Chapter 5
Agents and Multi-Agent Systems

Marie-Pierre Gleizes, Valérie Camps, Anthony Karageorgos, and Giovanna Di Marzo Serugendo

Objectives This chapter details the main concepts of the multi-agent systems domain. After reading this chapter, the reader will:

- Know what is an agent and what are its properties,
- Understand what is a MAS and what are its properties,
- Be able to explain what is the environment of a MAS,
- Understand how MASs represent a solution for artificial self-organising systems design.

5.1 Introduction

Complexity of near future and even nowadays applications can be characterised as a combination of aspects such as the great number of components taking part in the applications, the fact that knowledge and control have to be distributed, the presence of nonlinear processes in the system, the fact that the system is more and more

M.-P. Gleizes · V. Camps (✉)
IRIT, Université Paul Sabatier, Toulouse, France
e-mail: camps@irit.fr

M.-P. Gleizes
e-mail: Marie-Pierre.Gleizes@irit.fr

A. Karageorgos
Technological Educational Institute of Larissa, Larissa, Greece
e-mail: karageorgos@computer.org

G. Di Marzo Serugendo
Birkbeck College, University of London, London, UK
e-mail: dimarzo@dcs.bbk.ac.uk

G. Di Marzo Serugendo et al. (eds.), *Self-organising Software*,
Natural Computing Series,
DOI 10.1007/978-3-642-17348-6_5,

often open, its environment dynamic and the interactions unpredictable. In order to tackle the design of such complex systems, being able to engineer self-organising systems is a promising approach providing the required robustness. MASs are one of the most representatives among artificial systems dealing with *complexity* and *distribution* [20, 22]. The MAS paradigm appeared in the 1980s, and its specificity concerns collective behaviour. The first generation of works (1970–1980) focuses on distributed problem solving; in general, in these systems, knowledge and processing are distributed, but the control is centralised; furthermore, these systems are ad hoc ones with no reusability potential. The second generation (1980–1990) studies systems with decentralised control and generic systems to increase reusability. Interactions were the heart of most of the works on MASs during the third generation (1990–2000). The current generation is interested in the environment, the openness and the dynamics of these systems and the MAS technique is a way to design self-organising systems.

5.2 Agents

This section defines the agent concept and its main properties.

5.2.1 Agent Definition

An agent is a physical or software (virtual) entity [8, 24] which is:

- autonomous,
- situated in an environment and able to act in/on it,
- having a partial representation of its environment,
- able to communicate with others agents,
- having an individual objective/satisfaction function,
- having resources,
- able to perceive its environment,
- having skills and offering services.

Its behaviour is the consequence of its perceptions, knowledge, beliefs, skills, intentions, interactions, ...

An agent has the following life cycle:

- perception: the agent perceives new information coming from its environment,
- decision: the agent chooses the action(s) it has to do,
- action: the agent acts, it performs the action(s) chosen during the previous step.

Some examples of agents are: a user assistant in a system of information retrieval, an expert in a diagnosis system, a player in a soccer team, an ant in an anthill, ...

5.2.2 Agent Properties

5.2.2.1 Agent Autonomy

Autonomy is the most important property of an agent. This autonomy may qualify the existence of the agent with the property of viability: an agent is able to maintain some of its parameters in a given range. For example, a robot can maintain its energy at a given level in order to continue to "live". Autonomy may be autonomy of life: an agent is able to live independently of the other agents' existence. For example, in a collective of identical robots which have to sort boxes in a room, even if one robot is out of service, the others can continue to do their task.

But for an artificial agent, the most important significance of autonomy is the autonomy of control of decision. An agent can say "NO", which means that it can decide alone what its next action will be and to do or not an action. The agent is able to make its decision in an internal way according to its perceptions, knowledge and belief. Note that the agent must at least be able to perceive and to act to be qualified as autonomous. Depending on the application, an agent can be more or less autonomous, so the answer to the question "is this agent autonomous?" is not "yes" or "no" but "more" or "less". For example, in the army a common soldier is less autonomous than a military officer, but he has some kind of autonomy. What is important is that without autonomy, an entity cannot be qualified as an agent. For example, a chair is not an autonomous agent because it is a passive entity without perception and action abilities.

For an agent which can move inside its environment, its autonomy can be to decide alone if it has to move or not: the agent can say "GO".

5.2.2.2 Reactive Agent

A *reactive* agent [24] is an agent which is able to react to changes inside its environment and can maintain interactions with it. In general, it reacts in order to reach its own objective. The behaviour of this kind of agent is typically called a reflex behaviour. For example, an ant is the typical example of a reactive agent. Its behaviour is a set of conditions-actions rules such as when the conditions of a rule are satisfied by the environment perception and/or by its internal state, the actions are launched. An ant will collect food if it perceives food near itself inside its environment.

5.2.2.3 Proactive Agent

A *proactive* agent is able to generate and reach goals. He is not only event-driven but also goal-driven, as he can initiate means to achieve its goals [24]. For example, an assistant agent in an E-commerce system, may decide to drop the goal "buy the object O" because the price is too high and may decide to buy another object.

5.2.2.4 Social Agent

A *social* agent is an agent being aware of other agents and able to reason about them and to interact with them. In a MAS, all agents have to be social.

5.2.2.5 Communicating Agent

A *communicating* agent is an agent which interacts with other agents by sending and receiving messages. A lot of works have been carried out to study languages such as KQML [12], ACL-FIPA [7] and protocols [14].

5.2.2.6 Situated Agent

A *situated* agent is an agent which interacts with other agents through the environment and interacts with the environment by modifying it (e.g., by moving, by collecting resources, ...).

5.2.2.7 Adaptive Agent

An *adaptive* agent is an agent which can modify its behaviour during its life. Such an agent needs to have some memory in order to learn its future behaviour. Learning can be done with classical methods such as bucket brigade, reinforcement, genetic algorithms, ... In general, in MAS, the learning phase in an agent takes into account the other agents.

5.2.2.8 Mobile Agent

The different agents which can be designed may possess some of these properties. These properties are not always exclusive; for example, a situated agent may also send messages and may also be a communicating agent. Therefore, the different kinds of agents to be designed are very large.

5.2.3 Agent Architectures

Agents always have three main modules:

- The perception module is in charge of taking into account events coming from the agent's environment, which depends on the way the agent interacts with its environment. For communicating agents, it may be a mailbox to receive messages from other agents, or, for situated agents, it may be results of sensors detection.

- The decision module is composed of knowledge and decision support tools for enabling the agent to execute its task or to reach its objective. For example, the decision module of a reactive agent is a program composed of condition–action rules. For a cognitive agent, this module can be an expert system with an inference engine and a knowledge base.
- The action module manages all the activities to be done according to the results of the reasoning process. This is what can be externally observed as the behaviour of this agent. In systems where agents exchange messages, an action consists in messages sending. In systems where the agents have effectors, actions may be the results of actions done by an effector on its environment.

Depending on the properties of an agent, these modules may differ. Three main classes of agent architectures exist:

- Reactive architectures, an example of such an architecture is the eco-resolution architecture of reactive agents [10].
- Cognitive architectures, an example of this kind of architecture is the Belief-Desire-Intention architecture of intentional agents [17].
- Hybrid architectures, architectures which merge cognitive and reactive levels in the same agent, an example is the system.

More details about these architectures can be found in [20]. It is not easy to find the frontier between reactive and cognitive. Some architectures are clearly reactive, some others are cognitive, but some are more or less reactive or cognitive.

5.2.4 Agents and Objects

Objects are programming entities which have obvious similarities with agents but also main differences. Like objects, agents have an internal state and modular behavioural units (methods for objects, competencies or skills for agents). Both objects and agents may act to modify their state, and they communicate by message passing.

One of the most important differences concerns the autonomy of control. In the agent case, it is the receiver of a request (messages) which decides to execute or not the received request. It is the receiver which has the power to decide, whereas in the object case, it is the sender which takes the decision: the object which receives a method call, executes it.

Environment is a first-class concept in MAS and is less important for objects.

In object programming, no guide is available to implement properties of reactivity, proactivity, sociability of the agent, as interaction complexity and system dynamics. This capability to help the designer for embedding agents with these properties is called flexibility by Wooldridge and Ciancarini [23]. The object-oriented methodologies—because of the differences between objects and agents—are not directly usable for MASs development. It is why new models, new methodologies and new tools adapted to the agent concept have been developed. The relations (like “is-a” or “is-part-of”) are insufficient to model the organisational relations of

a complex system. Agents have a very significant social part, which affects their behaviour. It is not the case with objects. Because of their sociability, agents communicate. In object-programming, communications consist only in calling a method. In MASs, interactions between agents are richer (with messages typology, ontology) and complex to implement (with protocols). Moreover, an agent analyses messages and decides of its behaviour. Finally, interactions between objects are defined in a more rigid way. They cannot evolve with time, what is essential in some multi-agent applications.

MASs have shown their adequacy for designing (logically or physically) distributed, complex and robust applications. The agent concept is currently more than an efficient technology, it represents a new concept for software design where the agent is an autonomous software which pursues an objective, which is situated inside an environment and interacts with other agents with protocol languages. Often, an agent is considered as an "intelligent" object or as an abstraction level above objects and components.

Agents and objects are paradigms to help software systems designers. Because the agent concept is at a higher level of abstraction than the object one, it facilitates the specification and the design of complex applications. Programming of these systems is generally done by using common object languages, but some agent-programming languages are now well studied.

5.3 Multi-Agent Systems

This section presents the concept of multi-agent system and its main properties.

5.3.1 Definition

A Multi-Agent System (MAS) is defined as a set of interacting agents in a common environment in order to solve a common, coherent task. These agents try to achieve individual objectives which are sometimes conflicting.

There are three main classes of MAS:

- MAS in interaction with users. In general, in these systems, an agent is the user assistant and represents the user. The difficulties lie in having the right model of the user and in facilitating the interaction between the system and the user.
- Simulation with MAS. Simulation aims at modelling natural phenomena in order to predict and to understand them. Some entities in a simulation are designed as agents, and they interact with one another and with the environment. The global task of the system is not precisely specified, the system has to model the real system and to behave as the real system. The most difficult task of a designer is to adjust the parameters of the agents in order to observe the desired global behaviour.

- Collective solving problem. Agents in this kind of systems have to solve a common global task. For doing this, they usually have local knowledge and representations, partial skills and/or resources. They therefore need to cooperate in order to solve the global problem.

Note that sometimes problem solving can be done by simulation. Therefore, for some applications, the two last classes have the same purpose.

MASs are well adapted for applications with inherent distribution such as:

- *Spatial distribution.* Agents must be situated at different locations in an environment. For example, in traffic control, if there is an agent per vehicle, these agents are situated at different locations.
- *Functional/logical distribution.* Agents represent different skills and play different roles. For example, in an E-commerce application, some agents play the customers, and the others play providers.
- *Semantic distribution.* Agents use different vocabularies and ontologies. For example, on the Internet, agents from different communities use different languages and have to interact.

There are several representative applications of MASs such as: prey–predator systems, E-commerce, information retrieval, manufacturing control, soccer team (Robocup), crisis management systems, social animal simulations, ...

5.3.2 Multi-Agent Systems Properties

5.3.2.1 Autonomous System

A MAS is *autonomous* means that there is no external entity which controls this system. This property is enforced because agents inside the system are autonomous.

5.3.2.2 Distribution, Decentralisation

Inside a MAS, *data* (*knowledge*) *are distributed* inside all its agents. The main kinds of knowledge are:

- knowledge related to skills (needed for the application),
- knowledge related to world representation, belief that is: representation of the system environment, representation of the agents.

Moreover, *the control is decentralised*, there is no supervisor.

5.3.2.3 Asynchronism, Parallelism

Asynchronism and parallelism characterise the execution of the agents inside the system. *Asynchronous* means that agents do not wait an answer to their request,

they may go on working; otherwise they are synchronous. *Parallel* means that all agents can act simultaneously and not one after the other in a predetermined order; otherwise, it is sequential. Inside a MAS, the execution of the agents is generally asynchronous and parallel. At this point, note that it is important to highlight that there are two levels for specifying a MAS:

- the conceptual level: the MAS model must be done by designers asynchronous and parallel,
- the implementation level: the MAS can be program in a simulated asynchronous and parallel way or in real asynchronous and parallel programming.

5.3.2.4 Open/Closed System

An *open* MAS is a MAS in which agents can be removed (killed or suicide committed) at runtime. Moreover, these new agents can be designed by different designers. Otherwise the system is qualified as closed. For example, in an E-commerce system composed of customers and providers, customers (represented by agents) may occur or disappear in the system while the system continues to satisfy the other customers and providers. So, this kind of system is open. An example of closed MAS is an anthill simulation where the ants are created at the beginning of the simulation and cannot be created during the simulation.

5.3.2.5 Heterogeneous/Homogeneous

Heterogeneity and homogeneity of a system can be defined at different levels; in this chapter they will be defined at the agent architecture level. If in a system all agents have the same architecture, then the system is *homogeneous*. For example, an anthill, where the ants inherit of the same skills and perceptions means, is a homogeneous system. If the architecture of the agents inside the system differs, then this system is heterogeneous. For example, if in a same system a humanoid robot and an artificial ant have to cooperate, the system is qualified as heterogeneous. In a heterogeneous system, interoperability becomes one of the most important challenge.

5.4 Environment

The environment clearly appears as an essential concept to manage self-organisation of MASs. “Without an environment, an agent is effectively useless. Cut off from the rest of its world, the agent can neither sense nor act” [15]. This citation is also true for a system; the system needs to interact with its environment. Because the environment disturbs the system, this one can react by self-organising.

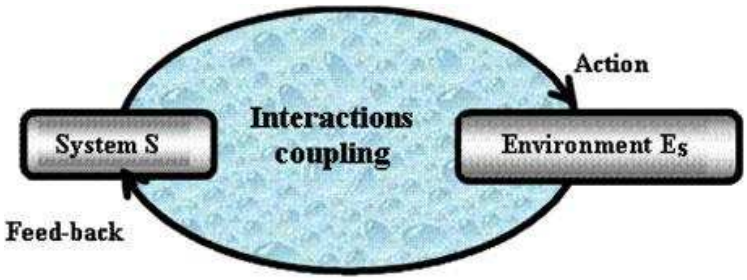

Fig. 5.1 Reciprocal influences between a system S and its environment ES

5.4.1 Environment Definition

As Weyns et al. [21] underlined in their state-of-the-art and research challenges of environment of MAS, a consensual definition of the environment term in relation to MAS does not exist. Nevertheless, we will propose a positioning and give a general one. Let W be the world in which we are situated. W is both composed of the system to conceive S and the environment of the system ES: $W = S \cup ES$. S is plunged into an environment ES with which it interacts and from which it cannot be dissociated. The *environment* ES is a common "space" for agents composing S. It can be defined as being all what is outside the system; that is, ES is the complementary of the system S in the considered world W, that is $ES = W \setminus S$. ES can be totally or partially perceived by the system S. S and ES are strongly coupled and have an influence on their respective behaviour: in that sense, we can speak about the activity of a system or an environment. More precisely, S locally perceives and acts in ES. Its behaviour modifies the state of ES which reacts by applying a pressure on S. For adapting to these constraints, the system makes a new action in the environment which in turn applies a new pressure which can be seen as a feedback (sometimes implicit), by the system in reference to its previous actions, etc. In fact, the environment in which a system is plunged, applies constraints to which the system must adapt. These interactions coupling leads to reciprocal influences that enable mutual fittings between the system and its environment.

ES is composed of objects that can be situated (their position can be determined) and/or passive or active. In accordance with the previous definition, the environment S of an agent A belonging to the MAS is defined by all that is not A. More precisely, it corresponds to the environment of S as well as the other agents of the system S. An interaction coupling between an agent and its environment exists too (see Fig. 5.1). According to the application, an agent can perceive locally of entirely its environment, but it also has to be able to act in this environment.

5.4.1.1 Types of Environments

One can distinguish:

- The *physical environment* which consists of material resources (active and passive objects such as pheromone, food item, obstacle, ... in the ants application) which are in the perception field or the effectors of an agent. In that case, the

physical environment can be an interaction medium: agents can interact in an indirect way, through the environment. For example, ants communicate by depositing pheromonal marks in the environment. The physical environment has more or less importance according to the application domain. Indeed, it is neglected in software agents (a message is supposed to reach its receiver) but is essential for situated agents such as, for example, robots moving in a discrete environment to achieve their objectives and ant colony foraging the environment to gather food.
- The *social environment* which is composed of known agents. In a MAS, agents constituting the system, simultaneously evolve in a same environment. They maintain more or less strong relations with other agents, depending on whether they have or not the necessary skills to achieve their individual objectives. To efficiently interact, agents need to have representations of other known agents, that is of agents with which they already interacted. These representations can be more or less complex for an agent. They can concern its own skills, skills of known agents, available resources or intentions or commitments of others agents, ... The more these representations are right, the more interactions are relevant and enable agents to rapidly achieve their objectives. For example, if an agent needs particular information, it asks help to agents it knows and which, from its point of view, are likely to have the required information. This point of view depends on representations owned by the agent. Thanks to these representations, agents can interact with other agents to collaborate, cooperate or coordinate.

Odell et al. [15] elaborate this previous variation of environments. According to them, "an environment provides the conditions under which an entity (agent or object) exists". More precisely, they distinguish:

- The physical environment which provides those principles and processes that govern and support a population of entities.
- The communication environment which provides those principles, processes, and structures that enable an infrastructure for agents to convey information.
- The social environment which is a communication environment in which agents interact in a coordinated manner.

These notions of social and physical environments have also been added in the AGR model [1, 9] in order to represent not only the physical part of an interaction but also its social aspect. Agents interact only with the environment which will react according to agent's influences and to the rules of change defined at both physical and social levels of interaction.

5.4.2 Environment Properties

Russell and Norvig [18] had defined and associated different properties with environments. These properties have been adopted by most of researchers in the multi-agent field:

- *Accessible vs. inaccessible*: an environment is accessible to an agent if its sensors detect all aspects that are relevant to the choice of the action.

  For example, in a cheese game where the agents of the MAS are the pawns of a same color, the environment of a pawn agent is accessible because for playing, this agent may observe all the chessboard. In soccer, where the agents of the MAS are players of a same team, the environment of a player is inaccessible because a player has a limited perception of its environment.
- *Deterministic vs. non-deterministic*: the environment is deterministic if the next state of the environment is completely determined by the current state and the actions selected by the agents.

  For example, in the chess MAS presented previously, the environment of a pawn agent is deterministic because when the pawn plays, it is alone to play, and it may forecast the next state of the game. In soccer, the environment of a player is non-deterministic because when the player acts, the other players can act simultaneously.
- *Static vs. dynamic*: the environment is dynamic for an agent if the environment can change while an agent is deliberating.

  For example, in the chess MAS, the environment of a pawn agent is static because when the prawn deliberates, nobody can play. In soccer, the environment of a player is dynamic because when the player deliberates, the other players can act.
- *Discrete vs. continuous*: the environment is discrete if there is a limited number of distinct, clearly defined percepts and actions.

  For example, in the cheese MAS, the environment of a pawn agent is discrete because there is a limited number of percepts and actions in a chess game. The real world is an example of continuous environment.

5.5 Multi-Agent Systems and Self-organisation

On the one hand, MASs are one of the most representative among artificial systems dealing with *complexity* and *distribution* [20, 22] and enable to deal with systems where the global behaviour emerges from the local interactions of the agents. On the other hand, self-organisation is a process which can lead to these emergent phenomena. Self-organisation is defined as a set of dynamical interactions whereby structures appear at the global level of a system from interactions among its lower-level components. The rules specifying the interactions are executed on the basis of purely local information, without reference to the global pattern [2]. So, naturally, self-organisation has represented an inspiration source for MASs designer.

The objective of most researchers in self-organising MASs is to embed self-organisation inside the MASs. This consists in finding relevant mechanisms to guide the agent behaviour at the micro-level, helping the agents to self-organise and to obtain at the macro-level, the behaviour of the system the designer expects. Therefore, designing such MAS requires to find rules to make the system achieve the required collective behaviour, that is "functions that are useful to the system's stakeholders"

[16], "the required macroscopic behaviour" [5], "a functionally adequate function" [3], ...

But the previous definition framework needs to be carefully instantiated with specific techniques enabling this self-organisation while allowing emergent functionalities to appear. The main question is: "How does this produce a complex system with the right behaviour at the global level?" The environment plays here its key role by constraining the system, and the system needs to be able to adapt to these constraints. There is an apparent antinomic situation in the idea of engineering applications with emergent functionalities. On the one hand, emergent behaviour is a behaviour which occurs and in a certain manner cannot be under control. On the other hand, a software designer wants the system he is building to achieve a desired function. Therefore, we can conclude by saying that we want to *control the emergent behaviour* of systems. The solution is then to better understand relations between micro- and macro-levels and to build a system able to self-organise and self-adapt to environmental dynamics.

Currently, some self-organising MASs have been implemented taking inspiration from the following three main domains: the biologic and natural one [13], the social one [11], and the artificial one [6] and most of the time using techniques based on: stigmergy, cooperation, gossip, natural selection, attraction and repulsion, potential fields, social relationships, trust, ... (most of them are detailed in this book). Self-organisation enables to design underspecified MAS because the system can self-adapt to new constraints not forecasted at the specification phase. The bottom-up approach of self-organisation simplifies the MAS design and reduces costs development. The next challenge concerns the validation of these systems. Validation is not yet obvious because self-organising systems lead to new challenges not yet taken into account by classical methods. In large-scale dynamic and adaptive systems such as self-organising systems, the methods, techniques and tools for validation are still in a research phase [19]. In general, formal methods [4] for certification, such as model checking, theorem proving, ..., are adequate for checking/proving desired properties of the system when the code is showing the following properties: it is static, and it runs in well-known environments. A static code is a code which does not evolve, and there is no learning at this level. A well-known environment means that the system does not face unexpected events or unexpected scenarios. Engineered complex systems verification and validation can only be achieved using simulation-based approaches. Nowadays, the most reliable way consists in iteratively improving the designed system using mathematical tools (statistical analysis, behavioural parameters optimisation) or semi-autonomous adaptive programming (*Living Design*).

5.6 Conclusion

In this chapter, we have introduced three main concepts for MASs understanding: agent, MAS and environment. A MAS is composed of a set of interacting agents, and it is plunged into an environment. The agent concept has a lot of meaning, the

elicitation of its properties leads to clarify and specify it more. Autonomy, local perceptions and its ability to perceive and act are the main features of an agent. The properties always verified of the studied MASs in this book are their autonomy, their control decentralisation and their knowledge distribution. A MAS evolves in discrete, dynamic and indeterministic environments and sometimes inaccessible ones. This chapter ends by highlighting how self-organisation and MASs are joined in order to design more complex systems.

5.7 Problems–Exercises

5.1 For each case, justify whether the entity is an agent or not; if necessary, you can define more precisely the system and its environment:

(a) A door in a classroom,
(b) An automatic door in a big store,
(c) A thermometer in an hospital room,
(d) A bee in a simulated beehive,
(e) An engineer expert in motors of cars in a car design firm.

5.2 The air space traffic controller is the entity which centralises data in order to manage a given air space area. This controller is a bottleneck, and to overcome this problem, we wish to agentify all planes to enable them to manage the air space traffic.

(a) Characterise this kind of agent with the properties given in Sect. 5.2.2.
(b) Characterise the MAS with the properties given in Sect. 5.3.2.
(c) Define the environment and characterise it with the properties given in Sect. 5.4.2.

5.3 Define and characterise the environment in the following cases:

(a) For a MAS of manufacturing control system composed of the following agents: the machines, the pieces to be manufactured and the workmen (each workman works on a machine).
(b) For a machine agent of the previous system described in (a).
(c) For a MAS composed of the players of a handball team.
(d) For a player of the handball team.

5.8 Further Reading

Multi-Agent Systems, A Modern Approach to Distributed Artificial Intelligence. A book surveying multi-agent systems. (G. Weiss, MIT Press, 1999.)

Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. This book presents developments in the field of multi-agent systems. (J. Ferber, Addison Wesley Longman, 1999.)
An Introduction to MultiAgent Systems. Introduction to agents and multi-agent concepts. (M. Wooldridge, John Wiley and Sons Limited: Chichester, 2002.)

References

1. Baez-Barranco, J., Stratulat, T., Ferber, J.: A unified model for physical and social environments. In: Environments for Multi-Agent Systems III. LNCS, vol. 4389, pp. 41–50 (2007)
2. Bonabeau, E., Dorigo, M., Théraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, London (1999)
3. Capera, D., Georgé, J.P., Gleizes, M.P., Glize, P.: The AMAS theory for complex problem solving based on self-organizing cooperative agents. In: 1st International TAPOCS Workshop at IEEE 12th WETICE, pp. 383–388. IEEE Press, New York (2003)
4. Clarke, E.M., Wing, J.M.: Formal methods: state of the art and future directions. ACM Comput. Surv. **28**(4), 626–643 (1996)
5. De Wolf, T.: Analysing and engineering self-organising emergent applications. Ph.D. thesis, Department of Computer Science, K.U. Leuven, Leuven, Belgium (2007). http://www.cs.kuleuven.be/tomdw/phd/PhD-TomDeWolf-29-May-2007.pdf
6. Di Marzo Serugendo, G.: Self-organisation in MAS. In: Di Marzo Serugendo, G., Gleizes, M.P., Karageorgos, A. (eds.) Tutorial at the 4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'05), Budapest, Hungary, 15 September 2005
7. Document Title Fipa: FIPA communicative act library specification (2001). http://www.fipa.org/specs/fipa00037/
8. Ferber, J.: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley, Reading (1999)
9. Ferber, J., Gutknecht, O.: Alaadin: a meta-model for the analysis and design of organizations in multi-agent systems. In: ICMAS, pp. 128–135. IEEE Computer Society, Los Alamitos (1998)
10. Ferber, J., Jacopin, E.: The framework of eco problem solving. In: Decentralized Artificial Intelligence, vol. II (1991)
11. Hassas, S., Castelfranchi, C., Di Marzo Serugendo, G., Karageorgos, A.: Self-organising mechanisms from social and business/economics approaches. Informatica **30**(1), 63–71 (2006)
12. Labrou, Y., Finin, T.: KQML as an Agent Communication Language. MIT Press, Cambridge (1994)
13. Mano, J., Bourjot, C., Lopardo, G., Glize, P.: Bio-inspired mechanisms for artificial self-organised systems. Informatica **30**(1), 55–62 (2006)
14. Odell, J., Parunak, H., Bauer, B.: Representing agent interaction protocols in UML. In: OMG Document ad/99-12-01. Intellicorp Inc, pp. 121–140. Springer, Berlin (2000)
15. Odell, J., Parunak, H., Fleisher, M., Brueckner, S.: Modeling agents and their environment. In: AOSE 2002. LNAI, vol. 2585 (2003)
16. Parunak, H.V.D., Brueckner, S.: Engineering Swarming Systems, pp. 341–376. Kluwer Academic, Dordrecht (2004)
17. Rao, A.S., Georgeff, M.P.: BDI agents: from theory to practice. In: ICMAS'95, pp. 312–319 (1995)
18. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall Series. Prentice Hall, New York (1995)
19. Slaby, J., Welch, L., Work, P.: Toward certification of adaptive distributed systems. In: Real-time and Embedded Systems Workshop (2006)

20. Weiß, G.: Multiagent Systems, a Modern Approach to Distributed Artificial Systems. MIT Press, Cambridge (1999)
21. Weyns, D., Parunak, H.V.D., Michel, F., Holvoet, T., Ferber, J.: Environments for multiagent systems: state-of-the-art and research challenges. In: Environments for Multiagent Systems. LNAI, vol. 3477 (2005)
22. Wooldridge, M.: An Introduction to Multi-Agent Systems. Wiley, New York (2002)
23. Wooldridge, M., Ciancarini, P.: Agent-Oriented Software Engineering: the State of the Art. LNAI, vol. 1957 (2002)
24. Wooldridge, M., Jennings, N.: Intelligent agents: theory and practice. Knowl. Eng. Rev. **10**(2), 115–152 (1995)