# Intelligent Agents: Multi-Agent Systems

Alfredo Garro[1], Alberto Falcone

*Department of Informatics, Modeling, Electronics and System Engineering, University of Calabria - via P. Bucci, cubo 41/C, 87036 Rende (Italy)*

Matteo Baldoni, Cristina Baroglio

*Department of Computer Science, University of Turin - Corso Svizzera 185, 10149 Turin (Italy)*

Federico Bergenti

*Department of Mathematical, Physical and Computer Sciences, University of Parma - Parco Area delle Scienze 53/A, 43124 Parma (Italy)*

Stefano Mariani

*Department of Sciences and Methods for Engineering - University of Modena e Reggio Emilia - Via Amendola 2, 42122 Reggio Emilia (Italy)*

Andrea Omicini

*Department of Computer Science and Engineering, Alma Mater Studiorum - University of Bologna - Via Zamboni 33, 40126 Bologna (Italy)*

Giuseppe Vizzari

*Department of Computer Sciences, Systems and Communications, University of Milano-Bicocca - Piazza dell'Ateneo Nuovo 1, 20126 Milan (Italy)*

---

**Abstract**

This chapter provides an overview of the fundamentals of the agent-oriented paradigm and of Multi-Agent Systems (MASs). Specifically, the first section introduces the agent-based paradigm and comments its main features. The second section discusses multi-agent systems and their main features by focusing on the concepts of autonomy, coordination, norms, and emerging behavior that jointly characterize agents and MASs. Finally, in the third and fourth sections, the two main approaches to use the agent-oriented paradigm to develop agent-

[1]Corresponding author

based simulations and agent-based software systems are discussed by analyzing the leading and most renowned reference software platforms.
*Keywords:* Agents, Multi-Agent Systems, Agent-Based Modeling and Simulation, Agent-Oriented Software Engineering, Autonomous Systems, Distributed Systems

## 1. Introduction

The agent is a metaphor that natively originated in AI (Artificial Intelligence), see Newell (1982). Its characteristics changed when the emphasis was put on multi-agent systems, rather than on single agents (Van der Hoek and Wooldridge, 2008).

A key event in the history of multi-agent systems was the recognition that agents can be fruitfully used to model and implement distributed systems (Huhns, 2012). The creation of FIPA (Foundation for Physical Intelligent Agents) originates from the recognition that multi-agent systems are a novel and promising approach for the implementation and deployment of distributed systems (Poslad and Charlton, 2001; Bellifemine et al., 2005); indeed, nowadays, multi-agent systems have two major applications outside of their role traditionally played in the AI domain:

- Engineering of distributed systems, with the introduction of Agent-Oriented Software Engineering (AOSE) (e.g., agent-oriented methodologies and agent-oriented programming languages).
- Agent-based Modeling and Simulation (ABMS), which is what this chapter is mostly about.

Agents are entities that observe their environment and act upon it so as to achieve their own goals (Russell and Norvig, 2003; Wooldridge, 2009). Two fundamental characteristics of agents are *autonomy* and *situatedness*. Autonomy means that agents have a sense-plan-act deliberative cycle, which gives them

control of their internal state and behavior. Whereas, agents are situated because they can sense, perceive, and manipulate the environment in which they operate. The environment can be physical or virtual, and it is understood by agents in terms of (relevant) data. Autonomy implies proactivity, i.e., the ability of an agent to take action towards the achievement of its objectives, without being solicited to do so.

From a programming perspective, agent-oriented programming was introduced by Shoham as "a specialization of *object-oriented programming*" (Shoham, 1993). The difference between agents and objects is clear. Citing Wooldridge (Wooldridge, 2009, Section 2.2): (1) Objects do not have control over their own behavior (this is summarized by the well-known motto "Objects do it for free; agents do it because they want it"), (2) Objects do not exhibit flexibility in their behavior, and (3) In standard object model, there is a single thread of control, while agents are inherently multi-threaded.

The agent-based paradigm also differs from the Actor Model (Hewitt et al., 1973) (and from Active Objects (Boer et al., 2017), largely inspired by the latter). Actors, in fact, do not have neither goals nor purposes, even though their specification includes a process. Agents, instead, use their deliberative cycle (as control flow), possibly together with the key abstractions of belief, desire, and intention, so as to formalize algorithms, for example processes for acting in their environment to pursue their goals. In other words, objects "do it" for free because they are data, agents are processes and "do it" because it is functional to their objectives, see also Baldoni et al. (2016).

The environment, in which agents are situated, does not exhibit the kind of autonomy that is typical of agents although it may evolve, also thanks to an internal process. Its activity, however, is not meant to pursue a goal and this makes environments more similar to active objects.

The binomial agent-environment is formalized by modeling approaches like Demazeau (1995), where the environment is seen as providing "the surrounding conditions for agents to exist...that mediates both the interaction among agents and the access to resources", and in particular by the Agent & Artifact meta-

model Omicini et al. (2008).

A system comprising a number of possibly interacting agents is called a *Multi-Agent System (MAS)*. At this level, it is widely recognized that further abstractions become handy, like organizations and interactions, which are aimed at enabling a meaningful and fruitful coordination of the autonomous and heterogeneous agents in the system Jennings and Campos (1997). Thus, agents are not only situated in a physical environment, they are also situated in a social environment where they get into relationships with other agents and are subject to the regulations of the society they belong to. A normative MAS is "a multiagent system together with normative systems in which agents on the one hand can decide whether to follow the explicitly represented norms, and on the other the normative systems specify how and in which extent the agents can modify the norms" Boella et al. (2007). The impact on the agent's deliberative cycle is that agents can reason about the social consequences of their actions.

## 2. Multi-agent Systems

There is no single definition for the word agent, and there is no single definition for the term multi-agent system (MAS). Notably, major accepted definitions share commonalities, such as the way the agents interact in a system: via the shared environment and/or via structured messages, possibly using ontologies and interaction protocols. Indeed, a MAS can be defined in terms of interacting entities, and in particular, these entities are the agents. Communication may vary from simple forms to sophisticated ones. A simple form of communication is that restricted to simple signals, with fixed interpretations. Such an approach was used by Georgeff in multi-agent planning to avoid conflicts when a plan was synthesized by several agents Georgeff (1988). A more elaborate form of communication is by means of a blackboard. A blackboard is a shared resource, usually divided into several areas, according to different types of knowledge or different levels of abstraction in problem-solving, in which agents may read or write the corresponding relevant information for their actions. Another form of commu-

nication is by message passing among agents, possibly using shared ontologies and interaction protocols to associate a formal meaning with the messages.

The main feature that is achieved when developing MASs is flexibility because a MAS can be extended, modified and reconstructed without the need for detailed rewriting of the application. The MAS also tends to prevent propagation of faults, to be self-recovering, and to be fault tolerant, mainly due to the redundancy of components.

It is crucial to distinguish between Automatic and Autonomous Systems. Automatic systems are fully pre-programmed and act repeatedly and independently of external influence or control. They can be described as self-steering or self-regulating, and they are able to follow an externally given path while compensating for small deviations caused by external disturbances. However, they are not able to define the path according to some given goal or to choose the goal dictating they paths. Whereas, Autonomous systems, such as a MAS, are self-directed toward a goal because they do not require outside control, rather they are governed through laws and strategies that clearly make a difference between traditional and multi-agent systems. If machine learning techniques are utilized, autonomous systems can develop flexible strategies to select their behaviors.

*Norms* are a fundamental ingredient of MASs, guiding the expected system behavior in specific situations and enabling a balance between the objectives of individual agents and those of the MAS. Through norms, the desirable behaviors for a population of a natural or artificial community is represented. Indeed, norms are generally understood as rules indicating actions that are expected to be pursued that are either obligatory, prohibitive, or permissive based on a specific set of facts. According to Hollander and Wu (2011), norms have been used to indicate constraints on behavior (Shoham and Tennenholtz, 1992), to create solutions to a macro level problem (Zhang and Leezer, 2009), and to serve as obligatory (Verhagen, 2000), regulatory, or control devices for decentralized systems (Savarimuthu et al., 2008). The most common norms are:

- Conventions, which are natural norms that emerge without any enforcement (Villatoro, 2011). Conventions solve coordination problems when there is no conflict between the individual and the collective interests; for example, everyone conforms to the desired behavior.

- Essential Norms, which are used to solve or ease collective action problems when there is a conflict between individual and collective interests (Villatoro, 2011; Villatoro et al., 2010). For example, "not to pollute urban streets is essential in that it requires individuals to transport their trash, rather than dispose of it on the spot, an act that benefits everyone".

- Regulative Norms. Regulative norms are intended to regulate activities by imposing obligations or prohibitions in performing an action.

- Constitutive Norms, which are affirmed to produce new goal norms or states of affairs, for example, the rules of a game like chess.

- Procedural Norms that are categorized as objective and subjective. Objective procedural norms represent the rules that express how decisions are really made in a normative system, while subjective procedural norms represent the instrument for individuals working in a system, for instance, back-office procedures.

*Coordination* is another distinguishing factor of a MAS. In fact, an agent exists and performs its activity in a society in which other agents exist. Therefore, coordination among agents is essential for achieving the goals and acting in a coherent manner. Coordination implies considering the actions of the other agents in the system when planning and executing one agent's actions. Coordination allows agents to achieve the coherent behaviour of the entire system. Coordination may imply cooperation, and in this case, the MAS works towards common goals to be achieved, but it may also imply competition, with agents having divergent or even antagonistic goals. In this latter case, coordination is important because the agent must take into account the actions of the others, for example competing for a given resource or offering the same service.

Another characterizing feature of a MAS is its *emergent behavior*. Emergent behavior in agents is commonly defined as behavior that is not attributed to any individual agent, but is a global outcome of agent coordination (Zhengping et al., 2007). This definition emphasizes that emergent behavior is a collective behavior. There are also other definitions. "Emergent behavior is that which cannot be predicted through analysis at any level simpler than that of the system as a whole [...] Emergent behavior, by definition, is what's left after everything else has been explained" (Dyson, 1997). This definition highlights the difficulty in predicting and explaining emergent behavior. If the behavior is predictable and explainable, then it will not be treated as emergent behavior and approaches could be designed to handle these behaviors. Emergence is also defined as the action of simple rules combined to produce complex results (Rollings A., 2003). This definition states that the rules applied to the individuals can be quite simple, but the collective behavior of the group may turn out to be quite complex or unpredictable. This effect has been proved also experimentally. So, emergent behavior is essentially any behavior of a system that is not a property of any of the components of that system, and it emerges due to interactions among the components of the system. Borrowing from biological models, such as an ant colony, emergent behavior can also be thought of as the production of high-level or complex behaviors through the interaction of multiple simple entities. Some examples of emergent behaviors:

- Bee colony behavior, where the collective harvesting of nectar is optimized through the waggle dance of individual worker bees;
- Flocking of birds cannot be described by the behavior of individual birds;
- Market crashes cannot be explained by "summing up" the behavior of individual investors.

Further details about multi-agent systems can be founded in Baldoni et al. (2010).

## 3. Agent-based Modeling and Simulation

Agent-Based Modelling and Simulation (ABMS) refers to a category of computational models invoking the dynamic actions, reactions and intercommunication protocols among the agents in a shared environment in order to evaluate their design and performance and derive insights on their emerging behaviour and properties (Abar et al., 2017). Agents and multi-agent systems are entities that can be effectively used to model complex systems made of interacting entities. This is why they have been adopted to study biological and chemical systems, especially when the systems become too complex for the analytical tools available from chemistry, physics and mathematical physics. The fact that agents and multi-agent systems are abstractions with executable counterparts, the agents and the multi-agent systems that many tools support, contributed to suggest the use of agent technology to simulate, among others, also biological and chemical systems. The size of the simulated systems, and the high level of accuracy of the simulated phenomena, calls for dedicated tools to enable domain experts to describe simulations with little, or no, interest on the engineering issues related to the underlying distributed systems. Agent-based technology provides such tools, and it supports domain experts in the construction of effective distributed systems with minimal emphasis on the inherent issues of large-scale distributed systems. Notably, even if dedicated tools are available, it is common to adopt tools designed to support agent-oriented software engineering in the scope of ABMS. In particular, a number of agent-based tools to support modeling and simulation of complex and/or distributed systems are available (Allan, 2009; Rakić et al., 2020):

1. Netlogo: it is a multi-agent programmable modeling environment which allows the simulation of natural and social phenomena. It is particularly well suited for modeling complex systems evolving over time. Indeed, modelers can give instructions to hundreds or thousands of agents operating independently. This makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level pat-

terns that emerge from their interaction. It comes with a large library of existing simulations, both participatory and traditional, that one can use and modify in different domains such as social science and economics, biology and medicine, physics and chemistry, and mathematics and computer science (Blikstein et al., 2006). In the traditional NetLogo simulations, the simulation runs according to the rules that the author specifies. A further feature of NetLogo is HubNet, a technology that lets domain experts use NetLogo to run participatory simulations. HubNet adds a new dimension to NetLogo by letting simulations run not just according to rules, but by direct human participation.

2. FLAME: it is a generic agent-based modelling system that can be used to develop applications in many areas. Models are created based upon a model of computation called (extended finite) state machines. The framework can automatically generate simulation programs that can run models efficiently on HPCs. It produces a complete agent-based application that can be compiled and built on the majority of computing systems ranging from laptops to HPC super computers. Furthermore, FLAME provides a Model Library, which is a collection of relatively simple models that illustrate the use of FLAME in different applications.

3. AnyLogic: it is a simulation tool that supports all the most common simulation methodologies available today: System Dynamics, Process-centric (AKA Discrete Event), and Agent-Based modeling. Its visual development environment significantly speeds up the development process. It has application models in Manufacturing–Logistics, Supply Chains–Markets Competition, Business Processes Modeling, Healthcare, Pharmaceuticals Simulation, Pedestrian Traffic Flows, Information, Telecommunication Networks Simulation Modeling, Social Process, Marketing Simulation, Asset Management, Financial Operations with Simulation Modeling, Warehouse Operations, and Layout Optimization.

4. Repast: The Repast Suite is a family of advanced, free, and open-source agent-based modeling and simulation platforms that have collectively been

under continuous development for many years. Repast Simphony is a richly interactive and easy to learn Java-based modeling system that is designed for use on workstations and small computing clusters. An advanced version is called Repast for High Performance Computing, which is a lean and expert-focused C++-based modeling system, that is designed for use on large computing clusters and supercomputers.

5. Jason: It is an interpreter for an extended version of AgentSpeak, which has been one of the most influential abstract languages based on the BDI architecture. Jason implements the operational semantics of that language, and it supports strong negation, so both closed-world and open-world assumptions are available. Annotations in beliefs are used for meta-level information and annotations in plan labels. Jason provides the possibility to run a multi-agent system distributed over a network and is the agent programming language in the well-known multi-agent programming platform JaCaMo (Boissier et al., 2020).
6. Framsticks: it is a three-dimensional life simulation project. Both mechanical structures ("bodies") and control systems ("brains") of creatures are modeled. It is possible to design various kinds of experiments, including simple optimization, co-evolution, open-ended and spontaneous evolution, distinct gene pools and populations, and modeling of species and ecosystems. Users of this software work on evolutionary computation, artificial intelligence, neural networks, biology, robotics and simulation, cognitive science, neuroscience, medicine, philosophy, virtual reality, graphics, and art.
7. Gephi: it is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs. It allows (i) Exploratory Data Analysis: intuition-oriented analysis by networks manipulations in real time; (ii) Link Analysis: revealing the underlying structures of associations among objects, in particular in scale-free networks; (iii) Social Network Analysis: easy creation of social data connectors to map community organizations and small-world networks; (iv)

Biological Network analysis: representing patterns of biological data; (v) Poster Creation: scientific work promotion with hi-quality printable maps. It works mainly with metrics related to centrality, degree (power-law), betweenness, closeness, density, path length, diameter, HITS, modularity, and clustering coefficient.

8. Stanford Network Analysis Platform (SNAP): it is a general purpose, high performance system for analysis and manipulation of large networks. It easily scales to massive networks with hundreds of millions of nodes, and billions of edges. It efficiently manipulates large graphs, calculates structural properties, generates regular and random graphs, and supports attributes on nodes and edges.

9. Multi-Agent Simulation Environment (MESA): it is a Python library designed to facilitate the creation, implementation, and analysis of agent-based models during a simulation scenario. Agent-based modeling involves representing individual agents with specific behaviors and interactions to observe the emergent properties of a system. MESA enables users to rapidly generate agent-based models, utilizing pre-built core components, like spatial grids and agent schedulers, or personalized implementations. Researchers can visualize these models through a web-based interface and analyze the results using Python's data analysis tools. MESA aims to serve as the Python 3-based alternatives to modeling environments such as NetLogo, Repast, or AnyLogic (see, e.g., Kazil et al. (2020)).

In addition to enablig tools and platforms, agent-based and multi-agent approaches to the modelling and simulation of biological systems have become quite widespread and have produced several results in the past few years. Kaul and Ventikos (2015) discuss some general properties of these approaches, compared in particular to mathematical modeling and Cellular Automata: the need to accommodate both the possibility to represent and manage heterogeneity, flexibility, as well as continuous features of the modelled system often led to choose agent-based approaches. Even in the recent context of the COVID19

pandemic, agent-based approaches have been successfully applied, also to less micro-scale biological systems and considering instead the overall context of the pandemics in a territory or whole country: Kerr et al. (2021) describes an open source agent-based simulator considering a variety of aspects, from country specific demographics information, to realistic transmission networks in different social layers and contexts, together with means for evaluating the plausible implications of interventions and restrictions on the diffusion of the contagion; Hinch et al. (2021) also describes an agent-based and open source simulator with similar aims, but this simulator is parameterized on UK demographics and calibrated to the UK epidemic.

## 4. Agent-Oriented Software Engineering

Since its early days in the late 1960s, *software engineering* has been constantly facing the problem of better understanding the sources of complexity in software systems (Sommerville, 2011). Over the last four decades, the complexity of interactions among parts has been progressively identified as one of the most significant sources of such a complexity. Software systems that contain a—possibly large—number of interacting parts are critical, especially when the graph of interactions changes dynamically, parts have their own thread of control, and parts are engaged in interactions governed by complex protocols (see, e.g., Wooldridge and Ciancarini (2001) for an in-depth discussion). As a consequence, a major research topic of software engineering has been the development of techniques and tools to understand, model, and implement systems in which interactions are the major source of complexity. This has led to the search for new computational abstractions, models, and tools to reason and to implement MASs, which have been recognised as prototypical examples of such systems. *AOSE* (*Agent-Oriented Software Engineering*) is a paradigm of software engineering that has been developed to target the inherent complexity of analysing and implementing MASs (see, e.g., Bergenti et al. (2004) for a comprehensive reference on the subject). A number of AOSE methodologies have

been proposed over the last two decades, and Kardas (2013) provides a recent survey of the state of the art of such methodologies.

Besides methodologies, the research on AOSE have been constantly interested in delivering effective tools to implement MASs Baldoni et al. (2021). *Agent platforms* are examples of these tools intended to offer generic runtime environments for the effective deployment and execution of MASs. A number of platforms have been proposed over the years, and Kravari and Bassiliades (2015) proposes an attempt to enumerate the platforms that survived the test of time. One of the most popular agent platforms is *JADE* (*Java Agent DEvelopment framework*), as described in Bellifemine et al. (2007), which consists of a middleware and a set of tools that help the development of distributed, large-scale MASs. JADE is widely used for industrial and academic purposes, and it can be considered as a consolidated tool. Just to cite a notable industrial example, JADE has been in daily use for service provision and management in Telecom Italia for more than nine years, serving millions of customers in one of the largest and most penetrating broadband networks in Europe (see Bergenti et al. (2015) for further details). In order to effectively address the inherent issues of the high-profile scenarios that agent platforms target, specific tools are needed to assist the development of complex functionality, and to promote the effective use of the beneficial features of agent technology as a software development technology. Nevertheless, approaching AOSE with the help of agent platforms alone is often perceived as a difficult task for two main reasons. First, the continuous growth of agent platforms has been increasing their inherent complexity, and the number of implementation details that the programmer is demanded to master for the construction of MASs is equally grown. Second, the choice of mainstream programming languages as the unique option to use agent platforms is now considered inappropriate in many situations because such languages do not natively offer the needed abstractions for the effective concretization of AOSE Bergenti et al. (2020).

The interest in *agent programming languages* dates back to the introduction of agent technologies and, since then, it has grown rapidly. Agent program-

ming languages turned out to be especially convenient to model and develop complex MASs. Nowadays, agent programming languages represent an important topic of research, and they are widely recognized as important tools in the development of agent technologies, in contrast with mainstream languages, that are often considered not suitable to effectively implement AOSE Baldoni et al. (2021). Agent programming languages are usually based on specific agent models, and they aim at providing specific constructs to adopt such models at a high level of abstraction. The features of the various agent programming languages proposed over the years may differ significantly, concerning, e.g., the selected agent's mental attitudes (if any), the integration with an agent platform (if any), the underlying programming paradigm, and the underlying implementation language. Some classifications of relevant agent programming languages have already been proposed to compare the characteristics of different languages and to provide a clear overview of the state of the art. Bădică et al. (2011) classifies agent programming languages on the basis of the use of mental attitudes. According to such a classification, agent programming languages can be divided into: *Agent-Oriented Programming* (*AOP*) languages, *Belief Desire Intentions* (*BDI*) languages, hybrid languages—which combine the two previous classes—and other languages—which fall outside of the previous classes. It is worth noting that this classification recognizes that BDI languages follow the *agent-oriented programming* paradigm, as defined in Shoham (1993), but it reserves special attention to them for their notable relevance in the literature. Bordini et al. (2006) proposes a different classification, where languages are divided into declarative, imperative, and hybrid. Declarative languages are the most common because they focus on automated reasoning, both from the AOP and from the BDI points of view Bordini and Hübner (2005). Some relevant imperative languages have also been proposed, and most of them were obtained by adding specific constructs to existing procedural programming languages Bergenti et al. (2018). Finally, the presence (or absence) of a host language is an important basis of comparison among agent programming languages Rodriguez et al. (2014).

Even if the early proposals dates back to the late 1990s, AOSE is still at

an early stage of evolution. While there are many good arguments to support the view that agents represent an important direction for software engineering, there is still need of actual experience to underpin these arguments. Methodologies and tools to support the deployment of MAS are beginning to become accepted for mission-critical applications, but slowly. Although a number of agent-oriented analysis and design methodologies have been proposed, there is comparatively little consensus among them. In most cases, there is not even agreement on the kinds of concepts the methodology should support. But, the research on AOSE is still active, and it has been recently revitalized by the view of AOSE in terms of *model-driven development* (Kardas, 2013; Bergenti et al., 2017).

## 5. Guidelines/Perspectives

Agents and multi-agent systems are intensively exploited to analyze through simulation biological and chemical systems. The literature reports various successful uses of the abstractions and of their executable tools (Politopoulos, 2007). As an example, in (Montagna et al., 2006) an agent-based framework, implemented on top of the TuCSoN agent coordination model and infrastructure, is used to model biological networks as MASs, where agents represent active biological components, such as proteins and enzymes, able to interact in a tuple-based environment (Denti and Omicini, 1999).

Notably, the study of the benefits and the costs of adopting agents and multi-agent systems to support scientific studies of biological and chemical systems has not been approached extensively. A well-known application field regarding the study of biological phenomena where agents and MAS have been successful exploited is the protein synthesis, a common and relevant phenomenon in nature. In this field, several approaches for predicting the three-dimensional structure of proteins are, for instance, available in literature (Jennings et al., 1998). Several of them exploit the chemical or physical properties of the proteins (e.g. Standley et al. (1998); Dudek et al. (1998) work on energy minimization whereas

Galaktionov and Marshall (1995); Sabzekar et al. (2017) use intra-globular contacts); other ones use evolutionary information (Piccolbon and Mauri, 1998). Some tools for the prediction of the three-dimensional structure of proteins have been presented in Douguet and Labesse (2001); Gough et al. (2001); Meller and Elber (2001). Laimer et al. (2015) employs a multi-agent prediction system for stability prediction upon point mutations, which have a strong impact on protein stability. Different agents employ statistical scoring functions (SSFs) and different machine learning approaches to produce predictions, which are then combined by another agents to remove outliers and compute a consensus prediction with an associated confidence estimation. A similar approach based on meta-predictors is presented in Garro et al. (2004). Grimes et al. (2021) describes a flexible, multi-agent approach to predictive classification problems: agents interact and share information socially in an arena with a variable number of participants, once again to reach a common decision. The system is applied retrospectively examining tumor characteristics acquired in the routine care of patients to make a binary classification, node-positive or node-negative, that predicts lymph node metastasis status.

These examples show that the autonomy of agents, their normed freedom of interaction, and the possibility of deploying large-scale systems with minimal care on issues related to distributed systems are few of the major benefits of approaching modeling and simulation of complex systems with agents. Furthermore, they closely represent how natural systems work by distributing a problem among a number of reactive, autonomous, deliberative, pro-active, adaptive, possibly mobile, flexible and collaborative entities. All these properties make agent technology more suited than other ones (e.g., a distributed system, an artificial intelligent system) for facing the challenges and high level of complexity of biological systems and related phenomena.

## References


Abar, S., Theodoropoulos, G.K., Lemarinier, P., O'Hare, G.M. (2017). Agent based modelling and simulation tools: A review of the state-of-art software. Computer Science Review 24, 13–33.

Allan, R.J. (2009). Survey of agent based modelling and simulation tools. Science & Technology Facilities Council New York.

Bădică, C., Budimac, Z., Burkhard, H.D., Ivanovic, M. (2011). Software agents: Languages, tools, platforms. Computer Science and Information Systems 8, 255–298.

Baldoni, M., Baroglio, C., Calvanese, D., Micalizio, R., Montali, M. (2016). Towards Data- and Norm-aware Multiagent Systems, in: Post-Proc. of the 4th International Workshop on Engineering Multi-Agent Systems, EMAS 2016, Revised Selected and Invited Papers, LNAI 10093, Springer. pp. 22–38.

Baldoni, M., Baroglio, C., Mascardi, V., Omicini, A., Torroni, P. (2010). Agents, multi-agent systems and declarative programming: What, when, where, why, who, how?, in: Dovier, A., Pontelli, E. (Eds.), A 25-Year Perspective on Logic Programming: Achievements of the Italian Association for Logic Programming, GULP, Springer. pp. 204–230.

Baldoni, M., Bergenti, F., El Fallah Seghrouchni, A., Winikoff, M. (2021). Special issue on current trends in research on software agents and agent-based software systems. Autonomous Agents and Multi-Agent Systems 35, 29.

Bellifemine, F., Bergenti, F., Caire, G., Poggi, A. (2005). Jade-A Java agent development framework, in: Multi-agent programming: Languages, platforms and applications, Springer. pp. 125–147.

Bellifemine, F., Caire, G., Greenwood, D. (2007). Developing multi-agent systems with JADE. Wiley Series in Agent Technology, John Wiley & Sons.

Bergenti, F., Caire, G., Gotta, D. (2015). Large-scale network and service management with WANTS, in: Industrial Agents: Emerging Applications of Software Agents in Industry, Elsevier. pp. 231–246.

Bergenti, F., Caire, G., Monica, S., Poggi, A. (2020). The first twenty years of agent-based software development with JADE. Autonomous Agents and Multi-Agent Systems 34.

Bergenti, F., Gleizes, M.P., Zambonelli, F. (Eds.) (2004). Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook. Springer.

Bergenti, F., Iotti, E., Monica, S., Poggi, A. (2017). Agent-oriented model-driven development for JADE with the JADEL programming language. Computer Languages, Systems & Structures 50, 142–158.

Bergenti, F., Monica, S., Petrosino, G. (2018). A scripting language for practical agent-oriented programming, in: Proceedings of the 8th ACM SIGPLAN International Workshop on Programming Based on Actors, Agents, and Decentralized Control (AGERE 2018), ACM. pp. 62–71.

Blikstein, P., Rand, W., Wilensky, U. (2006). Participatory, embodied, multi-agent simulation, in: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 1457–1458.

Boella, G., van der Torre, L.W.N., Verhagen, H. (2007). Introduction to normative multiagent systems, in: Boella, G., van der Torre, L.W.N., Verhagen, H. (Eds.), Normative Multi-agent Systems, 18.03. - 23.03.2007, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Boer, F.D., Serbanescu, V., Hähnle, R., Henrio, L., Rochas, J., Din, C.C., Johnsen, E.B., Sirjani, M., Khamespanah, E., Fernandez-Reyes, K., et al. (2017). A survey of active object languages. ACM Computing Surveys (CSUR) 50, 1–39.

Boissier, O., Bordini, R.H., Hübner, J., Ricci, A. (2020). Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo. MIT Press.

Bordini, R.H., Braubach, L., Dastani, M., Seghrouchni, A.E.F., Gomez-Sanz, J.J., Leite, J., O'Hare, G., Pokahr, A., Ricci, A. (2006). A survey of programming languages and platforms for multi-agent systems. Informatica 30.

Bordini, R.H., Hübner, J.F. (2005). Bdi agent programming in agentspeak using jason, in: International workshop on computational logic in multi-agent systems, Springer. pp. 143–164.

Demazeau, Y. (1995). From interactions to collective behaviour in agent-based systems, in: Proceedings of the 1st. European Conference on Cognitive Science, Saint-Malo. pp. 117–132.

Denti, E., Omicini, A. (1999). An architecture for tuple-based coordination of multi-agent systems. Software: Practice and Experience 29, 1103–1121.

Douguet, D., Labesse, G. (2001). Easier threading through web-based comparisons and cross-validations. Bioinformatics 17, 752–753.

Dudek, M., Ramnarayan, K., Ponder, J. (1998). Protein structure prediction using a combination of sequence homology and global energy minimization: Ii. energy functions. Journal of Computational Chemistry 19, 548–573.

Dyson, G.e.B. (1997). Darwin Among the Machines:The Evolution of Global Intelligence.

Galaktionov, S.G., Marshall, G.R. (1995). Properties of intraglobular contacts in proteins: An approach to prediction of tertiary structure, pp. 326–335.

Garro, A., Terracina, G., Ursino, D. (2004). A multi-agent system for supporting the prediction of protein structures. Integrated Computer-Aided Engineering 11, 259–280.

Georgeff, M. (1988). Communication and interaction in multi-agent planning, in: Readings in distributed artificial intelligence. Elsevier, pp. 200–204.

Gough, J., Karplus, K., Hughey, R., Chothia, C. (2001). Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure. Journal of Molecular Biology 313, 903–919.

Grimes, S., Zarella, M.D., Garcia, F.U., Breen, D.E. (2021). An agent-based approach to predicting lymph node metastasis status in breast cancer, in: 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), IEEE. pp. 1315–1319.

Hewitt, C., Bishop, P., Steiger, R. (1973). A universal modular ACTOR formalism for artificial intelligence, in: Nilsson, N.J. (Ed.), Proceedings of the 3rd International Joint Conference on Artificial Intelligence. Standford, CA, August 1973, William Kaufmann. pp. 235–245.

Hinch, R., Probert, W.J., Nurtay, A., Kendall, M., Wymant, C., Hall, M., Lythgoe, K., Bulas Cruz, A., Zhao, L., Stewart, A., et al. (2021). Openabm-covid19–an agent-based model for non-pharmaceutical interventions against covid-19 including contact tracing. PLoS computational biology 17, e1009146.

Van der Hoek, W., Wooldridge, M. (2008). Multi-agent systems. Foundations of Artificial Intelligence 3, 887–928.

Hollander, C.D., Wu, A.S. (2011). The current state of normative agent-based systems. Journal of Artificial Societies and Social Simulation 14, 6.

Huhns, M.N. (2012). Distributed Artificial Intelligence: Volume I. volume 1. Elsevier.

Jennings, N., Sycara, K., Wooldridge, M. (1998). A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems 1, 7–38.

Jennings, N.R., Campos, J.R. (1997). Towards a social level characterisation of socially responsible agents. IEE Proceedings-Software 144, 11–25.

Kardas, G. (2013). Model-driven development of multiagent systems: A survey and evaluation. The Knowledge Engineering Review 28, 479–503.

Kaul, H., Ventikos, Y. (2015). Investigating biocomplexity through the agent-based paradigm. Briefings in Bioinformatics 16, 137–152.

Kazil, J., Masad, D., Crooks, A. (2020). Utilizing python for agent-based modeling: The mesa framework, in: Social, Cultural, and Behavioral Modeling: 13th International Conference, SBP-BRiMS 2020, Washington, DC, USA, October 18–21, 2020, Proceedings 13, Springer. pp. 308–317.

Kerr, C.C., Stuart, R.M., Mistry, D., Abeysuriya, R.G., Rosenfeld, K., Hart, G.R., Núñez, R.C., Cohen, J.A., Selvaraj, P., Hagedorn, B., et al. (2021). Covasim: an agent-based model of covid-19 dynamics and interventions. PLOS Computational Biology 17, 1–32.

Kravari, K., Bassiliades, N. (2015). A survey of agent platforms. Journal of Artificial Societies and Social Simulation 18, 11.

Laimer, J., Hofer, H., Fritz, M., Wegenkittl, S., Lackner, P. (2015). Maestro-multi agent stability prediction upon point mutations. BMC bioinformatics 16, 1–13.

Meller, J., Elber, R. (2001). Linear programming optimization and a double statistical filter for protein threading protocols. Proteins: Structure, Function and Genetics 45, 241–261.

Montagna, S., Ricci, A., Omicini, A., DEIS, A., STUDIORUM Universita di Bologna, M. (2006). Agents & artifacts for systems biology: toward a framework based on tucson, in: Industrial Simulation Conference, pp. 25–32.

Newell, A. (1982). The knowledge level. Artificial intelligence 18, 87–127.

Omicini, A., Ricci, A., Viroli, M. (2008). Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems 17, 432–456. Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

Piccolbon, A., Mauri, G. (1998). Application of evolutionary algorithms to protein folding prediction. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 1363, 123–135.

Politopoulos, I. (2007). Review and analysis of agent-based models in biology. University of Liverpool .

Poslad, S., Charlton, P. (2001). Standardizing agent interoperability: The fipa approach. Multi-Agent Systems and Applications: 9th ECCAI Advanced Course, ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001 Prague, Czech Republic, July 2–13, 2001 Selected Tutorial Papers 9 , 98–117.

Rakić, K., Rosić, M., Boljat, I. (2020). A survey of agent-based modelling and simulation tools for educational purpose. Tehnički vjesnik 27, 1014–1020.

Rodriguez, S., Gaud, N., Galland, S. (2014). Sarl: a general-purpose agent-oriented programming language, in: 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), IEEE. pp. 103–110.

Rollings A., A.E. (2003). Andrew Rollings and Ernest Adams on Game Design.

Russell, S.J., Norvig, P. (2003). Artificial Intelligence: A Modern Approach. Pearson Education. 2 edition.

Sabzekar, M., Naghibzadeh, M., Eghdami, M., Aydin, Z. (2017). Protein $\beta$-sheet prediction using an efficient dynamic programming algorithm. Computational biology and chemistry 70, 142–155.

Savarimuthu, B., Purvis, M., Purvis, M. (2008). Social norm emergence in virtual agent societies, pp. 1485–1488.

Shoham, Y. (1993). Agent-oriented programming. Artificial Intelligence 60, 51–92.

Shoham, Y., Tennenholtz, M. (1992). On the synthesis of useful social laws for artificial agent societies (preliminary report), pp. 276–281.

Sommerville, I. (2011). Software engineering (ed.). America: Pearson Education Inc .

Standley, D.M., Gunn, J.R., Friesner, R.A., McDermott, A.E. (1998). Tertiary structure prediction of mixed $\alpha/\beta$ proteins via energy minimization. Proteins: Structure, Function, and Bioinformatics 33, 240–252.

Verhagen, H. (2000). Norm autonomous agents.

Villatoro, D. (2011). Self-organization in decentralized agent societies through social norms, in: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3, pp. 1373–1374.

Villatoro, D., Sen, S., Sabater-Mir, J. (2010). Of social norms and sanctioning: A game theoretical overview. International Journal of Agent Technologies and Systems (IJATS) 2, 1–15.

Wooldridge, M., Ciancarini, P. (2001). Agent-oriented software engineering: The state of the art, in: Agent-Oriented Software Engineering, Springer-Verlag. pp. 1–28.

Wooldridge, M.J. (2009). Introduction to multiagent systems, 2nd edition. Wiley.

Zhang, Y., Leezer, J. (2009). Emergence of social norms in complex networks, pp. 549–555.

Zhengping, L., Cheng, H., Malcolm, Y. (2007). A survey of emergent behavior and its impacts in agent-based systems, pp. 1295–1300.