Chapter 3

Agent-Based Modeling and Simulation of Biological Systems

Şebnem Bora and Sevcan Emek

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.80070

Abstract

Agent-based modeling and simulation is a powerful technique in simulating and exploring phenomena that includes a large set of active components represented by agents. The agents are actors operating in a real system, influencing the simulated environment and influenced by the simulated environment. The agents are included in the simulation model as model components performing actions autonomously and interacting with other agents and the simulated environment to represent behaviors in the real system. In this chapter, we describe how to develop an agent-based model and simulation for biological systems in Repast Simphony platform, which is a Java-based modeling system. Repast Simphony helps developers to create a scenario tree including displays of agents, grid and continuous space, data sets, data loaders, histogram, and time charts. At the end of this chapter, we present case studies developed by our research group with references to demonstrate local behavior of biological system.

Keywords: agent, agent-based modeling and simulation, biological systems, Repast Simphony

1. Introduction

In recent years, agent-based applications have been developed inspired by natural systems. The natural systems have a dynamic structure defined by a complex, distributed, open, heterogeneous, and large-scale systems. Therefore, it is too hard to model these systems in the artificial world. Agent-based modeling and simulation (ABMS) technique has advantage in explanation of the dynamics of the behavior in the complex systems including biological, physical, and social systems. ABMS which is used in the solution or modeling of a problem

in the literature seems to be inspired by living systems. Living systems offer an organization and operation at different levels ranging from the genetic to the social experience. The most common applications that can be shown in living systems are biological systems including human physiology which examine major systems such as cardiovascular system, immune system, nervous system, endocrine system, etc., and predator-prey relationship in the ecosystem, birds and fish flocks, organisms that live in colonies such as foraging ants, bees, wasps, and termites, and etc. [1].

ABMS allows the researchers an experimental experience to create, analyze, and explicate the relationship between the artificial and the real world. In comparison with other modeling approach based on mathematical and numerical analysis, control theory, biomechanical techniques, etc., ABMS is referred to as "individual-based model" [2]. Individual is called agent which has a set of attributes and autonomous behavior. Agents are situated in some set of spaces and time. Agents interact with other agents in the simulation environment. The simulation environment includes agents that perform their actions and achieve their goals.

In this chapter, we will focus on the use of computer simulation for building the agent-based models in biological systems. This chapter intends to provide brief descriptions of the agent-based models that illustrate how to build and implement case studies, which reflect the relationship in the real world.

This chapter is organized as follows: Section 2 gives a brief overview of ABMS; Section 3 presents the description of Repast Simphony toolkit which has ability to display and schedule in real time; Section 4 provides implementation of case studies involving different scenarios to better understand ABMS phenomena; and Section 5 concludes with a brief summary of this chapter.

2. Agent-based modeling and simulation

Agent-based modeling and simulation (ABMS) can be defined in very diverse disciplines like artificial intelligence, complexity science, game theory, etc. [3, 4]. ABMS provides a suitable simulation modeling technique for the analysis of complex systems and emergent phenomena in biological systems, social sciences, economy, management systems, etc. [5, 6]. ABMS is a computational model implemented as computer simulation in which there are individual entities and their behaviors and interactions. It focuses on rules and interaction among the individuals or components of the real system. In the ABMS, the systems are characterized by the autonomous and independent entities known as agents performing some kind of behaviors (action and interactions) in the simulation environment [7]. In the literature, it is possible to see many examples of agent-based modeling in the different fields including traffic control, biomedical research, ecology, energy analysis, etc. [4].

ABMS has advantage of creating a model compared to traditional approaches. No any set of formulas or mathematical equations are needed to build an agent-based model. ABMS focuses on the rules that will determine the behaviors of agents [8]. In order to develop an

agent-based model, firstly, it must be understood how to design and implement the model. In other words, the scenario of a real system must determine the limitations of the model. Some questions must be answered to initialize the model design, like what the agents should be in the model, what the agents' environment is, how to interact with each other and environment, how to define the rules determined the behaviors of agents, what are roles of the agents in the model, etc. [9].

There are some simulation software toolkits to perform ABMS [10]. Toolkits can facilitate to manage the simulation process. One of the most popular toolkits in the literature is Repast Simphony supported by libraries of predefined methods and functions [11, 12].

3. Repast Simphony

Repast (Recursive Porous Agent Simulation Toolkit) Simphony is an agent-based modeling and simulation framework based on the object-oriented programming using Java language. It is free and open source so that it offers the users the widespread use of the agent development environments. Repast Simphony uses Eclipse-integrated development environment (IDE) for developing computer code [13]. Repast Simphony tool offers researchers a flexible way to write models including graphical user interface, toolbar to control the simulation processes (start, step, pause, stop, exit, etc.), displaying agents and their environment, monitoring the output data (time chart, histogram bar), scheduling of simulations, parameter management, data sets, data loaders, etc. Repast Simphony is the most suitable simulation framework for agent-based model development. Classes of agents and their interactions are displayed in Repast Simphony. The output data are graphically presented in time charts and/or histogram bar. Repast Simphony allows the users to record inbuilt data to txt files and displays as movies or images. Also, the users obtain the snapshots of graphics and/or display. Repast Simphony has advantage to display, schedule, analyze, update, or manipulate a running simulation in real time.

After downloading the latest version on Repast Simphony from its web page, creating a new Repast Simphony project is very easy. The first step is to run Eclipse IDE. After the new Repast Simphony Project, which includes a source directory, and default package is created, the scenario directory structure is prepared by creating agent classes.

To build an agent-based model, it is necessary to create classes. More agent classes can be created according to the scenario of the model. The classes include any number of methods to describe the attributes and roles of agents. Setup or step methods are called for each iteration of the simulation. In the each iteration of the simulation, the simulation runtime is described with time steps or tick counts. During the simulation runtime, the agents perform their actions. The get and set methods, which describe agents' attributes, may update the value returned or stored in each tick count. The agents may continue or update their actions according to the results of the previous action they performed.

Agents are situated in continuous space and/or grid in the simulation environment which provides a context for interaction and communication of agents. Agents may be distributed

to the environment randomly or with some rules. They may have the energy to make them survive. If the agent's energy is exhausted, the agent may die. If the agent's energy reaches the reproduction threshold, it may reproduce. In the simulation environment, there are heterogeneous agents which have different types. For example, an agent may represent the animal, while the other may represent the human. A style class in two-dimensional (2D) or three-dimensional (3D) simulation environment can be created in a way that defines the physical properties of agents such as size, color, and shape. Global parameters associated with agent classes, including initial values of project given by users, may be defined in an xml file.

Repast Simphony provides the users a graphical user interface (GUI). GUI allows the users to manage the simulation processes and to control the parameters. GUI has a user panel that includes run options, parameters, and scenario tree. To form the scenario tree of a project, context builder Java file is defined in data loaders to display agents and the environment on which agents are located. It is possible to observe agents' behavior outputs on the plots and charts. Data sets are created to graphically illustrate time charts defining variables over time. The data set source is determined by pointing out the relevant methods. Histogram bar chart illustrates the distribution of variables.

4. Implementation of the case studies

Agent-based models utilizing Repast Simphony have been developed for a diverse range of scenario including biological systems. In this chapter, three different case studies are presented to better understand ABMS phenomena. These case studies described in subsections are highlighted local behaviors of a real system.

4.1. Sunn pest-wheat

This case study [14] presents the predator prey relationship model in the ecology. In this model, three types of agents are defined as sunn pest, wheat, and parasitoid. The sunn pest called bug agent in the model is both the predator and the prey roles. Wheat called habitat in the model is a cereal plant widely cultivated for food. The sunn pest is fed with wheat grain. The parasitoid is the predator which parasitizes the sunn pest's eggs. We have a grid where the sunn pests are randomly distributed illustrated in **Figure 1**.

The grid includes sunn pest, wheat, and parasitoid. In **Figure 2**, the green color shades indicate the growth of wheat, the red color cells indicate the sunn pest, and the white color cells indicate sunn pests' nymphs. About 7000 sunn pest agents and 1000 parasitoid agents are randomly distributed in the 28,000 grid cells.

In modeling of sunn pest-wheat scenario, agent classes and methods are built according to the definitions in **Tables 1–3**.

The simulation runs during 90th tick counts which is represented in sunn pests' lifecycle (biological stages) and cultivation cycle of wheat. The aim of this case study is to simulate

http://dx.doi.org/10.5772/intechopen.80070

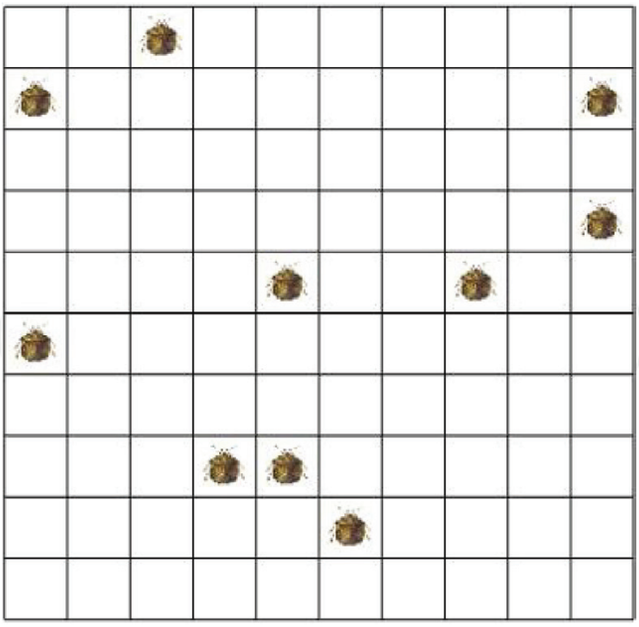

Figure 1. Distribution of the sunn pests on the 10 × 10 grid size.

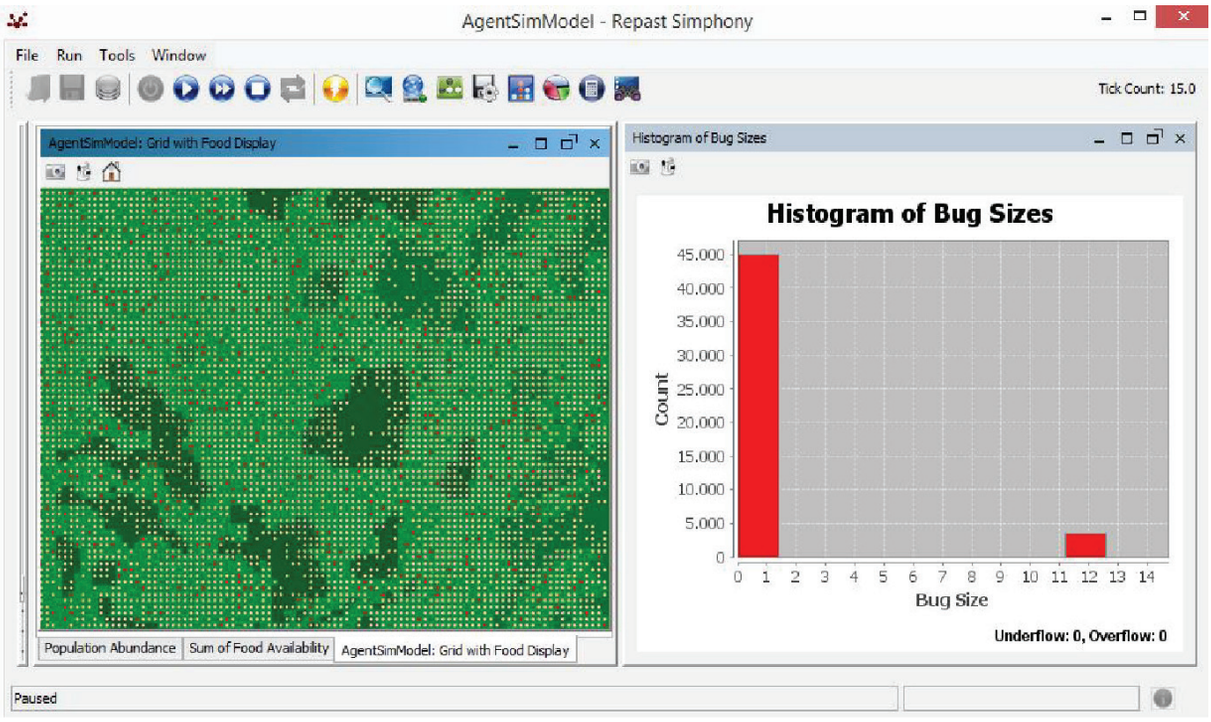

Figure 2. The graphical user interface during the running of the simulation [14].

the chemical and/or biological struggles against sunn pest and obtaining maximum gain to produce the wheat. The parasitoids are used only in biological struggles against sunn pest. In the initial time, all of agents distribute randomly on the grid. If the biological struggle is to be

Roles	Predator, prey
Attributes	Size, energy, gender, survival probability, state, generation
Actions	Move, grow, mortality, reproduce, die
Rules	Female and male ratio is 50%.
	Randomly goes to one of the neighbour cells around him and feeds and grows from that cell.
	When the adults come to the field, the simulation starts.
	If the sunn pest is female and its size is more than 12 mm (i.e. the biological state is mature), lay eggs 5 times, leaves a total of 80 to 150 eggs and dies.
	If the sunn pest is a male, it dies in condition that the probability of survival (95%) being smaller than a random number determined.
	Grows 0.3 mm per step in the embryo phase and except this; it grows as much as the amount that it eats.
	If the size of sunn pest is in the range of 0 - 0.8, its biological stage is an "Embryo".
	If the size of sunn pest is in the range of 0.8 - 2.0, its biological stage is a "First nymph".
	If the size of sunn pest is in the range of 2.0 - 3.5, its biological stage is an "Second nymph".
	If the size of sunn pest is in the range of 3.5 - 5.0, its biological stage is a "Third nymph".
	If the size of sunn pest is in the range of 5.0 - 6.0, its biological stage is a "Fourth nymph".
	If the size of sunn pest is in the range of 6.0 - 6.0, its biological stage is a "Fifth nymph".
	If the size of sunn pest is greater than 10.0 mm, its biological stage is in the "Adult".

Table 1. Local knowledge of sunn pest (bug agent).

done, the parasitoid agents are activated. Until the 15th tick count, sunn pest agents act on the grid and fed from the habitat cells. At the 15th tick count, female sunn pests lay eggs (embryos) and die. In **Figure 2**, white color cells on the grid indicate the embryos, and the histogram bar shows the sunn pests' total numbers for each biological stage. Through 90th tick counts, the sunn pests complete their lifecycle against the parasitoid. At the end of the simulation, the sum of food availability on the habitat cells has been observed illustrated in **Figure 3**.

In the result of this case study, the relationship between sunn pest and parasitoid is simulated with the agent-based modeling approach. This case study represents the behavior of a real biological system, even if it is not identified with all the details. In the computer simulation studies, some assumptions can be done, such as in this study, the climate conditions are not included in the simulation. The boundaries of the study must be specified, otherwise undesirable results can be obtained and the system drifts the chaos.

4.2. Bacteria – antibiotic

This work [15] presents bacterial population and resistance to antibiotics. Bacterial population known as bacterial flora are nonharmful microorganisms in the human body that live in the human skin, in the mouth, in the digestive system, etc. There are immune cells that suppress the bacterial flora. Immune cells and bacterial flora should always be balanced in the body. In this model, two types of agents are defined as bacteria and immune system cell. About 4000

Roles	Food value layer
Attributes	production rate, availability value
Actions	grow
Rules	Defined at certain ratio within each cell in the grid.
	Grows certain ratio in each step, and it becomes availability value.
	The sunn pest consumes food as much as its growth rate in the cell where it is located.
	Its color scale changes according to its production rate.

Table 2. Local knowledge of wheat (habitat cell).

Roles	Parasitoid
Attributes	—
Actions	move, hunt, kill
Rules	Randomly distributed in the grid.
	Parasitizes sunn pest's embriyo at the neighbouring cells around and locates in its cell.
	If there is no sunn pest's embriyo in the neighbouring cells, it changes its position and randomly moves to another cell.
	Remains in the grid until the end of the simulation.

Table 3. Local knowledge of parasitoid.

bacterial agents and 100 immune system cell agents are randomly distributed on the 100 × 100 grid cells. The grid represents a human tissue or organ.

Bacteria agents are grouped within themselves depending on the range of disease called virulence factor. The virulence factor is assigned between 1 and 4. In **Figure 4**, the white cells on the grid indicate the bacterial agents which have the virulence factor of 1, the yellow cells on the grid indicate the bacterial agents which have the virulence factor of 2, the red cells on the grid indicate the bacterial agents which have the virulence factor of 3, the purple cells on the grid indicate the bacterial agents which have the virulence factor of 4, and the blue cells on the grid indicate the immune system cell agents which have the virulence factor of 4.

The simulation has three parts: the first one is bacterial competition on flora, the second is antibiotic usage, and the third is antibiotic resistance. According to these parts, the local knowledge of bacterial agents and immune system cell agents is defined in **Table 4**.

In the first part of the simulation, the aim is to balance the population of bacterial agents and immune system cell agents on the grid. Also, bacterial agents compete with their neighbors for space and food resources. A food layer is defined in the simulation environment to live, grow, and reproduce. In the second part of the simulation, antibiotic usage is defined against the bacterial agents. An antibiotic layer is included in the simulation environment. The aim is to help the immune system cell agents and kill the bacterial agents.

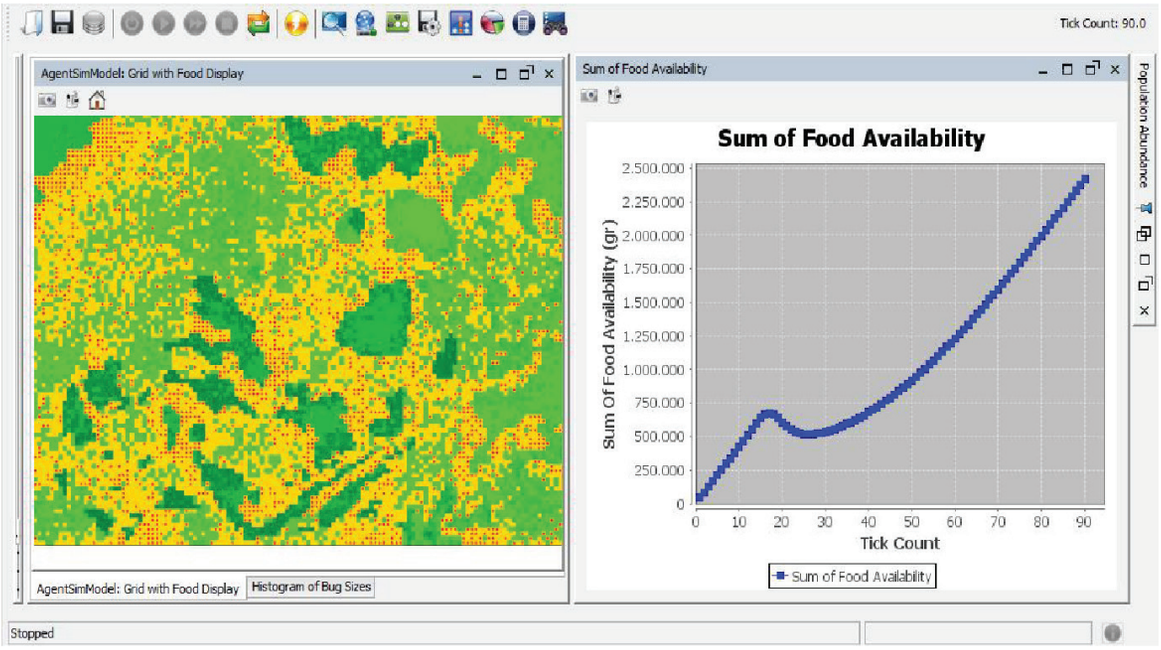

Figure 3. The graphical user interface at the end of simulation [14].

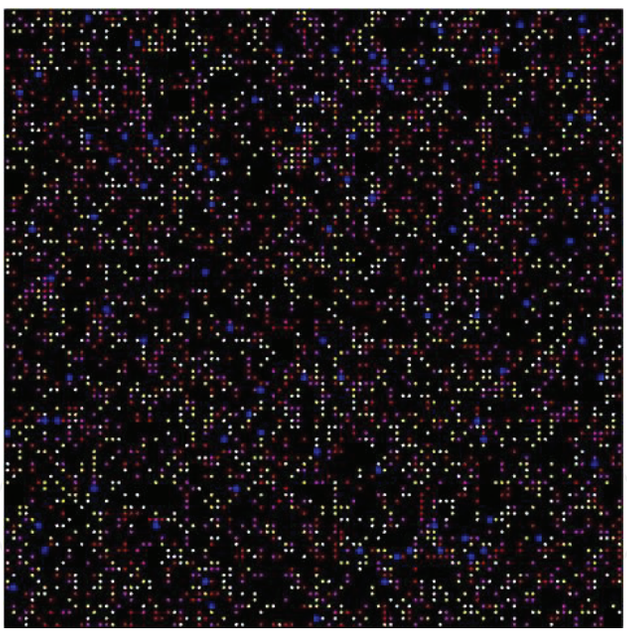

Figure 4. Bacterial agents and immune system cell agents on the grid at the initial time [15].

Figure 5 shows how the antibiotic usage suppresses the bacterial agent population when the immune system cell agents are insufficient. The third part of the simulation presents the relationship between antibiotic-resistant bacterial agents and immune system cell agents. Most of the bacterial agents with virulence factor between 2 and 4 are killed by the antibiotic, whereas rest is killed by immune system cell agents. However, bacterial agents with virulence factor of 1 survive because they are antibiotic-resistant. Bacterial agents with low virulence factor are

	Bacterial agents	Immune system cell agents
Roles	microorganism	microorganism
Attributes	virulance factor, survivalProbability, mutationProbability, size	Id
Actions	move, grow, reproduce	move, send signal, kill, disappear
Rules	Randomly distributed in the grid.	Observes the neighbour 48 cells.
	Divided up into empty cells during the simulation runtime.	If there is a bacteria agent in the neighbour cells, it kills and locates on that cell.
	With a low virulence factor reproduce very rapidly.	If it kills two bacteria, it sends signals to another immune system cell agent and dies.
	With the virulence factor of 1 is resistant to antibiotic which has a concentration value that can kill bacteria in each cell.	If there are more than 40 bacteria agents in the neighbour cells, it sends signals to other immune system cell agents.
		If there are between 2 and 15 bacteria agents in the neighbour cells, it does not see any danger state and disappear.

Table 4. The local knowledge of bacterial agents and immune system cell agents.

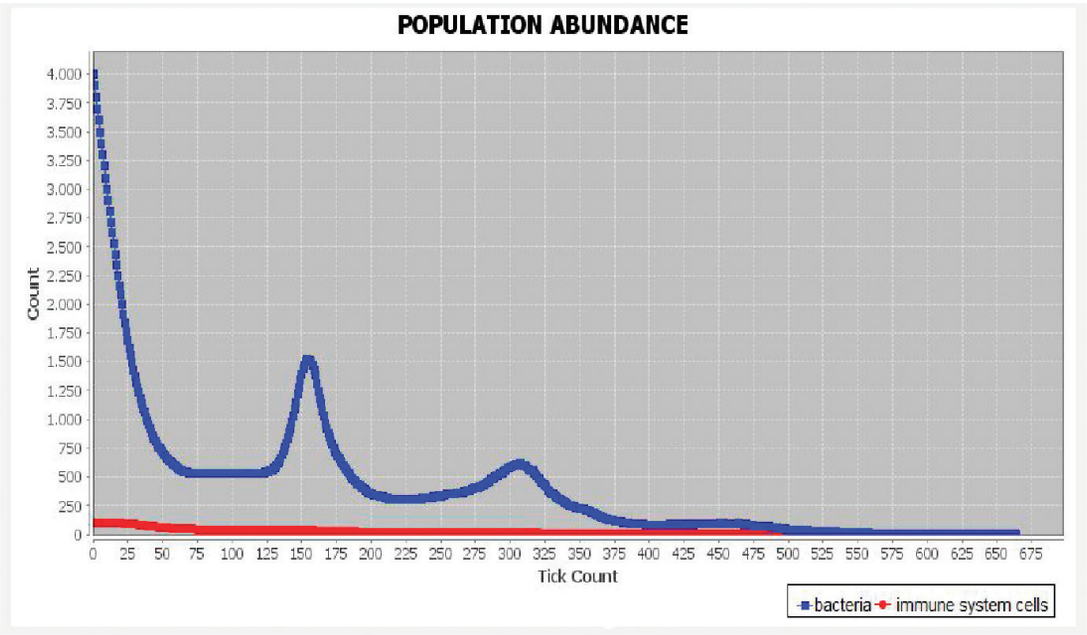

Figure 5. Relationship between bacterial agents and immune system cell agents in the antibiotic usage [15].

divided very rapidly so that the number of immune system cell agents is increased. **Figure 6** shows the struggle between immune system cell agents and bacterial agents on the grid. The grid, which indicates green color in **Figure 6**, represents tissue/organ.

Figure 7 shows, graphically, the populations of immune system cell agents and bacterial agents during the simulation runtime. At the initial time, there are 4000 bacterial agents and

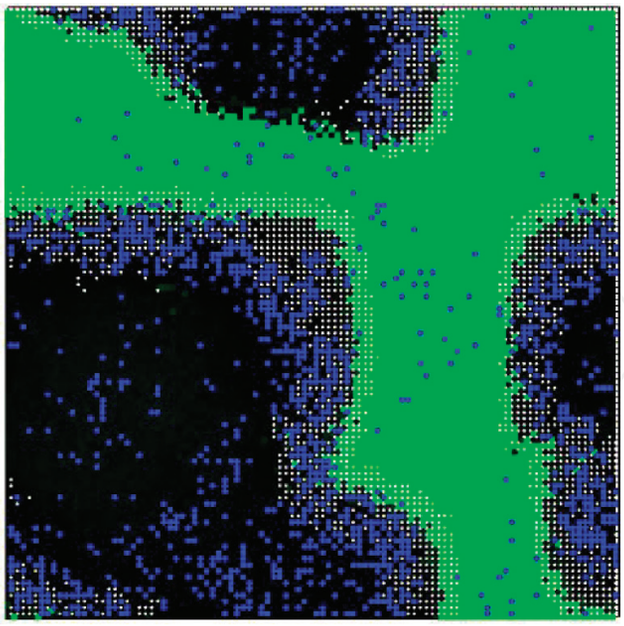

Figure 6. Struggle between immune system cell agents and bacterial agents [15].

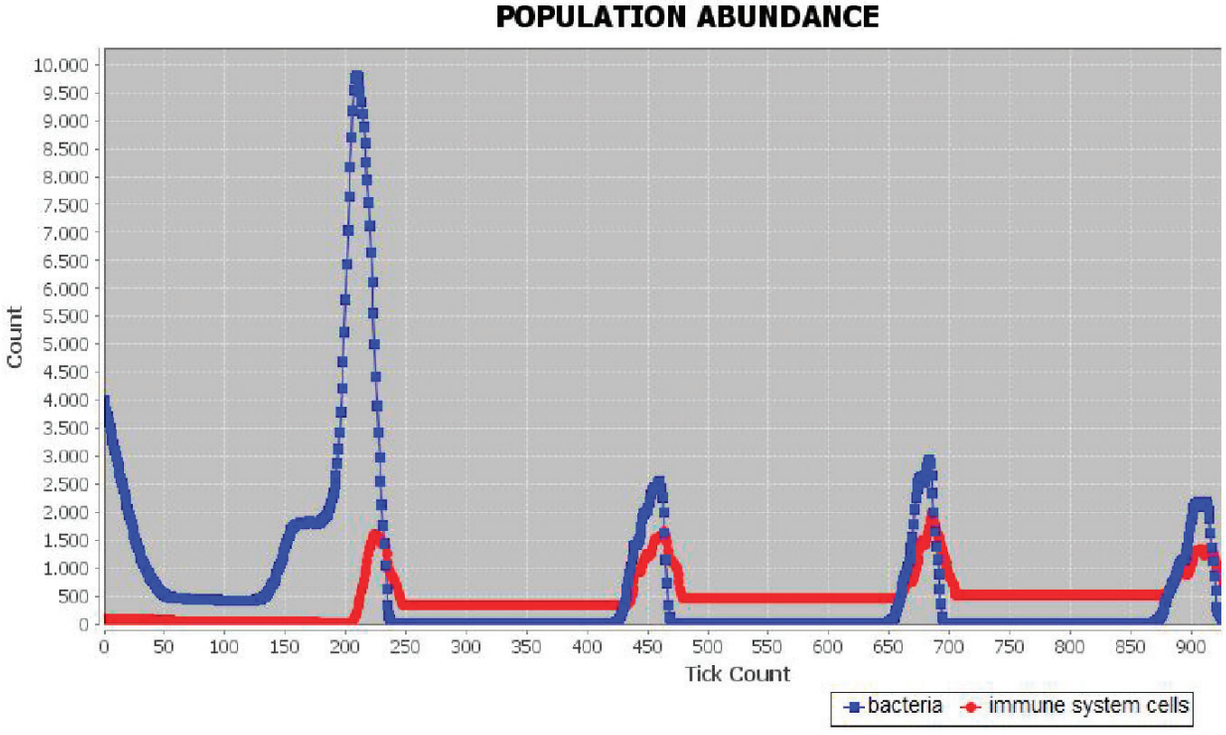

Figure 7. Relationship between bacterial agents and immune system cell agents in the antibiotic usage [15].

100 immune system cell agents on the grid. When simulation starts, immune system cell agents kill some of the bacterial agents. Surviving bacterial agents grow and reproduce. When the population of bacterial agents reaches the maximum value, the population of immune

system cell agents starts to increase. Antibiotic usage helps to reduce the population of bacterial agents because the population of bacterial agents is very large.

This case study provides an introduction to understand the dynamics of microbiological systems that take place in the process of bacterial evolution. During the simulation runtime, it is observed that how the system dynamics can be adaptive to external influences and effect interactions of them.

4.3. Homeostasis

Homeostasis is a steady state that regulates the keeping of state variables at a constant or stable condition. Homeostasis is defined as a closed-loop control system that balances changes of target values. Biological systems like human body struggle to control its internal environment against internal and external influences. If homeostasis is unsuccessful in the body, vital functions cannot continue to work and the system drifts into chaos. Almost all homeostatic control mechanisms involve negative feedback loop which provides long-term control to maintain a steady state. Negative feedback has a self-regulating mechanism for maintaining homeostasis. Negative feedback mechanism involves some important factors. The first one is sensor or receptor which senses changes in the system variables that need to be regulated. The second is a control center which has a set point or threshold value that keeps the optimal value of the system variable. The other is an effector which produces a response that eliminates or reduces the changes of the system variables. Negative feedback loop runs until the system variables are adjusted at optimum values.

There are many negative feedback control mechanisms in the biological systems. The human physiology is one of the best examples of the biological systems in which the negative feedback mechanisms are observed. Some negative feedback control mechanisms that occur in the human body include regulation of blood pressure, keeping the pH constant, regulation of oxygen and carbon dioxide concentration in the blood, hormonal regulation of blood glucose levels, thyroid regulation, the control of body temperature, etc.

In this chapter, an example of negative feedback control mechanism that occurs in the human body is presented with ABMS approach.

4.3.1. The control of the temperature: thermoregulation

In this case study [16], an agent-based homeostatic control model that regulates the body temperature during fever is presented. Fever is defined by an increase in body temperature above the normal range. Three types of agents, receptor agent, controller agent, and effector agents, are defined. Receptor agent is represented by thermoreceptor agent which senses changes in the body temperature. Controller agent has a set point which keeps the optimal value of body temperature. Effector agents are a set of dynamic autonomous agents which represent blood vessel that is a component of cardiovascular system [17]. Effector agents have been developed with ABMS approach [8]. The blood vessel is divided into segments. Each segment represents an agent. All of the agents in the negative feedback control mechanism are illustrated in **Figure 8**.

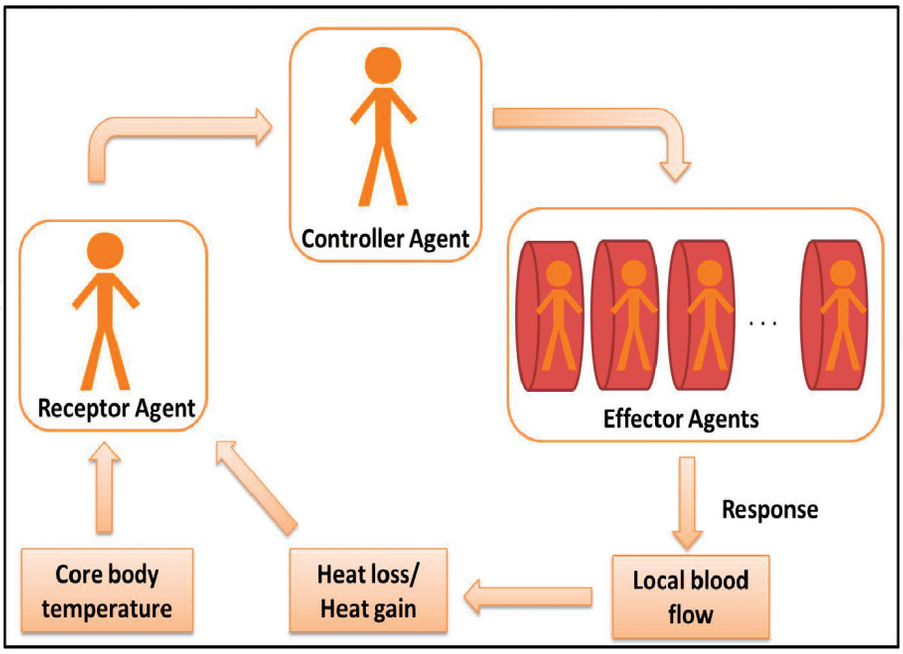

Figure 8. Negative feedback control mechanism of thermoregulation [16].

During the simulation runtime, each agent defined in the negative feedback control mechanism interacts with other agents. All of the agents interact with each other by using Java message service. Java message service supports "publish/subscribe" message delivery model. Receptor agent monitors the change of the body temperature and publishes it to the controller agent who has subscribed to the value. Controller agent receives message that includes the body temperature value. Controller agent compares the body temperature value to its set point value. Controller agent sends a message to the effector agents which start or stop the negative feedback mechanism. Effector agents produce a response based on the message that they receive, and they publish to the receptor agent to correct the deviation with negative feedback. Negative feedback control mechanism achieves a balance between heat production and heat loss. The output of the negative feedback mechanism is illustrated in **Figure 9**.

The simulation has a scenario of fever disease. This scenario achieves a balance with the agent-based negative feedback control mechanism as follows:

1. At the initial time, the core body temperature fluctuates between 36.7 and 37.2°C which is an acceptable normal range inside the human body. The set point of the body temperature is set to 37°C.

2. An infection that causes a fever disease is assumed that it starts with increasing the body temperature. The set point of the body temperature is set to 40°C at which the maximum value is assumed by homeostasis. The body temperature is less than the new set point of the body temperature.

3. Increased body temperature triggers shiver which is the reaction of the body. Shiver tries to gain the body heat which causes the constriction of the blood vessel called vasoconstriction. The controller agent publishes message "VASOCONSTRICTION" to the effector

http://dx.doi.org/10.5772/intechopen.80070

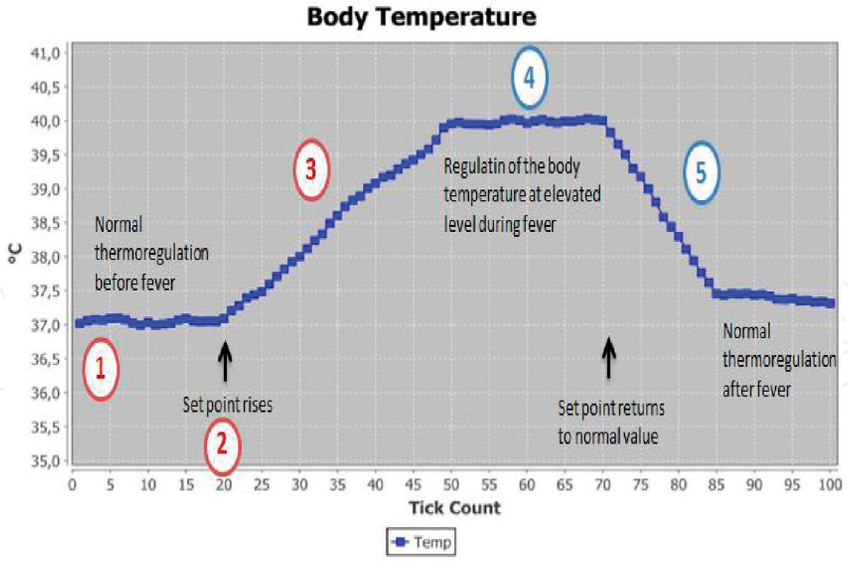

Figure 9. Regulation of the body temperature [16].

agents. The effector agents decrease their radius values to produce a response. Local blood flow parameter depending on the radius helps the heat gain.

4. The body temperature reaches the new set point of the body temperature. The infection is assumed to be cleared inside the body. The set point of the body temperature is set to 37°C. The body temperature is more than the new set point of the body temperature.

5. Condition at fourth step triggers sweat. Sweat tries to reduce the body heat which causes the dilation of the blood vessel called vasodilation. The controller agent publishes message "VASODILATION" to the effector agents. The effector agents increase their radius values to produce a response. Local blood flow parameter depending on the radius helps the heat loss. Thus, the body temperature returns to the optimal value.

In the result of this case study, it is observed graphically how the body temperature is regulated during fever. Agent-based negative feedback control mechanism can be called adaptation loop [18]. This is because the negative feedback control mechanism is run by a set of dynamic autonomous agents. In this mechanism, it is possible to observe their local behaviors.

5. Conclusion

This chapter has introduced the reader to ABMS, and it described implementations of different case studies utilizing the Repast Simphony toolkit. ABMS offers an extensible way to model biological systems consisting of autonomous and interacting agents which perform their actions and adapt their behaviors. Computer simulation helps the researcher to explore the behavior of a dynamic system. This chapter is concluded by observing interactions of real systems' components in the abstraction level.

Author details

Şebnem Bora[1] and Sevcan Emek[2]*

*Address all correspondence to: sevcan.emek@cbu.edu.tr

1 Ege University, Izmir, Turkey

2 Manisa Celal Bayar University, Manisa, Turkey

References

[1] Di Marzo Serugendo G, Gleizes M-P, Karageorgos A, editors. Self -Organising Software From Natural to Artificial Adaptation. London New York: Springer Heidelberg Dordrecht; 2011. 462 p. DOI: 10.1007/978-3-642 -17348-6

[2] DeAngelis DL, Grimm V. Individual-based models in ecology after four decades. F1000Prime Reports. 2014;**6**:39. DOI: 10. 12703/P6-39

[3] Borshchev A, Filippov A. From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools. In: The 22nd International Conference of the System Dynamics Society (SDS'04); 25-29 July 2004; Oxford, England

[4] Macal CM, North MJ. Agent-based modeling and simulation. In: IEEE Proceedings of the 2009 Winter Simulation Conference (WSC); Austin, TX; December 2009. pp. 86-98

[5] Kluegl F, Bazzan ALC. Agent-based modeling and simulation. The AI Magazine. 2012;**33**:29-40

[6] Bonabeau E. Agent-based modeling: Methods and techniques or simulating human systems. Proceedings of the National Academy of Sciences. 2002;**99**:7280-7287. DOI: 10.1073/pnas.08080899

[7] Bandini S, Manzoni ST, Vizzari G. Agent based modeling and simulation: An informatics perspective. Journal of Artificial Societies and Social Simulation. 2009;**12**(4):4

[8] Bora Ş, Evren V, Emek S, Çakırlar I. Agent-based modeling and simulation of blood vessels in the cardiovascular system. SIMULATION. Special Issue, 2017. DOI: 10.1177/0037549717712602

[9] Macal CM, North MJ. Tutorial on agent-based modelling and simulation. Journal of Simulation. 2010;**4**:151-162. DOI: 10.1057/jos.2010.3

[10] Nianogo RA, Arah OA. Agent-based modeling of noncommunicable diseases: A systematic review. American Journal of Public Health. 2015;**105**(3):20-31

[11] Nikolai C, Madey G. Tools of the trade: A survey of various agent based modeling platforms. Journal of Artificial Societies and Social Simulation. 2009;**12**(2):2

[12] Crooks AT, Heppenstall AJ. Introduction to agent-based modelling. In: Heppenstall AJ, Crooks AT, See LM, Batty M, editors. Agent-Based Models of Geographical Systems. Dordrecht: Springer; 2014. pp. 85-105

[13] Ozik J, Collier NT, Murphy JT, North MJ. The relogo agent-based modeling language. In: Proceedings of the 2013 Winter Simulation Conference (WSC'13); 8-11 December 2013. pp. 1560-1568. DOI: 10.1109/WSC.2013.6721539

[14] Bora CB, Emek S, Kose H. Agent-based modeling and simulation of the sunn pest-what relation and of the struggle against sunn pest in Turkey. In: Proceeding of the International Conference on Engineering Technology and Innovation (ICETI'17); 22-26 March 2017. pp. 340-347

[15] Bora CB, Emek S, Evren V, Bora Ş. Modeling and simulation of the resistance of bacteria to antibiotics. Periodicals of Engineering and Natural Sciences. 2017;**5**(3):396-408. DOI: 10.21533/pen

[16] Bora Ş, Emek S, Evren V. An agent-based approach in homeostatic control systems: Thermoregulation. In: IEEE 9th International Conference on Computational Intelligence and Communication Networks; 16-17 September 2017. pp. 113-116. DOI: 10.1109/CICN.2017.8319367

[17] Guyton AC, Hall JE. Textbook of Medical Physiology. 11th ed. Amsterdam: Elseiver Saunders; 2006

[18] Salehie M, Tahvildari L. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems. 2009;**4**:1-14. DOI: 10.1145/1516533.1516538

IntechOpen