Agent-Based Models and Simulations Tools

Subjects: Computer Science, Interdisciplinary Applications

Contributor: Alessia Antelmi, Gennaro Cordasco, Giuseppe D'Ambrosio, Daniele De Vinco, Carmine Spagnuolo

Agent-based models (ABMs) are one of the most effective and successful methods for analyzing real-world complex systems by investigating how modeling interactions on the individual level (i.e., micro-level) leads to the understanding of emergent phenomena on the system level (i.e., macro-level). ABMs represent an interdisciplinary approach to examining complex systems, and the heterogeneous background of ABM users demands comprehensive, easy-to-use, and efficient environments to develop ABM simulations. Many tools, frameworks, and libraries exist, each with its characteristics and objectives.

Keywords: agent-based model ; agent-based simulations ; agent-based tools

1. Introduction

Simulation models are one of the most effective and successful methods for studying real-world phenomena [1]. The term model refers to an abstract and simplified representation of the object of study that only considers the aspects relevant to the investigation; the concept of simulation indicates a model's manifestation realized through software for reproducing its dynamics and providing analyzable results. Among simulation models, agent-based models (ABMs) are one of the most adopted techniques for representing reality using a bottom-up approach [2]. Starting from a simple independent entity called an agent, the modeler can shape the global behavior of a complex system. An agent is an autonomous, independent entity that acts within an environment and interacts with it as well as with other agents, according to a set of rules the modeler defines. The combination of these interactions creates a reproduction of the reality under investigation that the modeler can evaluate to extract valuable data and meaningful information from its emergent behaviors. The resulting model will exhibit patterns, structures, and behaviors that were not explicitly programmed but arise through agents' interactions [3]. For these reasons, ABMs turn out to be a valuable tool for studying, explaining, and predicting complex phenomena, supporting researchers in investigating how the macroscopic behavior of a system depends on the micro-level properties, constraints, and rules [3][4][5].

ABMs are extremely powerful for studying problems centered on an individual's interactions with other individuals or the surrounding environment [4]. This intrinsic characteristic makes ABMs particularly suited to being applied in diverse research fields. Nowadays, ABMs are widely used by researchers in various disciplines, allowing them to test and assess new theories and observe and notice mechanisms never considered before. Applications domains for ABMs include social science [6], economics [7][8], climate change [9][10], epidemiology [11][12], transportation and logistics [13][14], and many others [2][5][15]. The widespread use of ABMs in these various application fields created the need for tools to easily and quickly develop simulations without requiring significant coding skills. Several ABM tools and platforms have been introduced to overcome this requirement by abstracting the complexity of the simulation implementation, allowing modelers to focus on the reality under investigation rather than the coding component.

2. Agent-Based Models and Simulations

2.1. ABM

Although no single formal definition of ABM exists in literature, it can easily identify some key components that ABMs share: agents, environment, and rules [2]. Agents model the living population, the environment determines the setting where the agents act, and the rules define the potential agent-to-agent and agent-to-environment interactions [16].

The main aspect of an agent is its ability to act autonomously in response to the surrounding environment while making decisions to achieve its internal goals. The modeler must define this decision-making process and compose the agent's behavior, which determines how the agent relates to other agents and environmental factors [17]. The agent's behavior includes simple actions such as moving and communicating, but also more complex operations allowing the population to evolve. Each agent maintains its attributes and the information acquired during the simulation within its state, which may

vary during its life cycle. Agents live and act within an environment defining how agents can move and are connected to other agents. An environment has a well-defined topology defined by fields such as spatial grids, continuous spaces, and networks. When the simulation environment must reproduce real-world places, ABMs can exploit Geographic Information System (GIS) data to replicate existing locations like buildings or towns [3][17][18]. All other information related to fields and other non-active objects is included in the environment state. The collection of all agents' and environment's states represents the simulation state, which holds all the information about the model delineating its status at a specific time of the simulation.

The modeler must be able to identify, model, and code agents, environment, and behavioral rules to realize an ABM.

2.2. ABM Tool

The effectiveness of ABMs collides with the difficulty of developing a model for researchers with low expertise in computer science. Therefore, the ABM community made a considerable effort to provide standardized software platforms to design, build, and execute ABMs. ABM tools take away many of the complexities of the model implementation, allowing the user to focus on the simulation outcomes rather than the development process [16][19][20]. ABM tools usually come in the form of frameworks and libraries, providing developers with (i) a framework consisting of a set of standard concepts for designing and describing a model and (ii) a library for implementing the framework, containing tools for the execution and the analysis of the simulation [15][21].

Over the years, numerous ABM platforms have been developed with different objectives and targets. The first discriminant factor is identifiable in the platform's purpose. A tool can be either general or special-purpose [22][23]. In the former case, this characteristic denotes the user can use the ABM platform to model any system of interest. In the latter case, the term special purpose implies that the system is oriented to a specific domain, thus including functionalities to address peculiar situations of a given research field. A further differentiation concerns the main development objective of the tool, usually identifiable in better ease of use or improved efficiency. Some ABM platforms emphasize easy-to-use interfaces with a reasonable learning curve that allows non-experienced programmers to produce models quickly [5][24][25]. The drawback of this approach is low scalability since graphical tools and domain-specific programming languages are not focused on performance. Several ABM toolkits for mainstream programming languages also exist. In this case, they offer high-performance capabilities but require technical skills to use them appropriately [16][19].

2.3. Desiderata of an ABM Tool

Researchers use ABMs to investigate and analyze complex phenomena to understand how each component and the interaction with other elements affect their emerging behavior [5]. Conducting this kind of experiment often demands building elaborated models in terms of the number of agents and the parameters regulating their interactions. The need for implementing and running such models implies the first two fundamentals desiderata of an ABM tool: efficiency and ease of use [5][16][22][26][27]. These two aspects are heavily influenced by the design of the ABM platform and its programming model and are often two conflicting objectives. Some ABM tools expose their functionalities via a GUI (Graphical User Interface) to enhance user experience and grant high ease of use while limiting the achievable model complexity by preventing the user from personalizing and/or adding more complex dynamics. Conversely, frameworks and libraries based on standard programming languages give room for developing complex ABM by offering generic facilities. As a downside, these tools demand adequate technical knowledge, which may result in a higher perceived difficulty [21], usually mitigated via proper documentation, examples, and tutorials [16][25][28].

Developing an ABM means defining different common patterns involving agents' behavior, environment, and interactions. ABM tools should provide ready-to-use methods and interface covering those patterns, allowing the modeler to easily and quickly implement standard actions such as movement and communication, agents' lifecycle and internal state management, environments creation using grids, continuous spaces, and networks, and interaction with the environment [16][29]. Defining a realistic simulation field is a critical feature to accommodate since the simulation environment may provide a rich set of information influencing the agents' behavior. As 80% of the data have a spatial/geographical nature or a geographical component, the ability to work with Geographic Information Systems (GIS) data becomes a fundamental requirement for any ABM tool. GIS-based systems use multiple spatial data models for representing and storing information about phenomena with spatial location and extent [17][18]

Other desiderata relating to the analysis of simulations include facilities for creating a graphical model visualization, tools for statistical and non-statistical analysis, real-time monitoring, and data visualization [3][5][16][19][21][29]. Among tools for

statistical analysis, random number generation assumes tremendous importance since ABMs usually include stochastic processes [3][21][30][31], i.e., processes influenced by a specific random component causing the simulation results to be volatile. Researchers need to handle this stochasticity through random number generators that enable them to reliably reproduce a model's behavior and investigate how random distributions affect it [16][21][32]. The intrinsic stochastic nature of most ABMs also affects the reproducibility of such models. In this context, the automated validation process becomes a critical step in ABM development as simulations must be run several times and their results aggregated [31][33][34].

As the last desideratum, it includes model exploration and optimization capabilities since modelers need to explore the model's parameter space experimenting with how the simulation behaves when given parameters vary [16][19][31][33].

3. ABM Tools

ActressMAS [35] is an agent-based framework written in .NET with the primary objective of being simple to learn and easy to use. ActressMAS is designed to allow the user to focus on the model logic rather than learning the framework, enhancing its accessibility at the expense of performance. According to its developers, ActressMAS should be used for applications that do not require fast execution speed or do not include a considerable number of agents.

AgentPy [36] is an open-source Python library for developing and analyzing ABMs integrated with IPython and Jupyter Notebooks, a web-based interactive development environment. AgentPy is designed for scientific applications and provides features for model exploration, numeric experiments, and advanced data analysis. The library offers functionalities to easily create models and their visualization that can be embedded within Jupyter notebooks. Moreover, AgentPy allows the modeler to run simulations in a parallel environment without writing parallel code.

Agents.jl [37] is a recent framework for agent-based simulations for implementing, running, and visualizing models exploiting the Julia programming language. This framework is mainly centered on granting efficiency and ease of use by exposing methods that allow the user to develop models with few lines of code. Agents.jl is available as a Julia library and is easily usable with the plethora of analytical tools of the Julia ecosystem. It offers the most common ABM-related features, including different environments, support for GIS data, and model exploration capabilities. Agents.jl also supports parallel and distributed computing to empower simulation execution.

Care HPS [38] is a C++ tool for modeling and executing ABMs on high-performance architectures while hiding the complexity of parallel and distributed programming. The tool abstracts the modeler from crucial and tricky tasks such as agent distribution, load balancing, and synchronization. Still, Care HPS remains easily extensible by expert developers.

Cormas (Common-Pool Resources and Multi-Agent Systems) [39] is a simulation platform based on the VisualWorks programming environment and the Smalltalk language. This platform is mainly dedicated to non-computer scientists and offers facilities to build, design, and analyze ABMs; however, it exchanges this ease of use with limited efficiency and scalability. Cormas editor allows the user to define agent behaviors through activity diagrams without including sophisticated features to keep its interface as simple as possible.

CppyABM [40] is a library for ABM development that combines the efficiency of C++ with the availability of Python libraries and exploits CMake to be platform-free. CppyABM offers all functionalities in both languages, enabling users to choose their preferred programming language. CppyABM relies on third-party packages to provide additional functionality while remaining a lightweight library. Other Python or C++ libraries can be installed separately and integrated into CppyABM.

EcoLab [41] is a framework for developing ABMs in C++ and executing them using TCL (Tool Command Language). This tool provides a GUI through the Tk toolkit and supports parallel and distributed processing by exposing utilities to manage communication. The user has to handle synchronization and partitioning manually.

Evoplex [42] is a platform for developing ABM based on C++, using CMake scripts to facilitate compilation and setup, thus making it cross-platform. Evoplex adopts a fully modular approach that separates the core library from the GUI and visualization tools. The APIs exposed by the core library allow the user to develop the model. At the same time, additional components are available to improve ease of use with an interactive GUI and a web visualization tool.

FLAME (FLexible Agent Modeling Environment) [43] is an agent-based modeling system for creating models runnable on most computing systems, ranging from laptops to HPC supercomputers. FLAME provides a formal framework for creating models based on the XXML language, a dialect of XML, that it uses to generate the source code for the simulation in C automatically. The FLAME engine automatically generates parallel code without any effort by the modeler by adopting a new programming language easy to understand.

FLAME GPU [44] is an extended version of the FLAME framework to write ABMs for Graphics Processing Units (GPUs) using the FLAME standard formal XXML language. Thanks to FLAME GPU, the user does not need to explicitly understand GPU programming languages or optimization strategies since the API available uses the FLAME template to generate the simulation program in CUDA for target GPU devices. Visualization of the simulation is available even for a massive agent population without suffering performance loss.

GAMA (Gis & Agent-based Modelling Architecture) [18] is an agent-oriented generic modeling and simulation platform. GAMA grants high ease of use by providing a simple agent-based programming language called GAML that offers a simple formalism to describe all the characteristics of the entities of an ABM. Moreover, the platform can manage simulations with hundreds of thousands of agents with good performance. The modular architecture, separating each aspect of the model into a specific component, and the facilities provided makes GAMA an accessible tool for non-expert developers with minimum learning requirements. This characteristic is enhanced by the full integration of GAMA with the Eclipse IDE, which provides convenient features such as auto-compilation, auto-completion, and the use of templates. Finally, the platform supports the integration of external modules to introduce additional functionalities, such as GAMAR [45], that enable the analysis of simulation results with R.

Insight Maker [46] is a graphical modeling and simulation tool focused on accessibility and availability of features rather than performance. This tool is a web application accessible through a standard web browser and includes features specific to a web environment, like user management and model searching and sharing. The main advantage of Insight Maker resides in its VPL (Visual Programming Language), which offers access to all its functionalities, including tools for data analysis and model exploration and validation. The simulator also includes API to build models and analyze their results programmatically.

JADE (Java Agent Development Framework) [47] is an industry-driven Java FIPA-compliant framework aiming to simplify the implementation of multi-agent systems. Thanks to its features, this tool has established itself as one of the most popular platforms in academic and industrial communities [22]. JADE provides a powerful and useful GUI enabling the user to control and configure the simulation during its execution and also supports debugging and development tasks. Moreover, this tool is designed to work on distributed systems abstracting most of the inherent complexities to the modeler. The core characteristics of JADE make the tool highly scalable, robust, easy to learn, and compatible with most Java-based platforms. Moreover, its popularity grants high user support, with complete documentation and many tutorials and examples available.

JAS-mine (Java Agent-based Simulation library—Modelling In a Networked Environment) [48] is a Java-based toolkit for discrete-event simulations designed to aid ABM development. Specifically, this platform aims to speed up model development, facilitate model documentation, and foster model testing and sharing. The core capabilities of JAS-mine reside in integrating I/O communication functionalities in the form of embedded relational database management systems tools and automatic CSV table creation. The database explorer included in the platform enables the user to inspect the database through Structured Query Language (SQL) style commands.

krABMaga [49] is a fast, reliable, discrete-event multi-agent simulation toolkit based on the Rust language for developing ABMs. Designed to be a ready-to-use tool for the ABM community, krABMaga embraces the architectural concepts of the well-adopted MASON simulation library to provide modelers with a familiar programming environment and decrease the learning curve of the framework. However, krABMaga re-engineered some aspects of the MASON architecture to exploit Rust's peculiarities and programming model. This framework comprises all functionalities required for developing and executing a model, including a visualization component and a convenient UI. Additional functionalities relate to running model exploration jobs on parallel, distributed, and cloud architectures.

MaDKit (Multi-agent Development Kit) [50] is a lightweight Java library for designing and simulating agent systems. The tool follows an organization-centered rather than an agent-centered approach based on the AGR (Agent/Group/Role) model. MaDKit provides several functionalities via APIs, including agents' lifecycle management and distribution, being mainly designed to be used by users with some programming knowledge.

MASON [51] is a discrete-event simulation toolkit written in Java for designing, executing, and visualizing ABMs. MASON provides functionalities and API supporting the most common needs of a modeler, including common agents' behavior, environment creation, and scheduling management. One of the main advantages of MASON is its snapshot system enabling the user to stop and save a simulation and resume it in another machine thanks to the compatibility provided by the Java Virtual Machine. Moreover, thanks to the existing extensions, additional features are available, including GIS data with GeoMASON [52], model exploration with ECJ [53], or the possibility of executing a simulation on distributed

systems and Cloud Computing with DistributedMASON [54]. Further, MASON is well-suited to computationally intensive models or long-running simulations.

MASS (Multi-Agent Spatial Simulation) [55] is a multi-agent and spatial simulation library designed to address the need for parallel ABMs. The architecture is based on the coordinator–worker approach, where the coordinator process spawns workers at different computing nodes to run parallel simulations. MASS automatically manages agent execution and migration as well as the simulation space through several APIs, which facilitate the model development (if the user has some basic knowledge of Java).

Mesa [56] is a Python-based ABM framework providing built-in core components to easily create, visualize, and analyze simulations. Mesa is one of the most used and actively supported ABM libraries, which exploits Python's popularity to provide ease of use and accessibility. One of the main advantages of Mesa is its extensibility allowing users to develop and share their components through an open-source ecosystem. This approach created a rich community providing extensions for any need, including the possibility to exploit a multi-processor system, support for GIS data, and advanced analysis.

NetLogo [57] is an agent-based modeling environment implemented in Java and Scala, and it is considered the standard platform for developing ABMs. The importance and popularity of NetLogo rose to prominence thanks to its community, which is continuously providing extensions such as GIS data usage, 3D visualization, and integration with other languages, such as Python with PyNetLogo [58] or Pylogo [59], or R with RNetLogo [60]. Other relevant extensions worth to be mentioned are HubNet [61] for creating participatory simulations and BehaviorSpace [62] for providing parameter-sweeping capabilities using distributed and parallel techniques. NetLogo allows modelers to develop their models through a simple-to-use dedicated modeling language while offering a VPL to create and edit components to realize any simulation. However, its accessibility leads to significant limitations regarding model complexity.

Pandora [63] is an ABM framework for large-scale distributed simulation providing two identical programming interfaces exposing the same functionalities in two different programming languages. pyPandora allows non-expert developers to develop models using Python quickly. C++ Pandora offers a more efficient interface in C++ to implement complex models, including the automatic generation of parallel and distributed code. Pandora includes Cassandra, a GUI tool with functionalities to design and analyze a single model execution or to set up a model exploration process. This tool can run large-scale ABMs, and deal with thousands of agents with complex behavior.

Repast (REcursive Porous Agent Simulation Toolkit) [64] is a family of agent-based modeling and simulation platforms available in several programming languages. Repast Simphony [65] is a Java-based modeling system that provides automated methods to perform all the common tasks required in a simulation and supports several crucial additional functionalities. The Simphony platform is based on a modular architecture adopting a plugin system that enables adding a wide range of external tools. Any other Repast version implements the core features of Repast Simphony. Repast4Py [66] is a Python-based framework that includes functionalities to develop distributed ABMs.

RepastHPC (Repast for High-Performance Computing) [67] is another member of the Repast suite; specifically, it is a C++-based modeling system designed for running on large computing clusters and supercomputers. This toolkit enables the execution of massive simulations containing hundreds of thousands of agents of very complex behavior whose execution requires high computational power. Although some built-in functions are available for developing a model, RepastHPC still requires users to have good programming experience since they have to manage different aspects of the parallel execution.

3. Conclusions

ABMs are an effective technique for studying complex systems via a bottom-up approach as, through ABM simulations, researchers from different fields can investigate phenomena that are too difficult to understand using traditional methods. ABMs represent an interdisciplinary approach to examining complex systems, and the heterogeneous background of ABM users demands comprehensive, easy-to-use, and efficient environments to develop ABM simulations. Over the years, many tools, frameworks, and libraries have been developed, each with its characteristics and objectives.

There is no perfect ABM platform, but modelers must choose the right tool primarily based on their technical skills and application requirements (e.g., elevated computational loads, GIS data management, visualization). Specifically, modelers must evaluate their confidence in coding in a specific programming language, the amount of advanced and non-advanced functionalities required, and the scale and complexity of the models. Based on these parameters, it is possible to identify the tool that best suits the user's needs, finding the right compromise between ease of use and efficiency. Given their

inherent distributed nature, ABMs lend themselves well to analyzing phenomena from the ever more attractive distributed computing domains such as federated learning [68][69][70][71] and blockchain systems [72][73][74][75].

References

1. Donkin, E.; Dennis, P.; Ustalakov, A.; Warren, J.; Clare, A. Replicating complex agent based models, a formidable task. Environ. Model. Softw. 2017, 92, 142–151.
2. Macal, C.M. Everything you need to know about agent-based modelling and simulation. J. Simul. 2016, 10, 144–156.
3. Macal, C.M.; North, M.J. Tutorial on agent-based modelling and simulation. J. Simul. 2010, 4, 151–162.
4. Siebers, P.O.; Macal, C.M.; Garnett, J.; Buxton, D.; Pidd, M. Discrete-event simulation is dead, long live agent-based si mulation! J. Simul. 2010, 4, 204–210.
5. Abar, S.; Theodoropoulos, G.K.; Lemarinier, P.; O'Hare, G.M. Agent Based Modelling and Simulation tools: A review of t he state-of-art software. Comput. Sci. Rev. 2017, 24, 13–33.
6. García-Magariño, I.; Lombas, A.S.; Plaza, I.; Medrano, C. ABS-SOCI: An Agent-Based Simulator of Student Sociogram s. Appl. Sci. 2017, 7, 1126.
7. Farmer, J.D.; Foley, D. The economy needs agent-based modelling. Nature 2009, 460, 685–686.
8. Bert, F.; North, M.; Rovere, S.; Tatara, E.; Macal, C.; Podestá, G. Simulating agricultural land rental markets by combini ng agent-based models with traditional economics concepts: The case of the Argentine Pampas. Environ. Model. Soft w. 2015, 71, 97–110.
9. Farmer, J.D.; Hepburn, C.; Mealy, P.; Teytelboym, A. A Third Wave in the Economics of Climate Change. Environ. Reso ur. Econ. 2015, 62, 329–357.
10. Hailegiorgis, A.; Crooks, A.; Cioffi-Revilla, C. An Agent-Based Model of Rural Households' Adaptation to Climate Chang e. J. Artif. Soc. Soc. Simul. 2018, 21, 4.
11. Waleed, M.; Um, T.W.; Kamal, T.; Khan, A.; Zahid, Z.U. SIM-D: An Agent-Based Simulator for Modeling Contagion in Po pulation. Appl. Sci. 2020, 10, 7745.
12. Antelmi, A.; Cordasco, G.; Spagnuolo, C.; Scarano, V. A Design-Methodology for Epidemic Dynamics via Time-Varying Hypergraphs. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, Au ckland, New Zealand, 9–13 May 2020; International Foundation for Autonomous Agents and Multiagent Systems: Richl and, SC, USA, 2020; pp. 61–69.
13. Kato, T.; Kamoshida, R. Multi-Agent Simulation Environment for Logistics Warehouse Design Based on Self-Contained Agents. Appl. Sci. 2020, 10, 7552.
14. Serrano-Hernandez, A.; Faulin, J.; Hirsch, P.; Fikar, C. Agent-based simulation for horizontal cooperation in logistics an d transportation: From the individual to the grand coalition. Simul. Model. Pract. Theory 2018, 85, 47–59.
15. Allan, R.J. Survey of Agent Based Modelling and Simulation Tools; Science & Technology Facilities Council: New York, NY, USA, 2010.
16. Castle, C.; Crooks, A. Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations; Worki ng Paper. CASA Working Papers (110); Centre for Advanced Spatial Analysis (UCL): London, UK, 2006; Volume 110.
17. Brown, D.G.; Riolo, R.; Robinson, D.T.; North, M.; Rand, W. Spatial process and data models: Toward integration of ag ent-based models and GIS. J. Geogr. Syst. 2005, 7, 25–47.
18. Taillandier, P.; Gaudou, B.; Grignard, A.; Huynh, Q.N.; Marilleau, N.; Caillou, P.; Philippon, D.; Drogoul, A. Building, com posing and experimenting complex spatial models with the GAMA platform. GeoInformatica 2019, 23, 299–322.
19. Gilbert, N.; Bankes, S. Platforms and methods for agent-based modeling. Proc. Natl. Acad. Sci. USA 2002, 99, 7197–7 198.
20. Bordini, R.H.; Braubach, L.; Dastani, M.; Fallah-Seghrouchni, A.E.; Gómez-Sanz, J.J.; Leite, J.; O'Hare, G.M.P.; Pokah r, A.; Ricci, A. A Survey of Programming Languages and Platforms for Multi-Agent Systems. Informatica 2006, 30, 33–4 4.
21. Railsback, S.F.; Lytinen, S.L.; Jackson, S.K. Agent-based Simulation Platforms: Review and Development Recommend ations. Simulation 2006, 82, 609–623.
22. Kravari, K.; Bassiliades, N. A Survey of Agent Platforms. J. Artif. Soc. Soc. Simul. 2015, 18, 11.

23. Pal, C.; Leon, F.; Paprzycki, M.; Ganzha, M. A Review of Platforms for the Development of Agent Systems. arXiv 2020, arXiv:2007.08961.

24. Rousset, A.; Herrmann, B.; Lang, C.; Philippe, L. A survey on parallel and distributed multi-agent systems for high perfo rmance computing simulations. Comput. Sci. Rev. 2016, 22, 27–46.

25. Nikolai, C.; Madey, G. Tools of the Trade: A Survey of Various Agent Based Modeling Platforms. J. Artif. Soc. Soc. Simu l. 2009, 12, 1–2.

26. Theodoropoulos, G.; Minson, R.; Ewald, R.; Lees, M.; Uhrmacher, A.; Weyns, D. Simulation Engines for Multi-Agent Sy stems. In Multi-Agent Systems: Simulation and Applications; Taylor & Francis: Abingdon, UK, 2009; Chapter 3.

27. Suryanarayanan, V.; Theodoropoulos, G.; Lees, M. PDES-MAS: Distributed Simulation of Multi-agent Systems. Procedi a Comput. Sci. 2013, 18, 671–681.

28. Tobias, R.; Hofmann, C. Evaluation of free Java-libraries for social-scientific agent based simulation. J. Artif. Soc. Soc. Simul. 2004, 7, 1–6.

29. Gupta, R.; Kansal, G. A Survey on Comparative Study of Mobile Agent Platforms. Int. J. Eng. Sci. Technol. 2011, 3, 194 3–1948.

30. Axelrod, R. Advancing the Art of Simulation in the Social Sciences. In Proceedings of the Simulating Social Phenomen a; Conte, R., Hegselmann, R., Terna, P., Eds.; Springer: Berlin/Heidelberg, Germany, 1997; pp. 21–40.

31. Axtell, R.; Axelrod, R.; Epstein, J.M.; Cohen, M.D. Aligning simulation models: A case study and results. Comput. Math. Organ. Theory 1996, 1, 123–141.

32. Heath, B.; Hill, R.; Ciarallo, F. A Survey of Agent-Based Modeling Practices (January 1998 to July 2008). J. Artif. Soc. S oc. Simul. 2009, 12, 9.

33. Bankes, S.C. Agent-based modeling: A revolution? Proc. Natl. Acad. Sci. USA 2002, 99, 7199–7200.

34. Brown, D.; Page, S.; Riolo, R.; Zellner, M.; Rand, W. Path dependence and the validation of agent-based spatial model s of land use. Int. J. Geogr. Inf. Sci. 2005, 19, 153–174.

35. Leon, F. ActressMAS, a .NET Multi-Agent Framework Inspired by the Actor Model. Mathematics 2022, 10, 382.

36. Foramitti, J. AgentPy: A package for agent-based modeling in Python. J. Open Source Softw. 2021, 6, 3065.

37. Datseris, G.; Vahdati, A.R.; DuBois, T.C. Agents.jl: A performant and feature-full agent-based modeling software of mini mal code complexity. Simulation 2022, 003754972110688.

38. Borges, F.; Gutierrez-Milla, A.; Luque, E.; Suppi, R. Care HPS: A high performance simulation tool for parallel and distri buted agent-based modeling. Future Gener. Comput. Syst. 2017, 68, 59–73.

39. Bommel, P.; Becu, N.; Le Page, C.; Bousquet, F. Cormas: An Agent-Based Simulation Platform for Coupling Human De cisions with Computerized Dynamics. In Proceedings of the Simulation and Gaming in the Network Society; Springer: S ingapore, 2016; pp. 387–410.

40. Nourisa, J.; Zeller-Plumhoff, B.; Willumeit-Römer, R. CppyABM: An open-source agent-based modeling library to integr ate C++ and Python. Softw. Pract. Exp. 2022, 52, 1337–1351.

41. Standish, R.K.; Leow, R. EcoLab: Agent Based Modeling for C++ programmers. arXiv 2004.

42. Cardinot, M.; O'Riordan, C.; Griffith, J.; Perc, M. Evoplex: A platform for agent-based modeling on networks. SoftwareX 2019, 9, 199–204.

43. Holcombe, M.; Coakley, S.; Smallwood, R. A general framework for agent-based modelling of complex systems. In Pro ceedings of the 2006 European Conference Complex Systems, Paris, France, 25–29 September 2006.

44. Coakley, S.; Gheorghe, M.; Holcombe, M.; Chin, S.; Worth, D.; Greenough, C. Exploitation of High Performance Comp uting in the FLAME Agent-Based Simulation Framework. In Proceedings of the 2012 IEEE 14th International Conferenc e on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Soft ware and Systems, Liverpool, UK, 25–27 June 2012; pp. 538–545.

45. Hai, H.B.; Contamin, L.; Choisy, M.; Brugière, A. Gamar: An R Interface to the GAMA Platform. Available online: https:// github.com/r-and-gama/gamar (accessed on 31 October 2022).

46. Fortmann-Roe, S. Insight Maker: A general-purpose tool for web-based modeling & simulation. Simul. Model. Pract. Th eory 2014, 47, 28–45.

47. Bellifemine, F.; Poggi, A.; Rimassa, G. Developing multi-agent systems with a FIPA-compliant agent framework. Softw. Pract. Exp. 2001, 31, 103–128.

48. Richiardi, M.G.; Richardson, R.E. JAS-mine: A new platform for microsimulation and agent-based modelling. Int. J. Micr osimulation 2017, 10, 106–134.

49. Antelmi, A.; Cordasco, G.; D'Auria, M.; De Vinco, D.; Negro, A.; Spagnuolo, C. On Evaluating Rust as a Programming L anguage for the Future of Massive Agent-Based Simulations. In Proceedings of the Methods and Applications for Mode ling and Simulation of Complex Systems; Springer: Singapore, 2019; pp. 15–28.

50. Gutknecht, O.; Ferber, J. The MadKit Agent Platform Architecture. In Proceedings of the Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems; Wagner, T., Rana, O.F., Eds.; Springer: Berlin/Heidelberg, German y, 2001; pp. 48–55.

51. Luke, S.; Cioffi-Revilla, C.; Panait, L.; Sullivan, K.; Balan, G. MASON: A Multiagent Simulation Environment. Simulation 2005, 81, 517–527.

52. Sullivan, K.; Coletti, M.; Luke, S.; Crooks, A. GeoMason: Geospatial Support for MASON. 2010. Available online: http s://cs.gmu.edu/~eclab/projects/mason/extensions/geomason/ (accessed on 31 October 2022).

53. White, D.R. Software review: The ECJ toolkit. Genet. Program. Evolvable Mach. 2012, 13, 65–67.

54. Cordasco, G.; Scarano, V.; Spagnuolo, C. Distributed MASON: A scalable distributed multi-agent simulation environme nt. Simul. Model. Pract. Theory 2018, 89, 15–34.

55. Chuang, T.; Fukuda, M. A Parallel Multi-agent Spatial Simulation Environment for Cluster Systems. In Proceedings of th e 2013 IEEE 16th International Conference on Computational Science and Engineering, Sydney, Australia, 3–5 Decem ber 2013; pp. 143–150.

56. Kazil, J.; Masad, D.; Crooks, A. Utilizing Python for Agent-Based Modeling: The Mesa Framework. In Proceedings of th e Social, Cultural, and Behavioral Modeling; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 3 08–317.

57. Uri, W. Modeling nature's emergent patterns with multi-agent languages. In Proceedings of the EuroLogo 2001, Linz, A ustria, 21–25 August 2001.

58. Jaxa-Rozen, M.; Kwakkel, J.H. PyNetLogo: Linking NetLogo with Python. J. Artif. Soc. Soc. Simul. 2018, 21, 4.

59. Abbott, R.; Lim, J. PyLogo: A Python Reimplementation of (Much of) NetLogo. In Proceedings of the 11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications—SIMULTECH, INSTICC, SciT ePress, Online, 7–9 July 2021; pp. 199–206.

60. Thiele, J.C. R Marries NetLogo: Introduction to the RNetLogo Package. J. Stat. Softw. 2014, 58, 1–41.

61. Jiang, L.; Zhao, C. The Netlogo-Based Dynamic Model for the Teaching. In Proceedings of the 2009 Ninth International Conference on Hybrid Intelligent Systems, Shenyang, China, 12–14 August 2009; Volume 2, pp. 49–53.

62. Railsback, S.F.; Ayllón, D.; Berger, U.; Grimm, V.; Lytinen, S.; Sheppard, C.; Thiele, J. Improving Execution Speed of M odels Implemented in NetLogo. J. Artif. Soc. Soc. Simul. 2017, 20, 3.

63. Rubio-Campillo, X. Pandora: A Versatile Agent-Based Modelling Platform for Social Simulation. In Proceedings of the S IMUL 2014, The Sixth International Conference on Advances in System Simulation, Nice, France, 12–16 October 2014; pp. 29–34.

64. North, M.J.; Collier, N.T.; Vos, J.R. Experiences creating three implementations of the repast agent modeling toolkit. AC M Trans. Model. Comput. Simul. 2006, 16, 1–25.

65. North, M.J.; Collier, N.T.; Ozik, J.; Tatara, E.R.; Macal, C.M.; Bragen, M.; Sydelko, P. Complex adaptive systems modeli ng with Repast Simphony. Complex Adapt. Syst. Model. 2013, 1, 1–26.

66. Collier, N.T.; Ozik, J.; Tatara, E.R. Experiences in Developing a Distributed Agent-based Modeling Toolkit with Python. I n Proceedings of the 2020 IEEE/ACM 9th Workshop on Python for High-Performance and Scientific Computing (PyHP C), Atlanta, GA, USA, 13 November 2020; pp. 1–12.

67. Collier, N.; North, M. Parallel agent-based simulation with Repast for High Performance Computing. Simulation 2013, 8 9, 1215–1235.

68. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-Efficient Learning of Deep Netwo rks from Decentralized Data. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Cad iz, Spain, 9–11 May 2016.

69. Polato, M. Federated Variational Autoencoder for Collaborative Filtering. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–8.

70. Ghimire, B.; Rawat, D.B. Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated L earning for Internet of Things. IEEE Internet Things J. 2022, 9, 8229–8249.

71. Imteaj, A.; Amini, M.H. Leveraging asynchronous federated learning to predict customers financial distress. Intell. Syst. Appl. 2022, 14, 200064.
72. Roesch, M.; Linder, C.; Zimmermann, R.; Rudolf, A.; Hohmann, A.; Reinhart, G. Smart Grid for Industry Using Multi-Agent Reinforcement Learning. Appl. Sci. 2020, 10, 6900.
73. Zhang, Z.; Yang, T.; Liu, Y. SABlockFL: A blockchain-based smart agent system architecture and its application in federated learning. Int. J. Crowd Sci. 2020, 4, 133–147.
74. Połap, D.; Srivastava, G.; Yu, K. Agent architecture of an intelligent medical system based on federated learning and blockchain technology. J. Inf. Secur. Appl. 2021, 58, 102748.
75. Rincon, J.; Julian, V.; Carrascosa, C. FLaMAS: Federated Learning Based on a SPADE MAS. Appl. Sci. 2022, 12, 3701.