



Development of Scenarios for Modeling the Behavior of People in an Urban Environment



Alexander Anokhin, Sergey Burov, Danila Parygin , Vyacheslav Rent, Natalia Sadovnikova , and Alexey Finogeev

Abstract The aim of the study is to improve intelligent methods for supporting city management tasks by monitoring the state of processes in the urban environment and deliberately changing their parameters in accordance with decisions obtained using predictive modeling. The chapter provides an analysis of the current state of the cyber-physical problem of modeling processes in systems with people interaction, existing methods for modeling the people movement in an urban environment, and projects for modeling the people movement in a city based on a multi-agent approach. The process of developing scenarios for moving agents in an urban environment is shown. The main components of the software solution responsible for simulating human behavior are presented.

Keywords Multi-agent modeling · Urban environment · Behavior scenarios · Decision support · Urban mobility · Modeling system

A. Anokhin (✉) · S. Burov · D. Parygin · V. Rent · N. Sadovnikova
Volgograd State Technical University, Lenina Ave. 28, 400005 Volgograd, Russia
e-mail: alex.anokhin.st@gmail.com

S. Burov
e-mail: sergey.burovic@gmail.com

D. Parygin
e-mail: dparygin@gmail.com

V. Rent
e-mail: vyacheslav61g@gmail.com

N. Sadovnikova
e-mail: npsn1@ya.ru

A. Finogeev
Penza State University, Krasnaya Str. 40, 440026 Penza, Russia
e-mail: alexeyfinogeev@gmail.com

1 Introduction

Decisions effectiveness improving when choosing options for urban environment development is one of the main problems of modern large cities. New approaches to the management of urban processes are required because of the increase in population, the complexity of urban processes, and the tightening of requirements for the quality of urban services provided. Models suitable for studying various options for changing the urban environment, organizing activities, and choosing the most effective solutions can be built on the basis of an analysis of the functions of the city, the processes, and behavior of residents in it [1].

The multi-agent approach is considered among many methods for modeling such systems. Such an approach is to reduce the initial complex problem into a set of simple tasks, and each “simple” problem is solved by a special program called an agent. An agent is an independent software system that consists of program objects that have the ability to receive an impact from the outside world, determine its reaction to this impact and, in accordance with this, form a response action.

Agents as software entities are able to perceive the environment and respond depending on the situation in which they are [2]. They are goal oriented, i.e. they receive tasks and perform them, interacting with each other and with the environment.

The problem of increasing the adequacy of the developed models requires the development of new approaches to modeling the behavior of actors taking into account the interaction with the environment, other actors and the implementation of a wide range of behavioral scenarios.

2 Existing Approaches to Modeling Human Behavior

Multi-agent modeling is currently an actively developing and promising area of IT [3]. A multi-agent system is a system of several interacting intelligent agents (programs). Entities in such a system are active (agents that respond to each other and the environment) and have individual characteristics. Modeling takes place from the bottom up, thus, behavior at the individual level forms the global level of the system, which allows achieving from low to high levels of abstraction. The simplicity of the initial implementation of such a system is due to incomplete data on the environment and the agents themselves that may cause subsequent difficulties in the interactions formalization [4].

The agent-based approach has been shown to be effective in describing a wide variety of processes. Nevertheless, there are too many unsolved problems associated with the implementation and ensuring the adequacy of multi-agent systems [5]. So, there are many approaches to modeling the behavior of intelligent agents in multi-agent systems. It makes sense to consider the main ones.

2.1 *Modeling the Behavior of Intelligent Agents in Multi-Agent Systems*

Rule Based Behavior Model. The organization of a rule-based behavior model assumes that the agent will behave identically in each similar situation, guided by the rule that the programmer set for him. Rules can be entered based on the current state of the agent, or they can be absolute and not depend on it. Setting rules based on the state of the agent or its parameters allows to provide some variety of behavior, but can't achieve realistic behavior [6].

Behavior Model Based on the Finite State Machine. The behavior of an agent with various states can be modeled on the basis of finite state machines. The state may reflect the conditions in which the agent is: hunger, thirst, fatigue, and others. The most famous variant of this approach is called GOAP and is successfully used in game projects on a global scale (F.E.A.R., S.T.A.L.K.E.R and others) [7]. This approach implies that the activity of the agent is regulated by his state and the purpose of this state. For example, the agent in a state of "hunger" must achieve the goal of "eat" before it can switch to another state. The need for an agent to achieve a goal determines the need to build a chain of actions aimed at achieving this goal.

Finite state machines allow you to simulate a fairly varied and realistic behavior of the agent. However, the addition of new states to the model involves a substantial revision of the entire model in order to add new relationships between the new and each of the existing states [8].

Behavioral Model Based on Behavior Tree. The behavior tree is an oriented acyclic graph whose nodes are agent behavior options. The tree starts from the root node, which sends the child a signal for execution and receives the state of the nodes in response: "Running" if the node is still running, "Success" in the case of successful implementation and "Failure" in case of failure. Variable agent behavior can be modeled using such a system. Also, the behavior tree, being a kind of finite state machine, has an undeniable advantage in the form that with the increase in the number of available states, the complexity of the tree does not grow as fast as the complexity of the automaton [9, 10].

Model of Social Forces. The model is based on Newtonian dynamics to describe the movement of pedestrians and shows several natural behavioral phenomena of pedestrians in the process of movement: pedestrians choose the shortest path; move at an individual speed, taking into account the situation, gender, age, restrictions; keep a certain distance from each other. The distance depends on the pedestrian flow density and speed. The pedestrian movement in the model of social forces is described by the sum of the forces acting on it. The position in space, the speed and acceleration of a pedestrian at any given time can be learned by compiling and solving a system of differential equations describing the action of social forces [11, 12].

The necessary accuracy of calculations and sufficient realism of pedestrian behavior can be obtained with the right settings. But the operation of the algorithm is characterized the by low speed with the high complexity of implementation, as

well as the need for high computing power. Moreover, the model has a low level of abstraction (micro level) [13].

2.2 Application of Multi-Agent Modeling in Gaming and Research Systems

It is also worth considering specific applications of multi-agent modeling. There are a number of applications designed to solve modeling problems in transport systems. Among such applications can be noted TRANSIMS, MIRO, MobiSim, ARCHISIM, SimMobility Freight and others.

There are also alternative multi-agent platforms such as GAMA, MadKit, Repast, Jade and NetLogo, which include basic components for creating agents from any data set, as well as the ability to perform large-scale modeling with millions of agents [14]. High-level agent programming languages, such as NetLogo or GAML, are offered to refine such systems.

The platforms under consideration can be universal, like GAMA, or highly specialized, like MATSim [15, 16], which has tools for modeling traffic flow using queue theory. The main advantage of the platforms under consideration over competitors is the use of multi-agent modeling, due to which the possibility of micromodeling of traffic flows and the environment is achieved [17, 18].

A multi-agent approach is also used in most computer games to simulate the behavior of non-player characters (NPC) [19]. It makes sense to pay attention to some specific games in the context of considering ways of modeling the behavior of the urban population.

Thus, in the series of games The Elder Scrolls [20], the Radiant AI game artificial intelligence system [21] is implemented, which describes the behavior of NPC and their behavior throughout the day. The behavior of the characters is subject to a plan according to which at a certain time of the day the character sleeps, takes food or walks around the city, while visiting places of interest and meeting with other characters for short conversations. This approach to modeling enables to create a realistic model of pedestrian movement in a settlement.

The Grand Theft Auto [22] series of games implements a larger-scale model of the city, which includes, in addition to pedestrians, vehicles, rail and air services. The optimization in this series is designed so that it does not clearly track the movement of all agents: characters that are not in the player's field of vision do not physically exist in the game, and appear only when they enter the field of view. A traffic system in which cars drive only on roads and observe traffic signals can be noted among the advantages of the model implemented in these games.

More realistic traffic system implemented in the Mafia game series [23]. Cars in these games observe the speed limit, independently determine from which row they can turn, and wait for the police in the event of an accident. In addition, the game is

able to track traffic violations: police can write a fine or detain a player for speeding, driving in the oncoming lane or through a red traffic signal, and for traffic accidents.

Multi-agent modeling can be successfully used in games where it is designed to diversify the game environment and provide a comfortable immersion of the player in its world, and in serious geographic information systems (GIS). The ability to model each agent with special properties individually is actively used in GIS. Due to this, it is possible to carry out accurate modeling of urban processes and predict changes in the urban environment.

3 The Proposed Approach to the Development of an Agent's Behavior Model in an Urban Environment

An adequate model in some approximation, which corresponds to the simulated part of the space, can be obtained by describing the basic rules of interaction and the necessary properties of agents. Each agent in the developed system is a model of a person (an actor with free will) or other entities of a living and artificial origin, who are participants in a conditionally dependent dynamic interaction and capable of transporting material and information resources.

The proposed method for constructing a model of state change allows working with events of a measurable scale. Rules of behavior are based on the essence of the roles and properties of the personality that shape its needs. The realization of needs launches certain scenarios of behavior, which, ultimately, are expressed in the sequence of actions of agents. The consequence of each action is a certain state in which the actor goes into and which causes a reaction due to rules, physical, social, legal and other restrictions in accordance with the conditions of the real world. A set of behavioral patterns of actors forms an integral model of the macro level [24].

An approach using the BDI paradigm [25] (a model of beliefs, desires, and intentions) was used in this study.

Beliefs is information about the patterns and condition of the environment that an agent can receive. The assumption is made that the agent's information about the world can be erroneous and incomplete, therefore it is only the agent's view of the world, and not reliable fundamental laws and environmental information.

Desires is the aggregate of all the goals that the agent would like to achieve. Desires must be consistent with each other. It is assumed that the agent will not be able to realize all his desires (goals), so he must limit their list and choose the most important ones.

Intentions is a collection of plans and scenarios for achieving desires. Intentions determine the direction of activity. The agent is trying to find ways in which he could fulfill his intentions. Intentions limit future choices, because an agent cannot create and accept new intentions that are incompatible with those that are already in the process of implementation. Intentions have a long life. The agent will make new

plans in case of failure of the current, until it succeeds. An intention can be deleted from the list only if the agent realizes that the intention is not feasible [26].

The BDI approach is based on the desire to analyze a person's mental activity, processes and understanding how and on what basis people make decisions. The basic algorithm of the BDI agent actions is as follows:

1. Determination the goals to which the agent will strive.
2. The choice of those goals, the implementation of which the agent will try to achieve.
3. Determination of methods and scenarios for achieving selected goals.

The behavior of an agent can change under the influence of factors such as weather conditions in the area of its location, the presence in the field of activity of certain physical objects, as well as the activities of other agents. The result of choosing a scenario for achieving goals based on a combination of data from the world and other agents is the impact: the movement of a person in space, as well as the opposite effect on other agents.

4 Development of the Module for Human Behavior Simulation

It was decided to implement a client–server architecture in order to separate the functionality of the application. The C# programming language, which has proven itself to be the best in the speed of developing complex projects, was used to write the server side of the system.

The client part of the system is written using the JavaScript language. The OpenLayers open source library was chosen as the framework for working with the map [27]. The WebSocket communication protocol is used to implement closer interaction between the browser and the platform, with the ability to work in interactive mode with support for real-time applications.

“.NetTopologySuite” framework used to work with map objects [28]. This framework is a port of the JTS Topology Suite framework written in Java.

Geometry classes support modeling points, lines, polygons, and collections. The geometry is linear in the sense that the boundaries are determined by linear interpolation between the vertices. The geometry is embedded in a 2-dimensional Euclidean plane. The vertices of the geometry can also have a value of Z .

User-defined accuracy models are supported for geometric coordinates. Computations are performed using algorithms that provide reliable geometric calculations for all exact models.

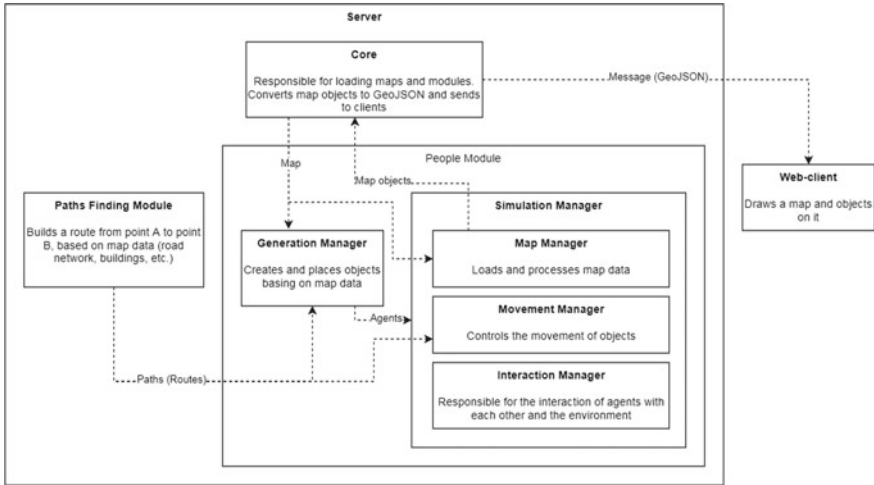


Fig. 1 Brief diagram of system components in the form of UML 2.0 components diagram

4.1 System Architecture

An extensible motion simulation system has been developed to meet the requirements described above. The architecture of this system is client-server, and also divides the program into a kernel and modules that implement individual tasks. This architecture is shown in Fig. 1.

The main part of the system is the server, which is responsible for the simulation. MapManager loads the map data on which the simulation will take place before starting the simulation. GenerationManager creates all objects and agents and places them on the map using its submodules. After that, each agent has a destination on the map to which it must get.

MovementManager is responsible for moving agents around the map and updating the values associated with it. The InteractionManager module is used to implement the interaction of agents with objects and with each other. StatisticManager collects statistics for all agents and objects in the system from the moment the simulation starts, and also provides access to statistics data from past simulations.

4.2 Development of Pedestrian Moving Algorithms

The construction of the pedestrian behavior algorithm will be considered as an example. In general, the process of modeling the movement of people is reduced to updating the state of objects with a certain frequency and changing this state, as well as the number of objects themselves and many other.

The system needs to create new agents to get started (see Fig. 2a). The scene is filled with the necessary number of agents (pedestrians) at the start of the program according to the developed algorithm. Each agent is assigned a route consisting of the starting and ending points of the path, as well as a set of points through which the agent must pass along the road. The agent disappears from the scene and must be replaced by a new agent as soon as it reaches the endpoint of its path.

The process of modeling agent behavior is as follows. The agent is initially assigned a standard scenario: following from the start point to the end with periodic updating of the state of the agent under the influence of external and internal factors. An agent may be assigned a new, non-standard scenario, as a result of updating the state: for example, following another agent, stopping movement or changing a route (see Fig. 2b).

The process of executing standard and non-standard scripts will be discussed later. The standard scenario involves following from the starting point of the route where the agent was created (left the house, from work or from transport) to the destination

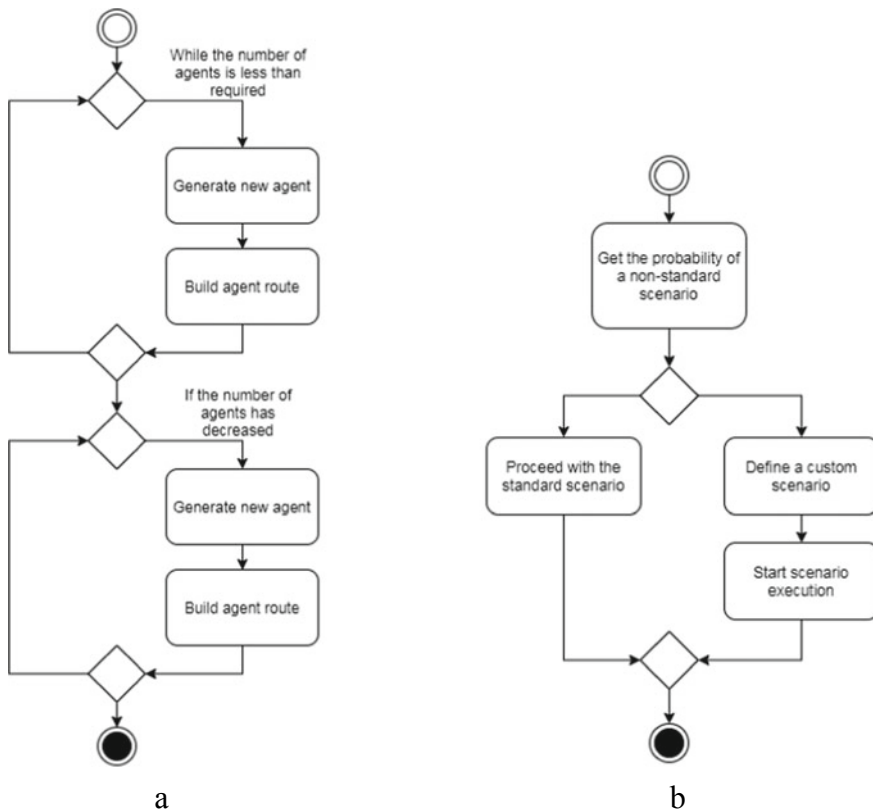


Fig. 2 Algorithms to run simulations: **a** algorithm of filling the scene with agents, **b** scenario assignment algorithm

point (work, home, store), where it will be deleted (see Fig. 3a). The agent path is indicated by dots. The option of waiting one agent for another with subsequent joint movement to a given point is a non-standard scenario (see Fig. 3b).

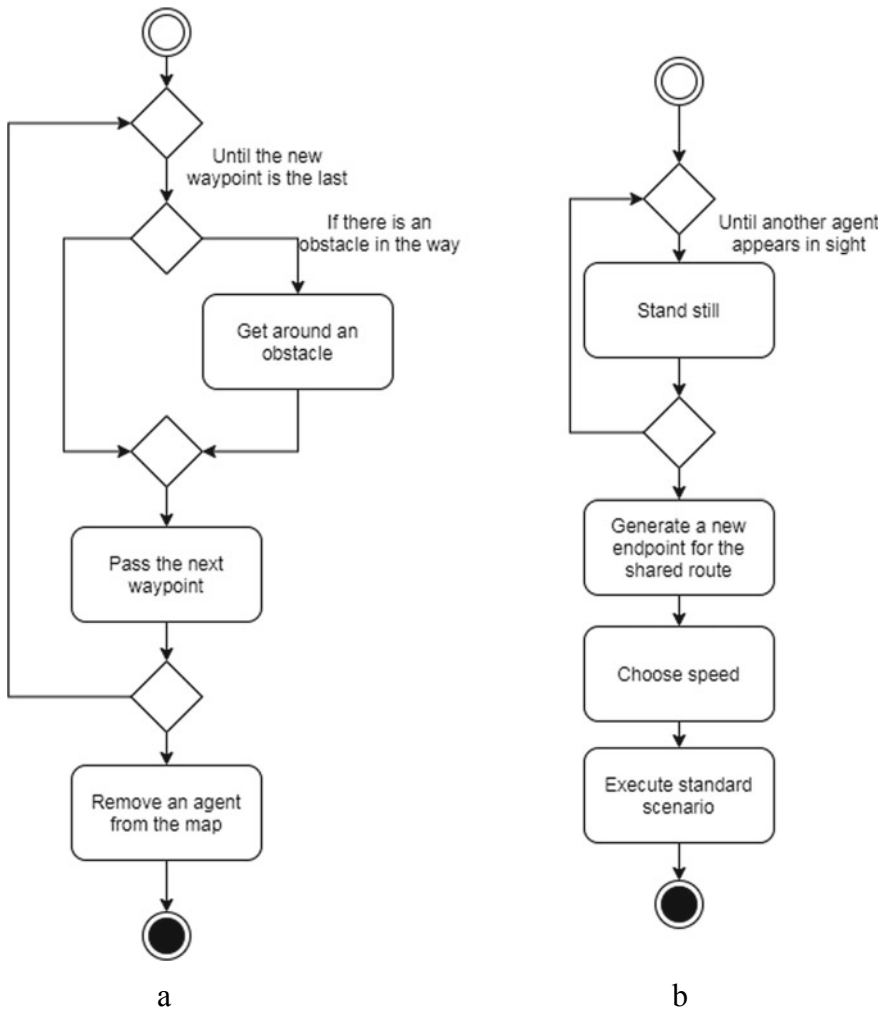


Fig. 3 Pedestrian moving algorithms: **a** standard scenario, **b** non-standard scenario



Fig. 4 Visualization of pedestrian behavior

4.3 Pedestrian Behavior Implementation Example

At the moment, several models of behavior of actors in the urban environment have been implemented. Modeling pedestrian behavior will be discussed in more detail below.

Figure 4 shows the result of modeling pedestrian behavior. The main program window is occupied by a map, on top of which individual agents (pedestrians) are shown in circles. As a result, points (simulated objects) appear on the map at the same coordinates where they are located on the server.

The main scenarios of pedestrian behavior are indicated by the numbers in Fig. 4:

1. The standard scenario of a pedestrian following from a start point to an end point.
2. The movement of pedestrians relative to each other if the speed of one pedestrian is higher than the speed of another.
3. The situation of stopping both pedestrians in one place, for example, for conversation.
4. Joint walking of pedestrians in one direction (the speed of this pair is equal to the speed of the slowest pedestrian).
5. The scenario in which one pedestrian waits for another in order to continue moving together.

The running model visualization is always available on the project website <https://live.urbanbasis.com/> [29]. The software package core implements simulation at the multiprocessor computing cluster of VSTU [30].

5 Conclusion

The main methods for modeling agent behavior and tools for its implementation are considered in the study. The main approaches to the construction of scenarios of the behavior of actors are studied: the behavior of individual agents in a multi-agent system is modeled on the example of pedestrians. The developed model demonstrated such advantages as the ability to individually customize each actor, due to which high variability of the behavior of the same type of entities is achieved. The bulkiness and the need for high computing power for operation can be noted among the shortcomings of the resulting model. The tasks for further correction of the model taking into account the current results are identified.

A generalized behavior model of the urban system, as a system describing the interaction of independent agents, system actors, is planned to be developed by complicating the behavior models of actors, as well as increasing the complexity of interactions between elements of the system. This will make it possible to predict changes in processes depending on changes in parameters (properties) of the urban environment. The presence of this forecast will allow to timely influence the current situation by choosing options for changing the situation depending on the set performance indicators. It is especially important that the simulation results can help in decision-making in the event of emergencies that have not previously occurred.

Acknowledgements The reported study was funded by Russian Foundation for Basic Research (RFBR) according to the research project No. 18-37-20066_mol_a_ved. The results of part 3 were obtained within the Russian Science Foundation (RSF) grant (project No. 20-71-10087). The authors express gratitude to colleagues from UCLab involved in the development of Live.UrbanBasis.com project.

References

1. Ustugova, S., Parygin, D., Sadovnikova, N., Finogeev, A., Kizim, A.: Monitoring of social reactions to support decision making on issues of urban territory management. *Procedia Computer Science* **101**, 243–252 (2016)
2. Timm, I.J., Woelk, P.-O., Knirsch, P., Tönshoff, H.-K., Otthein, H.: Flexible mass customisation: managing its information logistics using adaptive co-operative multiagent systems. In: 6th International Symposium on Logistics, Salzburg, Austria, pp. 227–232 (2001)
3. Parygin, D., Nikitsky, N., Kamaev, V., Matokhina, A., Finogeev, A., Finogeev, A.: Multi-agent approach to distributed processing of big sensor data based on fog computing model for the monitoring of the urban infrastructure systems. In: 5th International Conference on System Modeling & Advancement in Research Trends, pp. 305–310. IEEE (2017)
4. Yanishevskaya, A.G., Pesterev, P.V.: The architecture of the multi-agent search. *Dyn. Syst. Mech. Mach.* **6**(2), 94–101 (2018)
5. Anokhin, A., Sadovnikova, N., Kataev, A., Parygin, D.: Modeling of agents behavior to implement gaming artificial intelligence. *Caspian J. Contr. High Technol.* **2**(50), 85–99 (2020)
6. Tsalgatidou, A., Loucopoulos, P.: Rule-based behaviour modelling: specification and validation of information systems dynamics. *Inf. Softw. Technol.* **33**(6), 425–432 (1991)

7. Goal-Oriented Action Planning. <https://alumni.media.mit.edu/~jorkin/goap.html>. Last accessed 2020/04/20.
8. Syahputra, M.F., Arippa, A., Rahmat, R.F., Andayani, U.: Historical theme game using finite state machine for actor behaviour. *J. Phys: Conf. Ser.* **1235**, 012122 (2019)
9. Sekhavat, Y.A.: Behavior trees for computer games. *Int. J. Artif. Intell. Tools* **26**(02), 1730001 (2017)
10. Anokhin, A., Kataev, A.: Finite-automaton model for controlling the behavior of intelligent agents in educational games. *Inf. Technol. Sci. Educ. Manag.* **4**(14), 75–80 (2019)
11. Helbing, D., Molnar, P.: Social Force Model for Pedestrian Dynamics. *Phys. Rev. E* **51**(5), 4282–4286 (1998)
12. Wang, P.: Understanding social-force model in psychological principles of collective behavior. <https://arxiv.org/abs/1605.05146>. Last accessed 2020/05/12
13. Benjamin, P., Erraguntla, M., Delen, D., Mayer, R.: Simulation modeling at multiple levels of abstraction. In: 1998 Winter Simulation Conference, IEEE, Washington, DC, pp. 391–398 (1998)
14. Parygin, D., Usov, A., Burov, S., Sadovnikova, N., Ostroukhov, P., Pyannikova, A.: 2020) Multi-agent approach to modeling the dynamics of urban processes (on the Example of Urban Movements. *Commun. Comput. Inf. Sci.* **1135**, 243–257 (2020)
15. Umnitsyn, M., Nikishova, A., Omelchenko, T., Sadovnikova, N., Parygin, D., Goncharenko, Y.: Simulation of malicious scenarios using multi-agent systems. In: 7th International Conference on System Modeling and Advancement in Research Trends, IEEE, Moradabad, pp. 3–9 (2018)
16. Global MATSim scenario for the 2018 FIFA World Cup in Russia. <https://www.otslab.ru/en/the-global-matsim-scenario-for-the-fifa-world-cup-2018-in-russia>. Last accessed 2020/03/20
17. MATSim scenario for Krasnoyarsk. <https://www.otslab.ru/en/the-matsim-scenario-for-krasnoyarsk>. Last accessed 2020/02/02
18. Borshchev, A., Filippov, A.: From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools. In: 22nd International Conference of the System Dynamics Society, Oxford, England (2004)
19. Warpefelt, H.: The non-player character: exploring the believability of npc presentation and behavior. <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A912617&dsid=9143>. Last accessed 2020/05/17
20. The Elder Scrolls. <https://elderscrolls.bethesda.net/ru>. Last accessed 2020/05/14
21. Bethesda's Radiant AI as the future of role-playing games, <https://dtf.ru/flood/16882-radiant-ai-ot-bethesda-kak-budushchee-rolevyh-igr>, last accessed 2020/04/22.
22. Grand Theft Auto V. <https://www.rockstargames.com/>. Last accessed 2020/03/25
23. Mafia. <https://mafia-game.com/>. last accessed 2020/03/10
24. Johnson, J., Sarkisian, N., Williamson, J.: Using a micro-level model to generate a macro-level model of productive successful aging. *Gerontologist* **55**(1), 107–119 (2015)
25. Caillou, P., Gaudou, B., Grignard, A., Truong, C.Q., Taillandier, P.: A simple-to-use BDI architecture for agent-based modeling and simulation. *Adv. Intell. Syst. Comput.* **528**, 15–28 (2017)
26. Samigulina, G.A., Samigulina, Z.I.: Cognitive agent development for SMART management systems. *Inf. Technol. Sci. Educ. Manag.* **4**(14), 39–43 (2019)
27. A high-performance, feature-packed library for all your mapping needs. <https://openlayers.org/>. Last accessed 2019/09/16
28. .NetTopologySuite. <https://github.com/nettopologysuite/nettopologysuite>. Last accessed 2019/10/09
29. OsmLifeSimulation. <https://live.urbanbasis.com/>. Last accessed 2020/06/01
30. Multiprocessor computing complex (cluster). <https://evm.vstu.ru/index.php/labs/hpc-lab/about-hpc>. Last accessed 2020/05/31