R. (eds.) ECAI/ATAL 1994. LNCS, vol. 890, pp. 291–306. Springer, Heidelberg (1995)
14. Busetta, P., Howden, N., Rönnquist, R., Hodgson, A.: Structuring BDI Agents in Functional Clusters. In: Jennings, N.R. (ed.) Intelligent Agents VI. LNCS (LNAI), vol. 1757, pp. 277–289. Springer, Heidelberg (2000)
15. Chopra, A.K., Singh, M.P.: Producing compliant interactions: Conformance, coverage, and interoperability. In: Baldoni, M., Endriss, U. (eds.) DALT 2006. LNCS (LNAI), vol. 4327, pp. 1–15. Springer, Heidelberg (2006)
16. Collier, R.: Debugging agents in agent factory. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) ProMAS 2006. LNCS (LNAI), vol. 4411, pp. 229–248. Springer, Heidelberg (2007)
17. da Costa Pereira, C., Tettamanzi, A.G.B.: Goal Generation from Possibilistic Beliefs Based on Trust and Distrust. In: Baldoni, M., Bentahar, J., van Riemsdijk, M.B., Lloyd, J. (eds.) DALT 2009. LNCS (LNAI), vol. 5948, pp. 35–50. Springer, Heidelberg (2010)
18. Dastani, M., Brandsema, J., Dubel, A., Meyer, J.-J.C.: Debugging BDI-Based Multi-Agent Programs. In: Braubach, L., Briot, J.-P., Thangarajah, J. (eds.) ProMAS 2009. LNCS (LNAI), vol. 5919, pp. 151–169. Springer, Heidelberg (2010)
19. Dastani, M., van Riemsdijk, M.B., Dignum, F., Meyer, J.-J.C.: A Programming Language for Cognitive Agents Goal Directed 3APL. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) PROMAS 2003. LNCS (LNAI), vol. 3067, pp. 111–130. Springer, Heidelberg (2004)
20. De Wolf, T., Holvoet, T.: Design patterns for decentralised coordination in self-organising emergent systems. In: Brueckner, S.A., Hassas, S., Jelasity, M., Yamins, D. (eds.) ESOA 2006. LNCS (LNAI), vol. 4335, pp. 28–49. Springer, Heidelberg (2007)
21. Dix, J., Hindriks, K.V., Logan, B., Wobcke, W.: Engineering Multi-Agent Systems (Dagstuhl Seminar 12342). Dagstuhl Reports 2(8), 74–98 (2012)
22. The Environment Interface (September 2014), `https://github.com/eishub`
23. El Fallah-Seghrouchni, A., Suna, A.: CLAIM: A computational language for autonomous, intelligent and mobile agents. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) PROMAS 2003. LNCS (LNAI), vol. 3067, pp. 90–110. Springer, Heidelberg (2004)
24. Esteva, M., De La Cruz, D., Sierra, C.: ISLANDER: An electronic institutions editor. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: part 3, pp. 1045–1052. ACM (2002)
25. Evertsz, R., Fletcher, M., Frongillo, R., Jarvis, J., Brusey, J., Dance, S.: Implementing industrial multi-agent systems using JACK^{TM}. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) PROMAS 2003. LNCS (LNAI), vol. 3067, pp. 18–48. Springer, Heidelberg (2004)
26. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multi-agent systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
27. Fornara, N., Colombetti, M.: Specifying and enforcing norms in artificial institutions: A retrospective review. In: Sakama, C., Sardina, S., Vasconcelos, W., Winikoff, M. (eds.) DALT 2011. LNCS (LNAI), vol. 7169, pp. 117–119. Springer, Heidelberg (2012)

28. García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., Vasconcelos, W.: A distributed architecture for norm-aware agent societies. In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) DALT 2005. LNCS (LNAI), vol. 3904, pp. 89–105. Springer, Heidelberg (2006)
29. Giorgini, P., Mylopoulos, J., Perini, A., Susi, A.: The Tropos methodology and software development environment. In: Social Modeling for Requirements Engineering, pp. 405–423 (2010)
30. Gorodetsky, V., Karsaev, O., Samoylov, V., Konushy, V.: Support for Analysis, Design, and Implementation Stages with MASDK. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 272–287. Springer, Heidelberg (2009)
31. Hindriks, K.V.: Modules as Policy-Based Intentions: Modular Agent Programming in GOAL. In: Dastani, M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS (LNAI), vol. 4908, pp. 156–171. Springer, Heidelberg (2008)
32. Hindriks, K.V.: Programming Rational Agents in GOAL. In: El Fallah Seghrouchni, A., Dix, J., Dastani, M., Bordini, R.H. (eds.) Multi-Agent Programming, pp. 119–157. Springer US (2009)
33. Hindriks, K.V., De Boer, F.S., Van der Hoek, W., Meyer, J.-J.C.: Meyer. Agent Programming in 3APL. Autonomous Agents and Multi-Agent Systems 2(4), 357–401 (1999)
34. 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), 370–395 (2007)
35. Jennings, N.R.: Agent-oriented software engineering. In: Imam, I., Kodratoff, Y., El-Dessouki, A., Ali, M. (eds.) IEA/AIE 1999. LNCS (LNAI), vol. 1611, pp. 4–10. Springer, Heidelberg (1999)
36. Kaminka, G.A., Pynadath, D.V., Tambe, M.: Monitoring teams by overhearing: A multi-agent plan-recognition approach. Journal of Artificial Intelligence Research 17(1), 83–135 (2002)
37. Keiser, J., Hirsch, B., Albayrak, Ş.: Agents do it for money - accounting features in agents. In: Dastani, M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS (LNAI), vol. 4908, pp. 42–56. Springer, Heidelberg (2008)
38. Khan, S.M., Lespérance, Y.: Prioritized goals and subgoals in a logical account of goal change: A preliminary report. In: Baldoni, M., Bentahar, J., van Riemsdijk, M.B., Lloyd, J. (eds.) DALT 2009. LNCS (LNAI), vol. 5948, pp. 119–136. Springer, Heidelberg (2010)
39. Lam, D.N., Barber, K.S.: Debugging agent behavior in an implemented agent system. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2004. LNCS (LNAI), vol. 3346, pp. 104–125. Springer, Heidelberg (2005)
40. De Loach, S.A., Garcia-Ojeda, J.C.: O-MaSE: A customisable approach to designing and building complex, adaptive multi-agent systems. International Journal of Agent-Oriented Software Engineering 4(3), 244–280 (2010)
41. De Loach, S.A., Wood, M.: Developing Multiagent Systems with agentTool. In: Castelfranchi, C., Lespérance, Y. (eds.) Intelligent Agents VII. LNCS (LNAI), vol. 1986, pp. 46–60. Springer, Heidelberg (2001)
42. Lützenberger, M., Küster, T., Konnerth, T., Thiele, A., Masuch, N., Heßler, A., Keiser, J., Burkhardt, M., Kaiser, S., Albayrak, S.: JIAC V: A MAS Framework for Industrial Applications. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS 2013, Richland, SC, pp. 1189–1190. International Foundation for Autonomous Agents and Multiagent Systems (2013)

43. Madden, N., Logan, B.: Modularity and Compositionality in Jason. In: Braubach, L., Briot, J.-P., Thangarajah, J. (eds.) ProMAS 2009. LNCS (LNAI), vol. 5919, pp. 237–253. Springer, Heidelberg (2010)
44. Müller, J.P., Fischer, K.: Application Impact of Multi-Agent Systems and Technologies: A Survey. In: Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks. Springer (2014)
45. Neville, B., Pitt, J.: PRESAGE: A Programming Environment for the Simulation of Agent Societies. In: Hindriks, K.V., Pokahr, A., Sardina, S. (eds.) ProMAS 2008. LNCS (LNAI), vol. 5442, pp. 88–103. Springer, Heidelberg (2009)
46. Nguyen, D.C., Perini, A., Tonella, P.: A Goal-Oriented Software Testing Methodology. In: Luck, M., Padgham, L. (eds.) AOSE 2007. LNCS, vol. 4951, pp. 58–72. Springer, Heidelberg (2008)
47. Novák, P.: Jazzyk: A Programming Language for Hybrid Agents with Heterogeneous Knowledge Representations. In: Hindriks, K.V., Pokahr, A., Sardina, S. (eds.) ProMAS 2008. LNCS (LNAI), vol. 5442, pp. 72–87. Springer, Heidelberg (2009)
48. Odell, J.J., Van Dyke Parunak, H., Bauer, B.: Representing agent interaction protocols in UML. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 121–140. Springer, Heidelberg (2001)
49. Okouya, D., Dignum, V.: OperettA: A Prototype Tool for the Design, Analysis and Development of Multi-agent Organizations. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Demo Papers, AAMAS 2008, Richland, SC, pp. 1677–1678. International Foundation for Autonomous Agents and Multiagent Systems (2008)
50. Oluyomi, A., Karunasekera, S., Sterling, L.: An agent design pattern classification scheme: Capturing the notions of agency in agent design patterns. In: 11th Asia-Pacific on Software Engineering Conference, pp. 456–463 (November 2004)
51. Omicini, A.: SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 185–193. Springer, Heidelberg (2001)
52. Padgham, L., Luck, M.: Prometheus: A practical agent-oriented methodology. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-oriented Methodologies, pp. 107–135. Idea Group Inc., Hershey (2005)
53. Padgham, L., Luck, M.: Introduction to AOSE tools for the conference management system. In: Luck, M., Padgham, L. (eds.) AOSE 2007. LNCS, vol. 4951, pp. 164–167. Springer, Heidelberg (2008)
54. Padgham, L., Winikoff, M., DeLoach, S., Cossentino, M.: A Unified Graphical Notation for AOSE. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 116–130. Springer, Heidelberg (2009)
55. Rao, A.S.: Agentspeak(l): BDI agents speak out in a logical computable language. In: Van de Velde, W., Perram, J. (eds.) MAAMAW 1996. LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg (1996)
56. Rao, A.S., Georgeff, M.P.: Modeling Rational Agents within a BDI-Architecture. In: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR 1991), Cambridge, MA, USA, April 22-25, pp. 473–484 (1991)
57. Ricci, A., Viroli, M., Omicini, A.: The A&A Programming Model and Technology for Developing Agent Environments in MAS. In: Dastani, M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS (LNAI), vol. 4908, pp. 89–106. Springer, Heidelberg (2008)

58. Ross, R.J., Collier, R., O'Hare, G.M.P.: AF-APL: Bridging Principles and Practice in Agent Oriented Languages. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2004. LNCS (LNAI), vol. 3346, pp. 66–88. Springer, Heidelberg (2005)
59. Singh, D., Hindriks, K.V.: Learning to Improve Agent Behaviours in GOAL. In: Dastani, M., Hübner, J.F., Logan, B. (eds.) ProMAS 2012. LNCS (LNAI), vol. 7837, pp. 158–173. Springer, Heidelberg (2013)
60. Sturm, A., Shehory, O.: The landscape of agent-oriented methodologies. In: Shehory, O., Sturm, A. (eds.) Agent-Oriented Software Engineering, pp. 137–154. Springer, Heidelberg (2014)
61. Thangarajah, J., Harland, J., Morley, D., Yorke-Smith, N.: Operational behaviour for executing, suspending, and aborting goals in bdi agent systems. In: Omicini, A., Sardina, S., Vasconcelos, W. (eds.) DALT 2010. LNCS (LNAI), vol. 6619, pp. 1–21. Springer, Heidelberg (2011)
62. Tinnemeier, N.A.M., Dastani, M., Meyer, J.-J.C.: Orwell's Nightmare for Agents? Programming Multi-agent Organisations. In: Hindriks, K.V., Pokahr, A., Sardina, S. (eds.) ProMAS 2008. LNCS (LNAI), vol. 5442, pp. 56–71. Springer, Heidelberg (2009)
63. Tiryaki, A.M., Öztuna, S., Dikenelli, O., Erdur, R.C.: SUNIT: A Unit Testing Framework for Test Driven Development of Multi-Agent Systems. In: Padgham, L., Zambonelli, F. (eds.) AOSE 2006. LNCS, vol. 4405, pp. 156–173. Springer, Heidelberg (2007)
64. van Oijen, J., La Poutré, H., Dignum, F.: Agent perception within CIGA: Performance optimizations and analysis. In: Müller, J.P., Cossentino, M. (eds.) AOSE 2012. LNCS, vol. 7852, pp. 99–117. Springer, Heidelberg (2013)
65. van Riemsdijk, M.B.: 20 Years of Agent-oriented Programming in Distributed AI: History and Outlook. In: Proceedings of the 2nd Edition on Programming Systems, Languages and Applications Based on Actors, Agents, and Decentralized Control Abstractions, AGERE! 2012, pp. 7–10. ACM, New York (2012)
66. van Riemsdijk, M.B., Dastani, M., Dignum, F.P.M., Meyer, J.-J.C.: Dynamics of Declarative Goals in Agent Programming. In: Leite, J., Omicini, A., Torroni, P., Yolum, p. (eds.) DALT 2004. LNCS (LNAI), vol. 3476, pp. 1–18. Springer, Heidelberg (2005)
67. van Riemsdijk, M.B., Dastani, M., Winikoff, M.: Goals in Agent Systems: A Unifying Framework. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 200, Richland, SC, vol. 2, pp. 713–720. International Foundation for Autonomous Agents and Multiagent Systems (2008)
68. Vasconcelos, W.W.: Norm verification and analysis of electronic institutions. In: Leite, J., Omicini, A., Torroni, P., Yolum, p. (eds.) DALT 2004. LNCS (LNAI), vol. 3476, pp. 166–182. Springer, Heidelberg (2005)
69. Walczak, A., Braubach, L., Pokahr, A., Lamersdorf, W.: Augmenting BDI Agents with Deliberative Planning Techniques. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2006. LNCS (LNAI), vol. 4411, pp. 113–127. Springer, Heidelberg (2007)
70. Winikoff, M.: An Integrated Formal Framework for Reasoning about Goal Interactions. In: Sakama, C., Sardina, S., Vasconcelos, W., Winikoff, M. (eds.) DALT 2011. LNCS (LNAI), vol. 7169, pp. 16–32. Springer, Heidelberg (2012)
71. Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. The Knowledge Engineering Review 10, 115–152 (1995)

72. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3(3), 285–312 (2000)
73. Zambonelli, F., Omicini, A.: Challenges and research directions in agent-oriented software engineering. Autonomous Agents and Multi-Agent Systems 9(3), 253–283 (2004)
74. Zhang, Z., Thangarajah, J., Padgham, L.: Automated testing for intelligent agent systems. In: Gomez-Sanz, J.J. (ed.) AOSE 2009. LNCS, vol. 6038, pp. 66–79. Springer, Heidelberg (2011)