Software Agent Platforms Available in 2023

Subjects: Computer Science, Artificial Intelligence

Contributor: Zofia Wrona , Wojciech Buchwald , Maria Ganzha , Marcin Paprzycki , Florin Leon , Noman Noor , Constantin-Valentin Pal

Agent systems provide a bottom-up approach to addressing complex tasks. Agents also provide a means of modeling and simulating phenomena that are difficult to model and understand via standard analytical methods. Although the main applications of agent systems pertain to computer science and are often related to artificial intelligence, there are increasingly numerous uses of agents in areas such as the life sciences, ecological sciences, and social sciences. In response to these interests, numerous platforms have been developed as general or application-focused tools. Over time, the field of agent platforms has evolved, with new platforms being developed and others being abandoned.

agent systems   agent-based systems   multi-agent systems   agent platforms   modeling   simulation   swarm intelligence   artificial intelligence

1. Short History of Agent Systems

Perhaps one of the pivotal moments in the development of agent systems was the Workshop on Distributed Artificial Intelligence held at MIT in June 1980, where 22 researchers presented their results and ideas [1]. After that, the 1980s saw the first attempts at defining the main concepts of the agents' domain. Interestingly, among these, one can also recognize the key problems that are still being studied today.

In 1984, Axelrod showed how cooperation can emerge from the interaction between selfish entities without centralized control [2]. He discussed several strategies for the iterated prisoner's dilemma, a game-theoretical problem that remains a topic of interest in political and social studies, as well as evolutionary biology.

In 1985, the actor model was introduced by Agha and Hewitt [3]. With all the "generalization" involved in this statement, it can be suggested that an actor can be seen as a "simplified version" of an agent. The actor is reactive as it only responds to the received messages, but it can be used to simulate more advanced behaviors, such as perceiving the environment and performing actions, in response to the incoming communication. The research involving actors contributed to the development of agent systems while also resulting in the development of its own tools, libraries, and platforms (e.g., Erlang [4], Akka [5], and Proto.Actor [6]). However, this pathway of research and development is beyond the scope of the work.

Furthermore, in 1986, the first (micro-)simulations were conceived; e.g., in the pursuit domain, where several predators aim at encircling and/or capturing a prey [7]. Nowadays, simulations in different domains are one of the highly popular application areas for multi-agent systems.

Based on psychological studies of practical reasoning ([8]), in 1987, Georgeff and Lansky developed the so-called procedural reasoning system (PRS), established on the foundation of the belief–desire–intention (BDI) model for intelligent agents [9]. A formalization of the BDI architecture, which remains very popular in the agent domain, can be found in [10]. It explicitly includes the agent's beliefs about the state of the environment and its own state. It also incorporates the concepts of establishing the means for reaching a specific goal and for creating a plan.

The first agents (agent systems) were implemented using the means available at the time; e.g., in Lisp [11] or Prolog [12]. However, as the complexity of applications increased and more experience was gathered, the need for specialized frameworks dedicated to constructing agents was recognized. One of the first attempts at delivering a complete agent platform was AGENT0 [13], a language that incorporated the idea of agent-oriented programming [14] where the agent has complete control over its own state (beliefs, capabilities) and behaviors (responding to messages, commitments).

The 1990s witnessed a steady stream of development in the agent systems domain where separate areas of interest can be identified. During that time, significant advances materialized in the theoretical study of agent systems. For instance, the ARCHON project [15] proposed a general-purpose architecture that could facilitate cooperative problem-solving in industrial applications.

The knowledge query and manipulation language (KQML) was formulated within the DARPA knowledge sharing effort [16] and proposed as a standardized means of agent communication. It separated the intent of the message, in the form of a so-called "performative" (inspired by speech act theory [17]), from its actual content. That effort was further refined within the FIPA agent communication language (ACL) [18] definition. Separately, a specific agent-based language, AgentSpeak, inspired by the PRS was formally introduced in [19]. Moreover, software agents were described and discussed in core contributions by Wooldridge and Jennings [20][21][22].

Related interests emerged in the field of autonomous cars, which have been conceptualized as autonomous agents. Here, several agent architectures were proposed (e.g., Touring Machine [23], InteRRaP [24]) and the first self-driving vehicles were put to the test (e.g., ALVINN [25] and, later, the Nomad rover [26] and Stanley [27]).

Another area of multi-agent systems' application that expanded greatly during that time was related to social simulations. Here, a few notable examples are the Sugarscape emergent economic model [28], the evolution of social corruption [29], the study of the population dynamics of Cyber-Anasazi [30], and a model for civil violence leading to "artificial genocide" [31].

Reflecting on the connection to the industry, one should note the (multi-stage) European AgentLink project [32][33][34], which studied the application of agent systems in domains such as telecommunications, information management, electronic commerce, and manufacturing. It is easy to notice that the vision of software agents formulated within these activities is materializing now, among other places, in the world of the Internet of Things (IoT) [35].

An interesting European initiative from 2013 was the COST agreement technologies action [36]. It aimed at coordinating efforts related to the new paradigm for next-generation distributed systems based on the concept of agreement between computational agents. In such systems, autonomous software agents negotiate with each other, typically on humans' behalf, to reach mutually acceptable agreements.

Despite constant research activities, at the beginning of the new millennium, interest in agent systems gradually declined. However, it appears that, in the past few years, the field has been experiencing a revival of interest accommodated by increased awareness of the importance of agent systems. This interest especially concerns domains such as autonomous cars and drones (e.g., pick-up and delivery routing problem [37]), simulations (e.g., evacuation behavior during emergencies [38]), smart cities [39][40], and the IoT. In these areas, new technological advances in the communication infrastructure (e.g., 5G [41][42]) are facilitating further automation of and interconnections between intelligent devices.

Today, agent systems are extensively used in medical and social simulations, particularly those that examine the spread of infectious diseases (e.g., the impact of mobility restrictions on the spread of COVID-19 [43] and strategies for minimizing the spread of epidemics in populations [44]). Moreover, agent systems are being employed for modeling in the electromobility sector [45][46][47][48], as well as in electricity markets [49].

Nevertheless, as far as industrial-grade systems are concerned, agent-based solutions are frequently being replaced with alternative approaches, such as microservices, serverless systems (FaaS), and workflow management systems (WMSs). This shift is driven by factors such as the broader range of technological options offered by agent system alternatives. For instance, microservices enable the implementation of various application services using different technologies tailored to their specific requirements. As such, they provide greater flexibility and adaptability within the overall system architecture. Furthermore, in many cases, the complexity of agent systems surpasses the actual needs and requirements of the tasks that are to be handled by the system, thereby increasing the cost of the system's development, maintenance (e.g., cross-compatibility), and deployment.

Regardless of the relative lack of popularity of agent systems in commercial applications, over time, the idea of their applicability has materialized in a broad range of different domains.

2. General-Purpose Platforms

2.1. Open-Source General-Purpose Platforms

ActressMas—An agent framework inspired by the actor model and implemented using .NET asynchronous operations. One of its primary design objectives is conceptual simplicity and support for high-level agent abstractions, making it valuable for teaching purposes. In addition to actor-specific reactive behavior, it incorporates mechanisms for proactive behavior and, as such, can be used for the implementation of multi-agent systems. It supports the concept of agent mobility by means of agents residing in distributable containers. Additionally, it adheres to some of the FIPA specification standards for agent platforms, but it is not fully FIPA-compliant;

Akka—A set of libraries dedicated to highly concurrent and distributed systems. In accordance with its design principles, the toolkit is scalable, fault-tolerant, resilient, and highly performative. Similarly to ActressMas, Akka is based on the actor model, where actors encapsulate states and behaviors. Here, the communication between entities is achieved in a message-driven fashion. The model is based on a hierarchical structure and the concept of actors' supervision (i.e., "parent" actors are the supervisors of task distribution and failure handlers).

Akka.NET—Port of the original Akka to C# and F# for systems developed in .NET and Mono. Similarly to Akka, the toolkit is designed for highly concurrent and event-driven applications. The platform is based on high-level actor abstractions and asynchronous and distributed design. The architectural principles of the actor model are the same as in the original Akka platform. An extension provided for Akka.NET that is worth mentioning is the Phobos 2.0 monitoring tool. In particular, the tool enables tracking the system cluster and capturing different actor metrics (e.g., the actor's mailbox latency). In contrast to Akka, Akka.NET is fully open source and is released under the Apache 2.0 license;

ASTRA—A Java-based agent-oriented programming (AOP) language for concurrent and distributed systems. The design of the language is based on simplicity and aims to resemble commonly known OOP languages (e.g., Java, C#). The platform is thus integrated with Java programming principles and, therefore, among other characteristics, it is strongly typed. On the side of AOP, ASTRA is inspired by AgentSpeak(L). As such, it incorporates all of its agent-related functionalities and, specifically, the concepts of the BDI paradigm (e.g., beliefs, events, and plan rules).

BDI4Jade—A platform extending the agent-related functionalities of JADE (e.g., distributivity, agent communication) with BDI paradigm concepts. It provides means for the development of reasoning agents by including definitions of modularized reasoning strategies, goals, beliefs, and plans. In addition to that, the platform adopts the concept of capability relationships [50].

JaCaMo—A platform for multi-agent programming consisting of three technologies: Jason, CArtAgO, and Moise. It is based on the JaCa programming model, which separates the programming for the logic of the agents from the programming for the environment. This separation of concerns is realized through a combination of different agent-based technologies, where (1) Jason facilitates the programming for the cognitive agents using the BDI approach, (2) CArtAgO enables the modeling of multi-agent virtual environments with artifacts, and (3) Moise allows for the definition of specifications for the agents' organizations.

Java Agent Development Framework (**JADE**)—A Java-based industrial-grade framework dedicated to the implementation of multi-agent systems. Its primary objective is standard compliance and, therefore, it is fully aligned with the FIPA specification. The communication between agents is undertaken in a peer-to-peer fashion through asynchronous message passing. JADE supports distributing agents on multiple containers running on different hosts. The agents are fully mobile, meaning that they can be moved from one container to another at system runtime. The platform offers a dedicated GUI and several debugging tools for monitoring the system. Besides the standalone functionalities, JADE is supported by several extensions, such as JADE Test Suite and Leap.

JADEX—A platform extending JADE with the principles of rational agents. The design of the platform follows the concepts of active components (ACs) and service component architecture (SCA). The JADEX components are able to both (1) act as passive service providers and (2) autonomously execute behaviors. Based on these features, they can be seen as the representation of the agents. The platform facilitates the cognitive BDI approach and the business process modeling notation workflow. It also supports mechanisms of dynamic agent discovery and mobility and allows the distribution of agents across multiple machines.

Janus—An agent platform implemented in SARL used in the creation of Web and desktop-based multi-agent applications. It can be used in both organizational-based and holarchy-based agent systems. It incorporates the concepts of BDI architecture, allowing for the development of reasoning agents. In terms of agent interactions, it supports both synchronous and asynchronous communication protocols. The SARL language used in the implementation of Janus adheres to ACL accordingly and, in large part, to the FIPA specification.

JS-son—A versatile and extendable JavaScript library for the development of reasoning agents. The library supports the BDI architecture and adheres to reasoning loops within agents. From the conceptual perspective, JS-son agents can be programmed according to three different approaches: (1) a traditional BDI approach, (2) a simplified belief–plan approach, and (3) a goal-based approach. The first approach follows a traditional BDI architecture with concepts of the agent’s beliefs, desires, and intentions. The belief–plan approach is a simplification of BDI where the agents use their perception of the environment and their internal state to execute plans through which they interact and update their own beliefs. Finally, the goal-based approach focuses strictly on the agent’s goals, which are treated as the driving forces behind their behaviors.

Smart Python Multi-Agent Development Environment (**SPADE**)—A multi-agent platform based on the XMPP/Jabber technology that offers features that assist in the construction of MASs. It is the first agent-based system that uses XMPP as its foundation. The communication between agents is achieved via the concept of communication channels, where the messages are dispatched using dedicated message templates. The platform facilitates the creation of agents (represented as users characterized with unique Jabber identifiers) and agent platforms (XMPP-based servers). It provides an extensible XML-based communication protocol that supports FIPA-ACL metadata.

Tuple Centres Spread over the Network (**TuCSoN**)—Java-based platform used for the development and coordination of multi-agent systems. Specifically, it is dedicated to model agents written in Java and tuProlog. Its main concept is based on tuple space coordination, where tuples (programmable in ReSpecT language) are data structures shared between and accessed by agents in order to facilitate their communication and coordination. The agents are able to interact in the tuple space through the use of primitives, which are inspired by LINDA's coordination model.

XKlaim—a renewed and enhanced version of KLAIM, a coordination language for modeling and programming distributed systems. In particular, the extension considers the incorporation of high-level programming (Java-based) constructs. Similarly to TuCSoN (and the original KLAIM), the platform is based on the LINDA communication concepts and defines the simulations in tuple space. The nodes (agents) are organized in hierarchical nets.

2.2. Open-Source General Simulations and Modeling Platforms

Agents Assembly—A domain-specific language based on assembly mnemonics for implementing multi-agent systems. It is part of a toolset intended for large-scale simulations in containerized environments that aims to streamline the development of agent systems for users without specialized programming skills. The language supports programming constructs that are directly related to agent systems (i.e., agents, behaviors, messages) and offers various mathematical statements and concepts commonly used in other programming languages. For the visualization of created systems, the AASM provides a dedicated graph modifier that also supports the creation of graphs based on statistical distributions. The language is translated in two steps: first into a target-agnostic intermediate representation and then into Python, based on the SPADE platform. AASM offers a dedicated GUI for simulation definition and management, a run environment, and an XMPP-based communication server.

AgentPy—A platform that facilitates agent-based modeling. It is designed strictly for scientific use; therefore, it provides tools for parameter sampling, Monte Carlo experiments, stochastic processes, and sensitivity analysis, among others. The implemented models are built using agents that are combined in groups called sequences. Such an approach allows the manipulation of multiple entities at once. Furthermore, they are placed in environments according to specified positions. Currently, three types of environment topologies are supported: (1) grids, (2) spaces, and (3) networks.

Agents.jl—A framework for Julia that supports the creation of agent-based models (ABMs). It is based on a modular and function-based design and, as such, is easily extendable. The platform supports the implementation of different types of agents, which can also be generated using Agents.jl macros. The simulations are conducted in structured spaces of four types: grids, graphs, continuous spaces, and OpenStreetMaps. The framework provides tools for collecting the simulations' aggregated data and presenting them visually.

AgentScript—An agent-based modeling framework that incorporates the semantics of NetLogo and relies on the model/view/control (MVC) architecture. As such, its design is based on the separation of concerns. Specifically, the system is divided into three parts: the first (model) defines the semantics of agents (turtles), connections (links), and patches; the second (view) represents the model graphically using a 2D/3D canvas (with support for GISs or geo-modules); and the third (control) provides functionalities allowing interaction with the system via a dedicated graphical user interface. By relying on the Javascript/Coffeescript programming language, the platform aims to be highly deployable.

DEVS-Suite—A simulator that is based on the parallel DEVS component and the cellular automata approach. It automates the experimentation process and creates time data trajectories in real time. Furthermore, it features hierarchical model libraries and can be used to animate implemented models. As such, the platform provides the interfaces for both separate and stacked time trajectories, which can be used in component tracking. In terms of cellular automata, DEVS-Suite offers independent tracking of any number of cells with distinct start times. The simulator contains “Models” divided into “CellularAutomata” (e.g., game of life, system biology chemotaxis, forest fire) and “Component” (e.g., single-input single-output, basic, and multiprocessor architectures) packages. However, these packages are not limited, and users can also add their own model packages alongside the provided ones;

EcoLab—An agent-based simulation framework that facilitates experiment-oriented metaphors. Initially, it was designed for an abstract ecology model, but as the environment evolved and more types of models were developed using it, it expanded to become a general-purpose platform. As EcoLab is implemented in C++, it incorporates several advanced algorithms and data structures. The models implemented with the platform can be modified and controlled dynamically at runtime. Additionally, EcoLab supports creating models of distributed agent topologies that can run on the MPI-based cluster. It also facilitates the checkpoint mechanisms, which make it possible to store the model’s state and periodically report it to the visualization client. As such, the models can be accessed via a dedicated GUI with graph and histogramming features;

fjåge—A lightweight platform for the development of multi-agent systems. It aims to be fast and easy to understand. The implemented agents are associated with unique identifiers, which facilitate the exchange of messages. Furthermore, it implements various types of agent actions and behaviors. The primary features of the platform were inspired by the JADE framework. Similarly to JADE, fjåge supports the distribution of agents across multiple containers. However, unlike JADE, the concept of container-running platforms is extended by introducing two types of container management: real-time and event-driven.

FLAME GPU 2—The second version of the GPU extension to the FLAME framework. It incorporates agent-related concepts into C and CUDA code. The agents implemented in FLAME GPU 2 are associated with specific types (corresponding to given models) and encapsulate the state in the form of sets of behaviors. The communication is undertaken in a message-driven manner, and each message, similarly to agents, is also identified by the corresponding type, which defines a given communication strategy.

GAMA—A simulation platform that facilitates the development of spatially explicit multi-agent simulations, including large-scale simulations. The model is based on a general approach and can therefore be used in a variety of applications. Additionally, in order to meet more specific needs, the platform incorporates several plugins (e.g., Remote.GUI for use in participative simulations). The GAMA platform uses a dedicated language called GAML, the design objective of which is to make it intuitive and easy to use by non-computer scientists. Agents in GAMA can be created from any dataset, including geographic information system (GIS) data.

Insight Maker—A modeling framework that employs multimethod models. The platform is capable of mapping conceptual models onto system descriptions through loop diagrams or pictures prepared in a dedicated diagramming tool. The prepared models can then be extended by adding dynamics using one of two modeling paradigms provided by the platform. The first one is called System Dynamics and focuses on the high-level behavior of the system, where the population is treated as a whole.

JAS-Mine—A platform that facilitates discrete-event simulations and the development of large-scale data-driven models. It is designed to follow the separation principle, where the data management is decoupled from its representation and is based on a three-layer architecture consisting of the model (i.e., implementation of simulation components), collector (i.e., processing of simulation statistics), and observer (i.e., simulation visualization). The simulation is handled and organized by the “scheduler” component, which is common to all agents and responsible for ordering the simulation events. Among its features, JAS-Mine also provides integrated I/O communication services (e.g., RDBMS and automatic CSV table creation), and it includes an advanced multi-run utility that supports experiment design.

JSimpleSim—A Java-based simulation and modeling framework that employs a discrete-event approach. It aims to facilitate the building of agent-based models while also supporting fast simulations. The models are interpreted as grids that embed the agents, with behaviors defined in an event-driven fashion (i.e., agents contain queues listing all of their events). The communication between agents employs two concepts: direct messaging and routing, where the messages are sent via dedicated ports. From the architectural perspective, the platform is characterized by the separation of simulation from modeling.

Multi-Agent Simulator of Neighborhoods/Networks (**MASON**)—A Java-based, discrete-event, multi-agent simulation platform for lightweight, large-scale simulations. It offers model libraries and visualization tools for both 2D and 3D applications. It is dedicated to multi-agent simulations with numerous agents that run on a single machine.

MASS—A library designed for parallelizing multi-agent and spatial simulations, enabling the modeling of emergent collective behavior. It addresses the parallelization challenges in prevalent shared-memory approaches by offering a parallel-computing capability for multi-agent and spatial simulations across a cluster of computing nodes. The simulation entities are represented by the distributed arrays of agents. The communication between these agents is undertaken through periodic data exchange. The agents can communicate with both the agents residing in the same place as well as the agents from other places.

Mesa—A modular framework used for the development of agent-based models. It is composed of modules that are organized into three categories: modeling, analysis, and visualization. The modeling modules define structures used in the development of agents. In particular, they include classes of agents, a definition of agents’ movement space (which can be seen as a grid), and a scheduler used to organize actions (which can also be added at runtime). Analysis modules provide tools for data collection and the capability of running the model with various parameter settings.

Multiphysics Object-Oriented Simulation Environment (**MOOSE**)—A finite-element, multiphysics framework that provides a high-level interface for nonlinear solver technology. Its design aims for it to be aligned with real-life problems and provide a modular and extendable architecture. As such, it offers a straightforward API. Among its other features, the platform provides fully coupled and fully implicit multiphysics solvers.

NetLogo—A modeling platform that is suitable for the representation of complex dynamic systems. In particular, it can facilitate the simulation of natural and social phenomena. Through the utilization of thousands of autonomous "agents", the platform allows exploring linkages between the micro-level behavior of individuals and the macro-level patterns of their collective interactions. In particular, NetLogo agents are grouped into "patches" (stationary agents), "turtles" (mobile agents), and "links" (connections between agents). The platform provides utilities that allow monitoring and modifying agents at runtime. It also offers a set of visualization tools, including plots (e.g., line plots, bar plots, etc.) and model visualization in 2D or 3D.

Repast Suite—A collection of agent-based platforms that consists of three open-source frameworks: **Repast Symphony**, **Repast HPC**, and **Repast4Py**:

(1) **Repast Symphony** is a Java-based toolkit that offers features for agent-based modeling. It allows development using ReLogo (a dialect of Logo), Groovy, or Java, as well as including a pure Java point-and-click execution environment with built-in logging and graphing utilities. The platform offers a hierarchically nested definition of space. It allows for the visualization of 2D and 3D environments, networks (with the JUNG network modeling library), and geographical spaces (GIS support).

(2) **Repast High-Performance Computing (HPC)** is the C++ version of Repast Symphony optimized and rewritten from its previously created Java-based version. It incorporates the fundamental concepts of its predecessor, such as contexts and projections, while optimizing them to function in a parallel-distribution manner. It was specifically designed for utilization with large computing clusters and supercomputers;

(3) **Rephast4Py** is the latest addition to the Repast Suite and is built on the foundation of Repast HPC. It offers features and functionalities facilitated by the Python programming language. In contrast to its high-performance counterpart, Repast4Py prioritizes accessibility and ease of use. Its distributed agent-based modeling capability enables the development of complex system models that capture the scale and relevant details of many social problems;

SIMILAR—Agent-based modeling (ABM) meta-model based on the influence reaction model (IRM4MLS). It is based on two-step action processing, where firstly agents generate "influences" (i.e., decisions made based on the agent's internal state) and then the system "reacts" (i.e., evaluating effects according to the environment's state). It is dedicated to complex systems that use knowledge from multiple viewpoints. The platform includes a generic and modular formal model and a simulation API preserving the formal model structure. To enhance efficiency, the API suite is separated between two kernels (a micro-kernel and extended kernel), which the developers may fine-tune according to their needs.

SpaDES—An R meta-package that facilitates the implementation of event-based models, particularly spatial models (i.e., raster-based, event-based, and agent-based models). Its main objective is modularity, through which the users can extend the platform with additional functionalities. The SpaDES core is based on discrete event simulation (DES). The platform is composed of modules represented by structured R scripts that utilize event scheduling and offer distinct semi-autonomous algorithms. The dependencies of modules, individual parameters, and input and output data are defined in module metadata.

2.3. Commercial Platforms

AnyLogic—A modeling tool designed to support business simulations across a wide range of industries by offering dedicated libraries for domain-specific purposes. The software facilitates the multimethod modeling strategy and, as such, incorporates several modeling approaches, including agent-based, discrete-event, and system dynamics. All of these methods can be used separately or in combination with each other. With respect to the ABM, AnyLogic supports the development of various types of agents that can be encapsulated within one another or can form populations. The platform offers three types of spaces in which the agent can reside: continuous 3D space, grid-based discrete space, and geographic information system (GIS) space.

ExtendSim—A suite of simulation tools that enable continuous process modeling and discrete-event simulation. It offers different modeling approaches, including Monte Carlo, state/action and agent-based modeling. In the simulator, the agents are represented as ExtendSim blocks. They are perceived as individual decision-making entities, and their interaction is block-to-block-based.

FlexSim—A tool that enables the modeling, simulation, prediction, and visualization of various business systems across multiple industries. The modeling is undertaken in 3D virtual environments. The agents are referred to as model objects. They are used to generate events that are executed chronologically in a synchronized, discrete-event manner. The platform also facilitates agent (object) grouping with the support of different types of groups. Furthermore, FlexSim provides functionalities that can be used in agent learning; hence, it can serve as a tool to evaluate and implement the reinforcement learning algorithms.

FlexSim HC—A specialized simulation software package designed for the healthcare industry. The tool is provided by the same company that developed FlexSim. It offers a variety of 3D models specifically tailored for modern medical facilities, providing a virtual environment for healthcare professionals to visualize and optimize their operations. From the conceptual perspective, it follows similar principles as FlexSim; however, the concept of 3D objects has some differences.

GoldSim—A platform that enables users to visually and dynamically model and simulate complex systems in a range of fields, including engineering, science, and business. It is based on a “visual spreadsheet” interface, which facilitates graphical creation, manipulation, and analysis of the data, equations, and models. For instance, it provides tools for probabilistic analysis, such as Monte Carlo simulations. The GoldSim simulations are built from modules that follow the top-down hierarchical approach.

Java Agent Construction Kit (**JACK**)—A Java-based development platform for creating MASs with the BDI approach. JACK agents can be described by means of their desires and can implement reasoning behaviors in response to both proactive (when the agent’s knowledge changes) and reactive (in response to external events) triggers. JACK also provides generic templates that can help new developers in the creation of simple agent plans. As defining pre-compiled plans ensures system performance and predictability, JACK is suitable for time-sensitive and mission-critical applications.

(1) **C-BDI** is another framework distributed by the AOS company that developed JACK that aims to facilitate building human–machine teams based on the BDI approach. It aims at facilitating applications with distributed team reasoning, resilience, and scalability. The framework utilizes an explainable AI in its models, making the applications more efficient.

(2) **CoJACK** is the third product distributed by the same company and is intended specifically for the modeling of human behaviors. It applies the principles of cognition theory inside the virtual actors, making it suitable for modeling individual and team behaviors.

Simio—A platform that offers an object-based modeling environment that enables the construction of 3D models directly from a top-down 2D view. The platform facilitates step-based model creation in which models are created incrementally. Similarly to previously described platforms (AnyLogic and GoldSim), Simio also combines discrete-event, agent-based, and continuous simulation approaches.

Simudyne—SDK with an agent-based approach for building models that mimic the real world’s complexity. Simudyne’s agents can interact with each other and react to environmental changes. The relationships between them are represented by networks composed of links, which can be constructed using specific built-in graph-generation algorithms. The platform also allows the modification of the network’s links during runtime, which supports their dynamic evolution.

Simul8—A tool that offers support for simulations across various domains, including those involving continuous processes, discrete events, or agent-based models. Among its features, it offers a drag-and-drop interface that simplifies the creation of simulations, provides a scenario manager that allows testing diverse simulation ideas, and incorporates tools enabling exporting of the simulation’s results to external files.

3. Special-Purpose Platforms

In many cases, domain-specific platforms address the need for agent-based simulation and modeling of various complex phenomena. Several domains, including healthcare, transportation, economics, and ecology, are heavily influenced by human behavior. Hence, traditional modeling techniques often fall short of capturing the complexities of such systems. Interesting examples include simulations of the evacuation of crowded spaces [51] and of traffic congestion [52]. In these cases, the behaviors of individuals are difficult to simulate as they depend on complex reasoning processes and can be subject to unpredictable changes. Agent-based simulations provide the capability to model individuals and simulate their interactions. As individuals are represented by agents, they exhibit autonomous behaviors, which allow a more realistic and detailed understanding of the system dynamics. Due to this, there is a need for dedicated frameworks that can facilitate the implementation of such agent-based models. The domain-specific platforms provide particular features that can be used specifically in the given area of implementation.

3.1. Cognitive, Social, and Affective Agent Platforms

ACT-R—A system that resembles a programming language but its underlying constructs reflect assumptions about human cognition taken from empirical findings of psychological experiments. The framework architecture is based on the modules that allow simulating cognitive processes (e.g., decision making, perception, and attention memory) and, as such, facilitate cognitive reasoning in agents. The agents rely on the knowledge base, which includes “facts”, “concepts”, and “rules” that dictate their behaviors.

Cormas—Agent-based modeling platform designed to emulate communal resource management and the influence that a population has on its surrounding natural environment. The software is aimed at modeling the interactions among the different participants involved in managing sustainable natural resources. The platform enables the creation, examination, and analysis of simulated scenarios.

DALI—A meta-interpreter built on top of Sicstus Prolog. It is a logic programming language that allows the usage of computational logic in the context of agent systems. The primary objective of the specification of this language is the identification and formalization of the basic patterns for proactivity, reactivity, autonomous “thinking”, and “memory”.

GOAL—A programming language designed for creating agents with cognitive capabilities. Specifically, it facilitates the programming at the “knowledge level”. The information possessed by the agent is represented using KR semantics (e.g., OWL/prolog), and these semantics are also used in agent communication. GOAL facilitates dynamic changes in the beliefs and goals of an agent and allows for the structuring of their decision-making processes.

GROWLab—A Java-based software toolbox for modeling social phenomena in the field of geographic conflict research. Specifically, it is intended for applications that require the integration of actual GIS data. Its design considers the implementation of hierarchical agent structures. In particular, it provides notions of “layers” (i.e., groups of agents), “topologies” (i.e., agent interconnections), and “configurations” (i.e., agent hierarchies). Furthermore, the platform provides different types of spaces (e.g., grids, hexagonal spaces, geographic spaces).

JASON—An interpreter for an extended version of the AgentSpeak programming language. In particular, it is characterized by speech act-based inter-agent communication. It is used for creating MASs based on the BDI architecture (e.g., ref [53] used JASON in DEVS simulation).

Neural MMO—An open-source research platform that allows for the simulation of populations of agents within virtual worlds generated using procedural methods. As such, the tool can facilitate foraging tasks, including survival, exploration, and combat, involving a large number of agents generated over several hours. The agents reside in auto-generated environments (initially at random locations) and perform the action in each simulation tick according to the observed game state.

PAXelerate—A software that creates simulations of movements of passengers in airplane cabins. It aims to improve the layout of the boarding-process cabins. The system consists of two main parts: (1) a cabin editor that allows for changes to the cabin layout and (2) a simulator that runs the boarding simulation. The passengers are represented by computer-generated agents with randomly generated physical characteristics who find their seats using an A-Star path-finding algorithm.

Sim2APL—A library facilitating social simulations with intelligent agents. It is based on the 2APL architecture and enables agents to coordinate their actions using time intervals called “ticks”. Each tick represents a complete thinking process for the agents, involving sensing, reasoning, and acting. Unlike the standard 2APL, Sim-2APL relies on agents generating action references instead of directly affecting the environment.

Soar—A cognitive architecture designed to create intelligent agents. It aims to generate agents with the ability to mimic the complete range of cognitive capabilities of a human, such as learning, decision making, and adapting. Agents themselves are composed of fixed blocks provided by the architecture. They make the decisions based on (1) sensed data, (2) the contents of the working memory, and (3) long-term knowledge.

Self-Organizing Social and Inductive Evolutionary Learning (**SOSIEL**)—A multi-agent algorithm simulating social phenomena that considers large areas and long time spans. Each agent has its personalized knowledge, and sharing it is bounded by spatial criteria. Each agent can learn from the experiences of others and can adapt their practices based on that knowledge.

3.2. Platforms for Learning Agents

Brax—A physics engine that aims to facilitate developments in the areas of robotics, human perception, materials science, and reinforcement learning, among others [54]. It supports both single-device simulations and distributed, parallelized ones. The engine was implemented using JAX and, as such, it can facilitate the environment’s simulation of millions of “physics steps” per second and support fast agent training.

Deepmind Garage—A toolkit for the development and assessment of reinforcement learning algorithms. It provides features for defining custom environments in which agents (robots) are represented by 2D observation points executing 2D actions (defined as velocities). The state of the agents is updated in a step-based manner. Furthermore, the platform provides a set of libraries that contain constructs that can be used in the implementation of agent training algorithms.

Deepmind lab—A platform that offers a set of tasks requiring agents to implement 3D navigation and puzzle solving. These tasks are created by adapting levels from Quake III Arena, a first-person shooter game, as well as custom levels designed by the community using the platform’s built-in level editor. Its primary aim is to serve as a testing ground for artificial intelligence research, particularly in the field of deep reinforcement learning.

Gazebo—A 3D dynamic software package that allows for the simulation of populations of robots in both indoor and outdoor environments. It is based on a robust physics engine (DART) with support for kinematic and dynamic applications.

Gymnasium—Open-source Python library for creating and analyzing reinforcement learning algorithms. It achieves this goal by providing a standardized API that enables communication between learning algorithms and environments. The actions are executed on the environment in a step-based manner, where “steps” refer to the agent observations (i.e., rewards resulting from actions).

Habitat—Simulation toolkit that facilitates the development and training of embodied AI agents. It allows visualizing agents in photorealistic 3D environments. The toolkit consists of two main components: Habitat-Sim and Habitat-Lab. Habitat-Sim is a 3D high-performance simulator with, among other assets, enabled physics, rigid-body structures, CAD models of spaces, and configurable sensors.

MAgent2—A research tool designed for multi-agent reinforcement learning (MARL) with the goal of scaling up to a large number of agents. As such, Magent2 provides implementations of baseline algorithms, including parameter-sharing DQN, DRQN, and a2c, which are used in agent learning. The platform allows for the development of customized agent behaviors executable in a decentralized manner. The communication and collaboration between agent entities are undertaken using global observations of the environment and of the states of other agents. Similarly to the previous platforms, the state of the system is defined in a step-based manner.

Mava—A comprehensive framework for creating multi-agent reinforcement learning (MARL) that offers modular abstractions. It follows the executor–trainer paradigm inspired by the actor–learner concept in distributed single-agent reinforcement learning. Mava facilitates the interaction between systems and the environment. It supports distributed system training and provides various components and architectures for system creation.

MuJoCo—A physics engine designed to support research and development applications in the robotics and machine learning fields. It is able to handle high-dimensional states and action spaces and supports multiple types of controllers (e.g., position-based, velocity-based, torque-based). The physical interactions that can be modeled in the system include contacts, friction, and joint constraints.

OpenSpiel—A research platform for developing and studying games and algorithms in the domain of reinforcement learning and game theory. It offers several simulations (e.g., n-player zero-sum, cooperative and general-sum, and perfect and imperfect information games) and provides tools for agent-based modeling and analysis.

PySC2—An environment intended for training agents based on Starcraft II that provides a comprehensive simulation of the game. It implements algorithms facilitating training agents using reinforcement learning techniques. The platform supports real-time game-state observation and reward access, enabling an informed decision-making process in agents.

Unity ML-Agents—A game-based platform for developing intelligent agents. It provides tools for creating virtual environments for simulations in 2D, 3D, VR, and AR formats, as well as facilitating the handling of sensory input, such as visual and auditory data. The platform supports both single-agent and multi-agent training, including distributed training in which multiple agents are trained simultaneously.

3.3. Platforms for Modeling and Simulating Environments and Ecosystems

Alchemist—A simulation framework that incorporates chemistry terminology within an agent-based system. Initially, the platform was dedicated strictly to chemistry-oriented multi-compartment stochastic simulations [55]. However, the chemistry-inspired concepts of the Alchemist's meta-model are now treated in an "abstract" manner. For instance, the model defines "nodes", "reactions", "molecules", and "concentrations".

BioDynaMo—A simulation framework designed for multi-scale agent-based modeling of biological systems. Its design principles emphasize two main aspects: (1) a high-performance engine and (2) a modular component design. The agents are used to represent individual cells that reside in a 3D environment and are capable of dividing (creation of new agents), moving, and interacting with each other.

FAME—Toolkit that facilitates the development of agent-based models and simulations for energy systems. It consists of several modules, including FAME-Core, FAME-Prepare, FAME-Io, and FAME-MPI. The agent-based models are being developed using FAME-Core. It aims to provide built-in functionalities to simplify ABM development for programmers without specialized knowledge.

ForestSim—An agent-based model for researching policy and sustainability issues related to woody biomass-based biofuels and bioenergy options. The platform is Java-based and relies on the MASON toolkit for simulation management. It is designed to be modifiable for the purpose of studying more specific research areas.

Framesticks—A system for modeling artificial life and observing the effects of aimless evolutionary mechanisms. It allows for separate simulations of the physical and neural components of a creature, taking into consideration genotype–phenotype mappings. The physical simulation consists of representing the creature's body as a finite number of points and considering the effects of forces applied on it by other points, as well as outside physical forces (i.e., gravity, friction, elasticity). The neural part of the system represents a creature's brain as a neural network.

HexSim—A computer application specifically designed for constructing plant and animal population models in the field of computational biology research. Users can define the structure, complexity, and data requirements of their models within the HexSim platform. It leverages spatial data to capture aspects such as landscape structure, habitat quality, and the distribution of stressors.

LANDIS-II—A modeling framework for simulating and studying the long-term ecological processes in forests. It represents trees and shrubs as individual agents that model their growth, mortality, and regeneration. The communication is undertaken through the effects of the agents' actions observable in the environment. As such, the platform enables the modeling of species interactions, including competition and facilitation

lemlab—An agent-based modeling platform designed to facilitate the development and testing of local energy market (LEM) models. The platform is based on a modular design that incorporates a generic LEM architecture, which allows for the adaptation of the system to meet the research needs of the user. Lemlab includes integrated time-series data for simulating agents as several types of both power producers and consumers, such as household loads, photovoltaic systems, wind turbines, heat pumps, and electric vehicles.

Osmose—An R package for simulation of aquatic wildlife that can represent multiple different species in one environment. Specifically, the platform facilitates the implementation of individual-based and spatially explicit models. Its core concept is based on the co-occurrence of predators and prey. Each specimen (agent) represented in the simulation is characterized by a distinct set of parameters, such as weight, age, and size.

Simona—An agent-based discrete-event modeling tool dedicated to distributed grid simulations. It is primarily used for modeling power grids, their analysis, and planning. The grids are designed using input data that must adhere to the dedicated PowerSystemDataModel format.

TerraME—A programming environment designed to simulate the impact of human-related and natural phenomena across anisotropic regions. It enables the implementation of three distinct modeling paradigms; namely, agent-based models, cellular automata, and network models. The simulations executed in TerraME are based on the events realized in discrete time steps.

3.4. Platforms for Transport-Related Simulations

AGADE Traffic—An agent-based platform for traffic simulation. It utilizes NetLogo agents to model traffic participants and provides a graphical user interface for their visualization. The platform allows the specification of routes for individuals by defining their origins and destinations.

Carla—A platform for autonomous driving simulations. It includes built-in protocols and components for mobility modeling (e.g., city structures and vehicles). It enables control of static and dynamic actors used to represent vehicles.

MATSim—An agent-based platform that provides a set of methods for running and implementing large-scale mobility simulations. It comprises “modules” (e.g., modules with configuration options) that can be used independently or in combination with each other.

Microsoft AirSim—A simulator for ground vehicles and aircraft that has versions using both the Unity engine and Unreal Engine. The environment is open-source and cross-platform and supports both software-in-the-loop simulation (with flight controllers) and hardware-in-loop simulation (for physically and visually realistic simulations). It defines vehicles as autonomous robots composed of shapes and sets of points, mapping them to corresponding forces. These robots are simulated in an environment that incorporates physics phenomena, such as gravity or air density.

The Open Racing Car Simulator (**Torcs**)—A portable, multi-platform game that can be used for both entertainment and research purposes. It incorporates an advanced physical model that allows for realistic car racing scenarios and offers a range of components, including racing tracks, opponents, and cars.

References

1. Davis, R. Report on the Workshop on Distributed AI; MIT Artificial Intelligence Laboratory: Cambridge, MA, USA, 1980.
2. Axelrod, R. The Evolution of Cooperation; Basic Books: New York, NY, USA, 1984.
3. Agha, G.; Hewitt, C. Concurrent Programming Using Actors: Exploiting Large-Scale Parallelism. Readings Distrib. Artif. Intell. 1988, 24, 398–407.
4. Ericsson Computer Science Laboratory Erlang. Available online: https://web.archive.org/web/20230610060832/https://www.erlang.org/ (accessed on 12 April 2023).
5. Lightbend, Inc. Akka. Available online: https://web.archive.org/web/20230606075845/https://akka.io/ (accessed on 12 April 2023).
6. Asynkron AB Proto.Actor. Available online: https://web.archive.org/web/20230610060753/https://proto.actor/ (accessed on 12 April 2023).
7. Benda, M.; Jagannathan, V.; Dodhiawala, R. On Optimal Cooperation of Knowledge Sources—An Empirical Investigation; Technical Report BCS-G2010-28; Boeing Advanced Technology Center, Boeing Computing Services: Seattle, WA, USA, 1986.

8. Bratman, M. Intention, Plans, and Practical Reason; Harvard University Press: Cambridge, MA, USA, 1987.
9. Georgeff, M.P.; Lansky, A.L. Reactive Reasoning and Planning. In Proceedings of the Sixth National Conference on Artificial Intelligence AAAI, Seattle, WA, USA, 13–17 July 1987.
10. Rao, A.S.; Georgeff, M.P. BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA, USA, 12–14 June 1995; pp. 312–319.
11. Fum, D.; Guida, G.; Tasso, C. A Distributed Multi-Agent Architecture for Natural Language Processing. In Proceedings of the 12th Conference on Computational Linguistics, COLING ’88, Budapest, Hungary, 22–27 August 1988; Association for Computational Linguistics: Stroudsburg, PA, USA, 1988; Volume 2, pp. 812–814.
12. Novick, D. Modeling belief and action in a multi-agent system. In Proceedings of the Proceedings , AI, Simulation and Planning in High Autonomy Systems, Tucson, AZ, USA, 26–27 March 1990; pp. 34–41.
13. Torrance, M.C.; Viola, P.A. The AGENT0 manual. Technical Report STAN-CS-91-1389; Stanford University: Stanford, CA, USA, 1991.
14. Shoham, Y. Agent-oriented programming. Artif. Intell. 1993, 60, 51–92.
15. Jennings, N.R. The ARCHON System and its Applications. In Proceedings of the 2nd International Working Conference on Cooperating Knowledge Based System, Keele, UK, 14–17 June 1994.
16. Finin, T.; Fritzson, R.; McKay, D.; Mcentire, R. KQML as an Agent Communication Language; Association for Computing Machinery: New York, NY, USA, 1994; pp. 456–463.
17. Austin, J.L. How to Do Things with Words; Oxford University Press: New York, NY, USA, 1962.
18. Foundation for Intelligent Physical Agents, Agent Communication Language. 1997. Available online: https://web.archive.org/web/20220308163048/http://www.fipa.org/specs/fipa00018/OC00018.pdf (accessed on 12 April 2023).
19. D’Inverno, M.; Luck, M. Engineering AgentSpeak(L): A formal computational model. J. Log. Comput. 1998, 8, 233–260.
20. Jennings, N.; Wooldridge, M. Software Agents. IEE Review 1996, 42, 17–20.
21. Wooldridge, M. What Agents Aren’t: A Discussion Paper; IET: London, UK, 1996.
22. Wooldridge, M. Agent-based software engineering. IEE Proc. Softw. Eng. 1997, 144, 26–37.
23. Ferguson, I. Touring Machines: Autonomous agents with attitudes. Computer 1992, 25, 51–55.

24. Müller, J.; Pischel, M. The Agent Architecture InteRRaP: Concept and Application; Deutsches Forschungszentrum für Künstliche Intelligenz: Kaiserslautern, Germany, 1993.
25. Pomerleau, D.A. ALVINN: An Autonomous Land Vehicle in a Neural Network. In Proceedings of the Advances in Neural Information Processing Systems; Touretzky, D., Ed.; Morgan-Kaufmann: San Mateo, CA, USA, 1988; Volume 1.
26. Jordan, R.E.; Andreas, E.L.; Makshtas, A.P. Heat budget of snow-covered sea ice at North Pole 4. J. Geophys. Res. Ocean. 1999, 104, 7785–7806.
27. Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffmann, G.; et al. Stanley: The Robot That Won the DARPA Grand Challenge. In The 2005 DARPA Grand Challenge: The Great Robot Race; Buehler, M., Iagnemma, K., Singh, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–43.
28. Epstein, J.; Axtell, R. Growing Artificial Societies: Social Science from the Bottom Up; A Bradford Book; MIT Press: Cambridge, MA, USA, 1996.
29. Hammond, R.; Dynamics, E.; Institution, B.; University, J.H. Endogenous Transition Dynamics in Corruption: An Agent-Based Computer Model; Number 19 in Working paper (Center on Social and Economic Dynamics); Center on Social and Economic Dynamics: Washington DC, USA, 2000.
30. Axtell, R.; Epstein, J.; Dean, J.; Gumerman, G.; Swedlund, A.; Harburger, J.; Chakravarty, S.; Hammond, R.; Parker, J.; Parker, M. Population Growth and Collapse in a Multiagent Model of the Kayenta Anasazi in Long House Valley. Proc. Natl. Acad. Sci. USA 2002, 99 (Suppl. S3), 7275–7279.
31. Epstein, J.M. Modeling civil violence: An agent-based computational approach. Proc. Natl. Acad. Sci. USA 2002, 99, 7243–7250.
32. AgentLink Phase I. 1998. Available online: https://web.archive.org/web/20230610060918/https://cordis.europa.eu/project/id/27225 (accessed on 12 April 2023).
33. AgentLink Phase II. 2000. Available online: https://web.archive.org/web/20230610060927/https://cordis.europa.eu/project/id/IST-1999-29003 (accessed on 12 April 2023).
34. AgentLink Phase III. 2004. Available online: https://web.archive.org/web/20230610061017/https://cordis.europa.eu/project/id/002006 (accessed on 12 April 2023).
35. Savaglio, C.; Ganzha, M.; Paprzycki, M.; Bădică, C.; Ivanović, M.; Fortino, G. Agent-based Internet of Things: State-of-the-art and research challenges. Future Gener. Comput. Syst. 2020, 102, 1038–1053.

36. Sasha, O. Agreement Technologies; Law, Governance and Technology Series; Springer: Dordrecht, The Netherlands, 2012.
37. Los, J.; Schulte, F.; Spaan, M.T.J.; Negenborn, R.R. An Auction-Based Multi-Agent System for the Pickup and Delivery Problem with Autonomous Vehicles and Alternative Locations. In Proceedings of the Dynamics in Logistics; Freitag, M., Kinra, A., Kotzab, H., Megow, N., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 244–260.
38. Chen, C.; Koll, C.; Wang, H.; Lindell, M.K. An interdisciplinary agent-based evacuation model: Integrating the natural environment, built environment, and social system for community preparedness and resilience. Nat. Hazards Earth Syst. Sci. 2023, 23, 733–749.
39. Clemen, T.; Ahmady-Moghaddam, N.; Lenfers, U.A.; Ocker, F.; Osterholz, D.; Ströbele, J.; Glake, D. Multi-Agent Systems and Digital Twins for Smarter Cities. In Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, SIGSIM-PADS ’21, Virtual Event, 31 May–2 June 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 45–55.
40. Kozhevnikov, S.; Svitek, M.; Skobelev, P. Smart Grid System for Real-Time Adaptive Utility Management in Smart Cities. In Proceedings of the 13th International Multi-Conference on Complexity, Informatics and Cybernetics (IMCIC 2022), Online, 8–11 March 2022; Volume 1, pp. 4–9.
41. Ghribi, C.; Cali, E.; Hirsch, C.; Jahnel, B. Agent-Based Simulations for Coverage Extensions in 5G Networks and Beyond. In Proceedings of the 25th Conference on Innovation in Clouds, Internet and Networks, ICIN 2022, Paris, France, 7–10 March 2022; Zhani, M., Limam, N., Borylo, P., Boubendir, A., dos Santos, C., Eds.; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2022; pp. 1–8.
42. Xu, J.; Dziong, Z.; Luxin, Y.; Huang, Z.; Xu, P.; Cabani, A. Intelligent multi-agent based C-RAN architecture for 5G radio resource management. Comput. Netw. 2020, 180, 107418.
43. Fazio, M.; Pluchino, A.; Inturri, G.; Le Pira, M.; Giuffrida, N.; Ignaccolo, M. Exploring the impact of mobility restrictions on the COVID-19 spreading through an agent-based approach. J. Transp. Health 2022, 25, 101373.
44. Bădică, A.; Bădică, C.; Ganzha, M.; Ivanović, M.; Paprzycki, M. Multi-agent Spatial SIR-Based Modeling and Simulation of Infection Spread Management. In Proceedings of the Computational Science—ICCS 2021, Krakow, Poland, 16–18 June 2021; Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 440–453.
45. Adenaw, L.; Lienkamp, M. Multi-Criteria, Co-Evolutionary Charging Behavior: An Agent-Based Simulation of Urban Electromobility. World Electr. Veh. J. 2021, 12, 18.

46. Lemiec, M.; Malinowski, K.; Szymoński, M.; Ganzha, M.; Paprzycki, M. Agent-based modelling of car platooning for traffic optimization. In Proceedings of the 2021 4th International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR), Batu Pahat, Malaysia, 6–8 September 2021; pp. 130–137.
47. Pniewski, R.; Sellin, D.; Stankevich, K.; Ganzha, M.; Paprzycki, M. Applying Software Agents to Make City Traffic Management Smarter. In Proceedings of the Second International Conference on Information Management and Machine Intelligence; Goyal, D., Gupta, A.K., Piuri, V., Ganzha, M., Paprzycki, M., Eds.; Springer: Singapore, 2021; pp. 651–659.
48. Pniewski, R.; Stankevich, K.; Ganzha, M.; Paprzycki, M. Modelling and Optimizing City Traffic Using an Agent Platform. In Proceedings of the 2nd International Conference on Artificial Intelligence: Advances and Applications; Mathur, G., Bundele, M., Lalwani, M., Paprzycki, M., Eds.; Springer Nature: Singapore, 2022; pp. 861–868.
49. Anwar, M.B.; Stephen, G.; Dalvi, S.; Frew, B.; Ericson, S.; Brown, M.; O'Malley, M. Modeling investment decisions from heterogeneous firms under imperfect information and risk in wholesale electricity markets. Appl. Energy 2022, 306, 117908.
50. Nunes, I. Capability Relationships in BDI Agents. In Proceedings of the The 2nd International Workshop on Engineering Multi-Agent Systems (EMAS 2014) at AAMAS 2014, Paris, France, 5–6 May 2014; pp. 56–72.
51. Chen, F.; Zhao, Q.; Cao, M.; Chen, J.; Fu, G. Adaptive Agent-Based Modeling Framework for Collective Decision-Making in Crowd Building Evacuation. J. Shanghai Jiaotong Univ. (Sci.) 2021, 26, 522–533.
52. Le, N.T.T. Multi-agent reinforcement learning for traffic congestion on one-way multi-lane highways. J. Inf. Telecommun. 2023, 1–15.
53. Bădică, A.; Bădică, C.; Buligiu, I.; Ciora, L.I. DEVS Modeling and Simulation Using BDI Agents: Preliminary Considerations. In Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, Novi Sad, Serbia, 25–27 June 2018; pp. 1–8.
54. Freeman, C.D.; Frey, E.; Raichuk, A.; Girgin, S.; Mordatch, I.; Bachem, O. Brax—A Differentiable Physics Engine for Large Scale Rigid Body Simulation. arXiv 2021, arXiv:2106.13281.
55. Pianini, D.; Montagna, S.; Viroli, M. Chemical-oriented simulation of computational systems with ALCHEMIST. J. Simul. 2013, 7, 202–215.