Modeling an Epidemic - Multiagent Approach Based on an Extended SIR Model

Mihailo Ilić(✉) and Mirjana Ivanović

Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia
{milic,mira}@dmi.uns.ac.rs
https://www.pmf.uns.ac.rs/

Abstract. This paper highlights the use of software agents in simulating real world phenomena. A brief overview of different approaches and tools for developing software agents and simulating real world phenomena are given. One of the more recent tools was utilized in this paper to develop a model of disease spread in a population of agents. Multiple factors affect the longevity of a pathogen in a given population, one of which is the deadliness of the disease whose impact is shown in this paper.

Keywords: Software agents · Agent systems · Modeling and simulation · Epidemics

1 Introduction

Software Agents, in the broadest sense, are programs which act on behalf of someone else. These agents are defined by their behaviour and their ability to act with a degree of autonomy. The first mention of an idea resembling today's understanding of software agents dates back to the seventies, when Hewitt described his *actor model* [14]. These actors communicate through messages, and can modify only their own private state. If they intend to modify the states of other actors, they must do that by sending messages, to which the target actor reacts to.

The majority of authors agree on some *universal* definitions [10], such as they are based on the actor model, and that all software agents share common properties [20,21] which include: *Autonomy* - they act on their own; *Social ability* - agents usually interact with other agents; *Reactivity* - they are able to react when prompted by other agents or when changes in the environment occur; *Proactiveness* - agents are not only reactive, but they actively take action which in most cases is goal-oriented, meaning they wish to achieve a specific goal.

Agents are capable of acting and reacting when confronted with other agents and the environment they're in. During the history of developing and using software agents various programming languages and frameworks have been developed.

N. T. Nguyen et al. (Eds.): ICCCI 2021, LNAI 12876, pp. 141–153, 2021.
https://doi.org/10.1007/978-3-030-88081-1_11

Programming Languages for Agent Development - Commonly used programming languages like Java, C, C++ and others all have their own tools and frameworks for handling agents and multi-agent systems. On the other hand, there exist various prototype languages which provide useful abstractions when constructing agents and agent based systems. According to [10], these languages can be classified into multiple groups based on their most relevant aspects regarding agent modeling: *Agent oriented programming (AOP) languages*, *Belief-desire-intention languages (BDI)* and *Hybrid languages*.

Agent Platforms - Various tools and platforms for developing agent systems and agent modeling have been developed over the years. A large number of both open source and commercial tools are available today, since agent based modeling is a powerful tool researchers can benefit from. The authors in [16] split modern agent frameworks into 2 main categories:

1. ***General purpose***: open source; commercial
2. ***Special purpose***: Cognitive, social and affective agents; Artificial intelligence research; Modeling and simulation; Transport-related simulations

The advantage of *general purpose platforms* is their lack of domain focus, which makes them flexible and allows for a wide range of uses. Because of this, various widely used programming languages are used to implement them, while some even have *graphical user interfaces*. Open source variants, such as AgentScript [1], JADE [5,11], Mesa [15] and Repast [7] use general purpose programming languages such as *Java, C#, JavaScript and Python*, while other like JaCaMo [4] rely on specialized languages for agent modeling like *AgentSpeak* [19] which is a BDI language. Commercial tools like AnyLogic [2] and Wolfram SystemModeler [8] provide a large number of features and supporting software, along with industrial strength simulations.

Special purpose platforms are designed to effectively handle specific problems. By sacrificing flexibility, they provide tools and mechanisms designed for solving specific problems. For modeling human behaviour the ACT-R [17] platform is used while for modeling relationships between societies and their environments, Cormas [3] is used. In artificial intelligence research, tools like MAgent [22] allow for multiagent reinforcement learning while MADP [6] is used in research of decision planing and learning in agent systems.

The goal of this paper is to illustrate how software agents can be applied in modeling real-world systems, such as the spread of disease. Multiple factors influence the longevity of a pathogen in a given population, which is shown in experiments conducted during this research.

The rest of the paper is organised as follows. The second section covers work done in the field of epidemic modeling. In the third section, a brief overview of the Mesa agent modeling framework is given, along with the explanation of the disease model constructed for the purpose of this experiment. Results of the simulations are shown in Sect. 4. Concluding remarks are given in the last section.

2 Related Work

Agent systems give the power to model behaviour of entities capable of making decisions when interacting with their environment. It is possible to simulate complex systems which include hundreds or thousands of agents. These models can be simple or complex, depending on the domains and problems that are planned to be solved by agent system. This is the reason that they find use in both academic research and industry. Intensive use of agents can be found in numerous domains like video games, transportation and logistics, power grids, medicine and many other.

The authors in [12] acknowledge the use of agent systems in urban planning, specifically in the field of road network planning. Existing systems model the behaviour of drivers in normal conditions, with the goal to optimise traffic light sequences among other things. However, these models fail to provide support for modeling traffic in uncommon conditions, which can be caused by a wide variety of events, such as natural disasters. The authors propose a new model, which takes into account finer details of entities involved in traffic [12]. These properties include car length and maximum speed, personality of drivers along with road infrastructure. The decision of a single driver to leave his/her car during a crisis event can have a huge impact on the entire system, blocking hundreds of other cars. The authors propose a new model called MOSAIIC, which takes into account these finer details while modeling traffic. The idea of simulating and handling such complex systems in unpredictable situations can be transferred to studying and simulating epidemiological outbreaks.

Agent systems also find their use in the field of medicine as well, more specifically - epidemiology. The authors in [9] recognize the potential of applying agent systems in helping to understand the dynamics of an epidemiological outbreak. By using the *GAMA* simulation development environment, a standard SIR (*Susceptible-Infected-Recovered*) model where each agent can be either Susceptible, Infected or Recovered has been constructed. The agents are placed in a continuous space, where each agent is a point in 2D space capable of moving through the environment with traits as speed and direction defined.

The authors in [18] give a general overview of agent based systems, including an explanation of their computational construction. They emphasize that it is not imperative to include every single detail in modeling complex systems, as technical limitations don't allow this. However, it is suggested that adding as much traits of the system as possible helps in recreating the real world system which allows researchers to create precise models. The agent model depicted in [18] is also focused on disease spread, relying on the standard SIR model which is also used in [9]. The opposite to the approach shown in paper [9], in our approach agents are placed in a grid space, where each cell in the grid represents a "room" which multiple agents can share. This way, the disease can only spread between agents located in the same cell. The simulation in [18] is conducted on a 20×20 grid, with each agent at a fixed position and the disease can only spread between agents in adjacent cells of the grid. Our approach presented in the paper builds upon the idea presented in [18] by allowing multiple agents to be located in the

same cell, and also allowing agents to migrate to adjacent cells at each step of the simulation.

As stated in [13], *human mobility* is one of the main factors which impact how fast a disease can spread through a population. The authors mention recent outbreaks such as SARS in 2003 and H1N1 in 2009. We are now experiencing a similar situation with the current COVID-19 pandemic. Given enough data, researcher can model the spread of disease on different levels (for example city wide or even world wide). In [13], a demonstration was conducted by modeling the metropolitan area of Zurich, Switzerland. Since the population size is roughly 1.5 million people, only 1% of the population (15 286 agents) was used to model the spread of seasonal flu. The authors propose linking transport and epidemiological modeling in order to simulate outbreaks of infectious diseases more precisely. They also mention the fact that the results of the simulation can vary depending on the number of parameters taken into account. Simpler models only account for healthy, infected and recovered individuals, while other parameters such as vaccination, immunity, gender, age etc. could help in giving more accurate results.

In this paper a simulation of an epidemic is conducted. Agents are placed on a grid, and are able to move between cells. Cells represent some shared space (e.g. a room or building), and on this level infected agents have a chance of spreading the disease to healthy agents. This chance increases if the healthy agent is exposed to multiple infected agents occupying the same cell. Our simulation extends the standard SIR model by including a new *deceased* state, which represents agents who have succumbed to the disease.

3 Modeling the Spread of Disease

One of the main goals of this paper is to present how one could model a real world problem using software agents. The topic of infectious diseases, more precisely viruses, and how they spread is catching more attention with the spread of *COVID-19*. The ongoing pandemic was also an inspiration for our model and simulations. By understanding how viruses affect the human body and how they spread, we can reduce their negative effect. The latter can be modeled using software agents, as shown in [13].

As we briefly presented in previous parts of the paper there are still some flexible, widely used programming languages, which support adequate utilization of software agents in solving specific problems. In our experiments we will use the Python-based Mesa framework.

Mesa Overview - *Mesa* is a recent open-source framework written in Python, which gives users the ability to quickly construct agent models. These models can then be easily visualized and analyzed using various other Python frameworks and libraries. One useful tool which this framework has built in [15] is the data collector which will be showcased in this paper.

The main concepts this framework revolves around are the *Model* and the *Agent*. These core classes serve as the building blocks of any agent model. Mesa

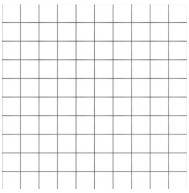
is highly modular [15], which means it doesn't make any assumptions in the way of how a model should operate and also provides multiple different *schedulers* which are in charge of handling the activation of agents.

The agent is the main actor in a Mesa model. A class which extends the *mesa.Agent* class needs to override it's *step* method. A Mesa model defines the passage of time as a series of *steps*, and by overriding this method in the Agent class, we can define the behavior of an agent at each step of the simulation.

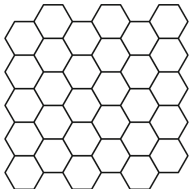
The model class models the entire system, which includes the agents and their environment. The class which inherits the *mesa.Model* class serves as a container for all other elements and global parameters which are used to run the simulation.

The space component represents the space which is populated by agents (Fig. 1). Multiple components are located in the *mesa.space* module, some of which are:

- *Continuous space* - Agents are treated as points and they can have an arbitrary position.
- *Grid* - This is the base class for a square grid. Multiple other classes extend the functionality of this one, like a grid realized as a torus.
- *HexGrid* - Represents a hexagonal grid.



(a) A 10x10 grid space



(b) A hexagonal grid space

Fig. 1. Examples of different *Grid* space components

Each space component provides functionality for selecting agents at specific coordinates, along with iterating through neighbouring cells or finding neighbouring agents within a certain radius. A type of grid space will be used in our simulation.

Another core component is the *scheduler*, located in the *mesa.time* module. Schedulers are components that we will use in our simulations, which control when agents are to be activated and run their step methods. The activation order can be crucial for the outcome of the entire system, so it is important to choose the right one. Some of the predefined schedulers are:

- *Base scheduler* - Calls each agent's step function in the order they were added.
- *Random activation* - Activates agents in a random order.
- *Simultaneous activation* - It simulates simultaneous activation of every agent.

Data collectors are a built in tool which enable users to collect multiple types of data such as model/agent level data, along with tables.

3.1 Definition of the Disease Model

In order to run a simulation and collect data, all of the components mentioned in the previous section need to be put in use. The goal of our experiment is to model disease spread among a population of agents. As stated in [13], one of the main factors of disease spread is agent mobility. Hence, we need to define a space which the agents can traverse in this model. For this purpose, a *MultiGrid* from the *mesa.space* module will be used. In this paper, cells in the grid will be analogous to rooms in a building, with agents being able to move between adjacent rooms (cells). Agents in this case represent people moving around inside a building. The reason behind using a MultiGrid is the possibility of having multiple agents occupy a cell. Unlike the standard SIR model used in [9,18] where each agent can either be susceptible, infected or recovered, we considered an additional state - "deceased", for agents which do not survive the infection. Accordingly, each agent can be in one of four states:

1. *Healthy* - Healthy agents which are not carriers of said disease.
2. *Infected* - Infected agents are carriers of the disease.
3. *Deceased* - Such agents have succumbed to the disease which is being modeled and no longer have the ability to move between cells and cannot infect further agents.
4. *Immune* - Agents which have survived the initial infection, and are now immune to the pathogen.

If an agent is infected, it has a chance of recovery or death. Once an agent recovers, it becomes immune and can no longer be infected. These values are defined globally, which means all agents are the same in terms of age, sex, immunity etc. In order to obtain more accurate models, one could implement these agent-level properties, but for this paper these factors haven't been taken into account. Infected agents also have the possibility of spreading the disease to another healthy agent occupying the same cell. If a healthy agent is located at a cell which is also occupied by infected agents, the chance of the infection spreading to the new agent is calculated as in [18].

$$P_{infection}[state_{a,s+1} = Infected | state_{a,s} = Healthy] = 1 - (1 - c)^k \quad (1)$$

The chance of an agent a to transition from a healthy state to an infected state between steps s and $s+1$ of the simulation depends on the number of infected agent k occupying the same cell. In this equation, c represent the initial chance of infection, which is a global parameter of the model.

At the end of each step, an agent also has the ability to migrate to a different cell. The chance of this occurring is also configured globally. Each agent can perform the following actions: *transmit the disease; migrate to another cell.* They also have predefined global chances to: *recover from the disease; pass away* - if this happens to an agent, it becomes inactive and performs no more actions during the simulation.

We used a random scheduler during modeling. To ensure the same results each time the simulation is run, it is imperative to set the seed of the random number generator to a fixed value.

A data collector was also used in order to collect data after each step in the simulation. The following values were computed after each step:

1. The number of healthy agents in the entire grid. Immune agents will also be included, since they have recovered from the disease.
2. The number of infected agents in the grid.
3. The number of deceased agents in the entire grid.
4. Multiple matrices containing the number of each type of agent according to their health status per cell.

Finally, all of these components are aggregated in the *DiseaseModel* class which extends the *mesa.Model* class. This class takes in multiple parameters for defining the simulation:

- *N* - The number of agents in the system.
- *Width* - The width of the grid.
- *Height* - The height of the grid.
- *Chance of death* - The possibility of one succumbing to the disease.
- *Chance of spread* - The chance of the disease spreading from one infected agent to a healthy one.
- *Chance of the illness ending* - The chance for an agent to reach the end of the illness, either by recovering and becoming immune or by succumbing to the disease.
- *Chance of migration* - The chance of an agent moving to an adjacent cell.
- *Infected on start ratio* - Defines the ratio of the entire population which will be disease carriers at the start of the simulation.

4 Experimental Results

By utilising the model described in Sect. 3.1, one could model the impact of a disease with different starting parameters. One key parameter is the chance of death of the disease. Pathogens rely on the bodies of hosts they inhabit, where they live and replicate at the expense of their victim. As all living things, their goal is to live long enough to replicate their genetic material. By doing too much harm to the bodies of their hosts and killing them off too soon, the pathogens lower the chance they have to effectively replicate and spread themselves to other hosts. As a result of them being to deadly (too harmful to their hosts), they have a negative effect on themselves.

It is possible to simulate this using the model mentioned in Sect. 3.1. Two different simulations will be run, with common parameter setting being: number of agents - 500, grid dimensions - 10×10, number of simulation steps - 100, chance of spread - 4.7%, chance of the illness ending - 16.67%, chance of migration - 50%, ratio of infected agents at the start of the simulation - 1%.

The authors in [18] concluded in their simulations that the chance of spread of 4.7% would yield a result of one infected agent infecting 1.6 healthy agents on average. Hence, we will use this value in our simulations as well. Furthermore, in [18] an agent carries the infection 3 to 6 steps in total. Similarly, with the chance of the illness ending set to 16.67%, an agent in our simulation will stop carrying the disease after 6 steps on average.

To see the effect the chance of death has on the population and on the spread of the pathogen, two different simulations will be run with different values for this parameter.

Lower Chance of Death - The first test conducted was with a lower chance of death in comparison to the second test discussed in the rest of the section. According to the World Health Organisation, as of April 2021 around ***2.1%***[1] of confirmed COVID-19 cases ended in death. With a chance of death set to this value, and 5 agents infected at start (1% of the entire population), the progress of the disease can be seen in Fig. 2.

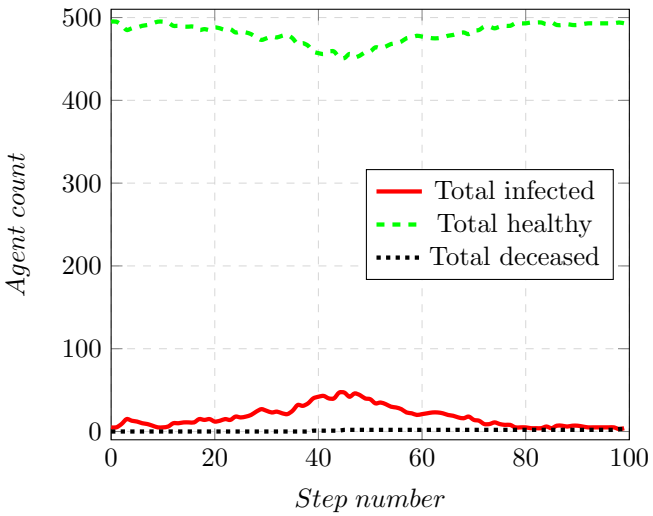


Fig. 2. The timeline of the disease during the 100 simulated steps with a morality rate of 2.1%

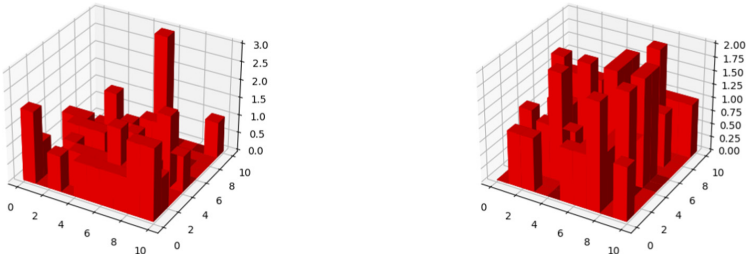
A slow but steady increase in the number of infected agents can be seen in the first half of the simulation. The peak is measured at step 44, with 47 infected agents at that time.

In Table 1, the agent count for each state is shown at some key steps. The pathogen persisted for the entirety of the simulation which lasted for 100 steps. Key steps include step 39 which marked the first death from the disease, and step 44 where the peak was reached.

[1] https://covid19.who.int/ - Official WHO worldwide COVID-19 statistics.

Table 1. Agent count by their status at various steps

Step #	Infected #	Healthy #	Deceased #
0	5	495	0
5	12	488	0
10	5	495	0
15	11	489	0
20	12	488	0
30	25	475	0
39	**40**	**459**	**1**
40	42	457	1
44	**47**	**452**	**1**
50	39	459	2
99	4	493	3



(a) Step 39 - First death occurred. (b) Step 44 - The peak of infections.

Fig. 3. Infected agents at key steps

Figures 3a and 3b visualise the number of infected agents on the entire grid at key steps.

Higher Chance of Death - This second simulation had a chance of death of 7%, which is more than 3 times greater than in the previous experiment.

When comparing Fig. 4 to Fig. 2, it is possible to see that the peak of infections happens earlier, and is much lower than in the first experiment. The pathogen acts too aggressively to it's host, with a 7% chance of killing the carrier, while only having a 4.7% chance of spreading. This makes it harder for the disease to make it to another host, and thus prolong the existence of the pathogen in the population. As noted in Table 2, the first death occurred earlier, at step 14, and the peak of infections is at step 23. As a result of this higher chance of death, the pathogen died out by step 50, with no more infected agents present at that time.

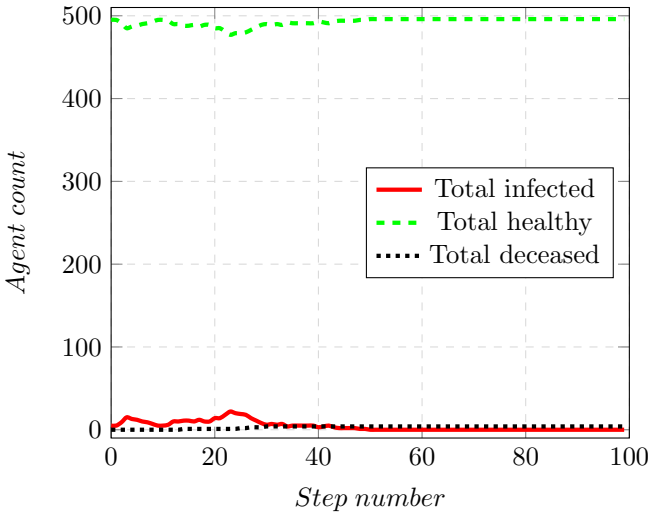


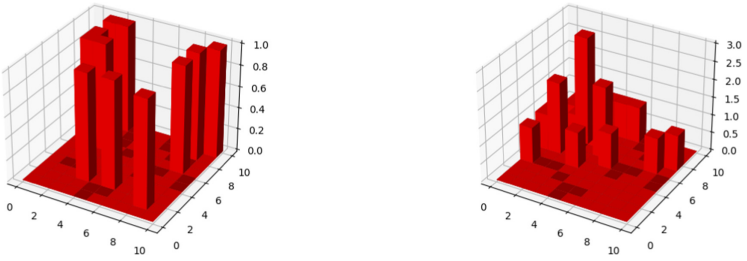
Fig. 4. The timeline of the disease during the 100 simulated steps with a mortality rate of 7%

Figures 5a and 5b visualise the number of infected agents on the entire grid at key steps of the simulation with a chance of death set to 7%.

By looking at Figs. 6a and 6b, we can see what difference the change in the chance of death made on the overall impact on the pathogen and population of agents. The disease which had a lower chance of death managed to reach far more agents in the population than it's counterpart with the higher mortality rate. At the end of the first experiment, 54.4% of agents are immune, but the pathogen

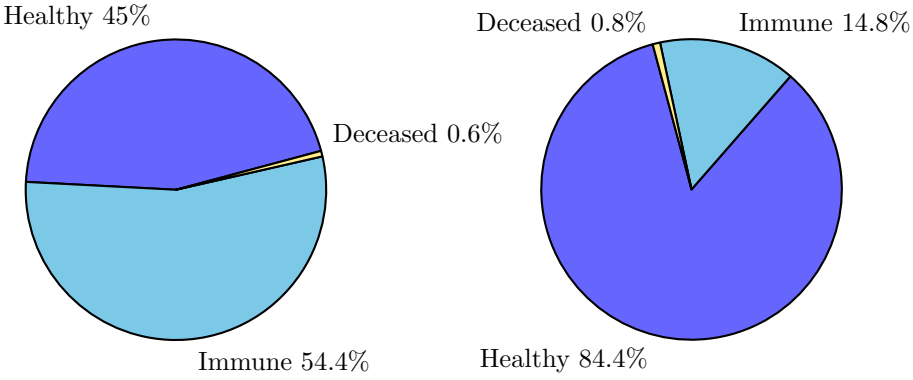
Table 2. Agent count by their status at various steps

Step #	Infected #	Healthy #	Deceased #
0	5	495	0
5	12	488	0
10	5	495	0
14	**11**	**488**	**1**
15	11	488	1
20	14	485	1
23	**22**	**477**	**1**
30	6	490	4
40	3	493	4
50	**0**	**496**	**4**



(a) Step 14 - First death occurred. (b) Step 23 - Peak of infections.

Fig. 5. Infected agents at key steps.



(a) State distribution for experiment 1. (b) State distribution for experiment 2.

Fig. 6. The distribution of healthy, immune and deceased agents at the end of both simulations.

is still present in the population, while in the second experiment, only 14.8% of all agents have become immune. Furthermore, the second pathogen failed to survive the entirety of the simulation. It was eliminated from the population at step 50.

5 Conclusion

Software agents are a powerful mechanism for simulating different processes in a wide range of domains. Given enough information of some real-world system, it is possible to simulate how it can evolve through time and how it will react to different starting conditions and different events which could unfold. Even subtle changes in starting parameters could give wildly different results. Systems such as proposed in [13] which take into account the fine details can help researchers get a better sense of the problems they face.

The value and versatility of software agents is further bolstered by the number of existing programming languages and tools which assist in working with them. These tools can cover a wide variety of use, with some being general purpose and others being more suited for specific fields of research.

In this paper a newer general purpose agent framework was utilised [15] in modeling the spread of disease through a population of agents. Even with a fairly simple model, it is possible to obtain interesting results. Further research of this topic would include adding finer parameters to the agents, making each one of them distinct in their behaviour, and looking into how their decision making impacts the overall spread of a pathogen.

Acknowledgements. The authors acknowledge financial support of the Ministry of Education, Science and Technological Development of the Republic of Serbia (Grant No. 620 451-03-68/2020-14/200125).

References

1. Agentscript project page. https://github.com/backspaces/agentscript/. Accessed 15 Apr 2021
2. Anylogic. https://www.anylogic.com/features/. Accessed 15 Apr 2021
3. Cormas. http://cormas.cirad.fr/indexeng.htm. Accessed 15 Apr 2021
4. Jacamo project. http://jacamo.sourceforge.net/. Accessed 15 Apr 2021
5. Java agent development framework. https://jade.tilab.com. Accessed 15 Apr 2021
6. Madp project. https://github.com/MADPToolbox/MADP. Accessed 15 Apr 2021
7. Repast. https://repast.github.io/. Accessed 15 Apr 2021
8. Wolfram systemmodeler. https://www.wolfram.com/system-modeler/. Accessed 15 Apr 2021
9. Bădică, A., Bădică, C., Ganzha, M., Ivanović, M., Paprzycki, M.: Multi-agent simulation of core spatial sir models for epidemics spread in a population. In: 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), pp. 1–7. IEEE (2020)
10. Bădică, C., Budimac, Z., Burkhard, H.-D., Ivanovic, M.: Software agents: languages, tools, platforms. Comput. Sci. Inf. Syst. **8**(2), 255–298 (2011)
11. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-agent Systems with JADE, vol. 7. Wiley, New York (2007)
12. Czura, G., Taillandier, P., Tranouez, P., Daudé, É.: MOSAIIC: city-level agent-based traffic simulation adapted to emergency situations. In: Takayasu, H., Ito, N., Noda, I., Takayasu, M. (eds.) Proceedings of the International Conference on Social Modeling and Simulation, plus Econophysics Colloquium 2014. SPC, pp. 265–274. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20591-5_24
13. Hackl, J., Dubernet, T.: Epidemic spreading in urban areas using agent-based transportation models. Future Internet **11**(4), 92 (2019)
14. Hewitt, C., Bishop, P., Steiger, R.: A universal modular actor formalism for artificial intelligence, IJCAI3. In: Proceedings of the 3rd International Joint Conference on Artificial Intelligence, pp. 235–245 (1973)
15. Masad, D., Kazil, J.: Mesa: an agent-based modeling framework. In: 14th PYTHON in Science Conference, pp. 53–60. Citeseer (2015)
16. Pal, C.-V., Leon, F., Paprzycki, M., Ganzha, M.: A review of platforms for the development of agent systems. arXiv preprint arXiv:2007.08961 (2020)

17. Ritter, F.E., Tehranchi, F., Oury, J.D.: Act-R: a cognitive architecture for modeling cognition. Wiley Interdiscip. Rev. Cognit. Sci. **10**(3), e1488 (2019)
18. Shoukat, A., Moghadas, S.M.: Agent-based modelling: an overview with application to disease dynamics (2020)
19. Weerasooriya, D., Rao, A., Ramamohanarao, K.: Design of a concurrent agent-oriented language. In: Wooldridge, M.J., Jennings, N.R. (eds.) ATAL 1994. LNCS, vol. 890, pp. 386–401. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-58855-8_25
20. Wooldridge, M., Jennings, N.R.: Agent theories, architectures, and languages: a survey. In: Wooldridge, M.J., Jennings, N.R. (eds.) ATAL 1994. LNCS, vol. 890, pp. 1–39. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-58855-8_1
21. Wooldridge, M.J., Jennings, N.R.: Intelligent agents: theory and practice. Knowl. Eng. Rev. **10**(2), 115–152 (1995)
22. Zheng, L., et al.: Magent: a many-agent reinforcement learning platform for artificial collective intelligence. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)