Revista Eletrônica de Iniciação Científica em Computação, 2025, 23:1
ISSN: 3085-8461 • doi: 10.5753/reic.2025.5295

Research Paper

A Structured Report on Agent-Based Simulation Development with the GAMA Platform

Aline Rodrigues Santos [Universidade do Estado de Santa Catarina | *aline.rs@edu.udesc.br*]
Fernando Santos [Universidade do Estado de Santa Catarina | *fernando.santos@udesc.br*]

Departamento de Engenharia de Software, Universidade do Estado de Santa Catarina, Rua Dr. Getúlio Vargas, 2822, Bela Vista, Ibirama, SC, 89140-000, Brasil.

Abstract. The use of agent-based simulations is becoming common and has been used to abstract complex concepts through visual demonstrations. This has driven the emergence of platforms for developing these simulations. In this context, GAMA stands out as an attractive option because of its wide range of features. However, GAMA still lacks materials to guide beginner developers. In order to fill this gap, this paper presents a structured report on agent-based simulation development with GAMA. The paper describes the main functionalities and the structure for developing a simulation with GAMA. In addition to that, the paper exemplifies these elements through the development of the Sugarscape simulation, known in the community. The combination of the structured report and the Sugarscape implementation provides a introductory guide to developing agent-based simulation in GAMA. Finally, the paper presents the challenges that a beginner developer may encounter and recommendations to address them.

Keywords: Agent-based simulation, GAMA, Sugarscape.

Received: 06 December 2024 • **Accepted:** 28 March 2025 • **Published:** 15 April 2025

1 Introduction

Agent-based simulations (ABS) have been used to replicate and study different phenomena. According to Wilensky and Rand [2015], the adoption of ABS not only helps in visualizing complex scenarios but also in simplifying and abstracting them. An example of this approach is illustrated by Costa [2023], who developed an ABS to demonstrate some of the main components of the complex Tupinambá society, allowing the explanation of phenomena and demonstrating a tribal society that lived in Brazilian territory. Another example is the simulation of Covid-19 spread by Teixeira and Santos [2020], which allowed to study the effects of isolation measures on the pandemic. ABSs were also used in the traffic domain, i.e., to study adaptive traffic signal control [Santos *et al.*, 2017].

Kleiboer [1997] describes the different functions of ABS, two of which are research and teaching support. This highlights the need for learning concepts and programming languages for ABS development. Although it is possible to develop ABS using general-purpose programming languages like Python or Java, the use of a specialized platform significantly simplifies learning and makes it more intuitive. For this reason, there are platforms that facilitate the development of agent-based simulations, such as NetLogo Wilensky [1999] and, especially, GAMA [Taillandier *et al.*, 2019].

The GAMA platform offers a programming language called GAML (Gama Modeling Language). Through GMAL it is possible to develop simple or complex simulations, such as 3D models that provide control over various visual aspects such as lighting, texture, and camera position or just 2D simple models. Additionally, the platform prioritizes the development of intuitive interfaces and allows the provision of different visualizations that can be customized and updated dynamically to observe the simulation progress. GAMA also allows specifying monitors to collect and display data about the simulation during its execution. The presence of these features illustrates how the GAMA platform can streamline the development of ABS [Taillandier *et al.*, 2019].

Despite offering various elements, a beginner in ABS interested in developing simulations on this platform may face challenges. The available documentation[1] for GAMA is extensive; however, the explanation of the essential elements for developing an ABS is dispersed across different topics, which poses comprehension challenges, especially for beginners. The learning curve can be steep for those who do not have prior experience with ABS platforms, which requires additional time and effort to familiarize themselves with the necessary elements for implementation. Therefore, there is a lack of an introductory guide that presents the essential elements for implementing ABS in GAMA using a simple and widely known simulation. Despite the fact that the GAMA documentation offers a tutorial on implementing the *Predator-Prey* simulation, it provides few explanations about the platform, its organization, the reasons for its use, and the functionality of each element used.

This paper investigates ABS development on the GAMA platform by means of a qualitative research [Creswell and Creswell, 2018]. The approach was based on a case study, whose objective is to explore ABS development in GAMA. The research question is: *what are the essential elements for the implementation of ABS in GAMA?* To answer that question, we focused on the Sugarscape simulation, widely known in the ABS community. The Sugarscape simulation was implemented in GAMA and made available online[2]. The essential elements revealed by the case study are described in this paper

[1] `https://gama-platform.org/wiki/Home`
[2] `https://github.com/agentbasedsimulations/sugarscape-gama-simulation`

through a structured report, as suggested by Wohlin *et al.* [2024]. The structured report and the implementation, when combined, provide an introductory guide to developing ABSs in GAMA. The paper also provides recommendations for addressing some limitations of GAMA perceived during the simulation development.

The remaining of this paper is organized as follows. Section 2 presents the required background on ABS, the GAMA platform, and the *Sugarscape* simulation. In section 3 we present the structured report on ABS development in GAMA through the development of the *Sugarscape* simulation. Finally, section 4 presents concluding remarks and future work.

2 Background

This section introduces concepts used in this paper. Firstly, it presents what ABS are. Then, the GAMA platform is explained, along with some of its various features. Later, the Sugarscape simulation is described, highlighting the main parts of its development.

2.1 Agent-based simulation

An agent-based simulation (ABS) is a practical approach to studying complex systems [Klügl and Bazzan, 2012]. This approach allows researchers to predict and to optimize these systems more effectively. Essentially, the simulation offers a tangible way to represent and explore complex situations, where agents play crucial roles, such as individuals, animals, or entities, interacting with each other and with a predefined environment.

According to Klügl and Bazzan [2012], an ABS is composed of three elements: a set of autonomous *agents*, endowed with rules governing their behavior; the specification of *interactions* between agents and the environment, responsible for producing the overall output of the system; and the simulated *environment*, which contains all other simulation elements, such as resources and other objects without active behavior. The result of an ABS can be observed through visualization of agents behavior or through graphs presenting data collected during simulation execution.

ABS allows researchers to study a particular phenomenon at both micro and macroscopic levels. At the microscopic level, different agent traits and decision-making strategies can be considered. Therefore, heterogeneity is easily incorporated in the simulation. That way, the ABS paradigm offers researchers the ability to make specific adjustments to agent behavior so that they can respond adaptively to the environment and other agents, influencing the observed results in the system under study. An example of this is the ABS developed by Pereira *et al.* [2011] to study human behavior in natural disaster situations and how to improve the performance of rescue teams. At the macroscopic level, the actions and interactions of agents can lead to emergent, often unpredicted, phenomena [Klügl, 2008]. Thus, the ABS paradigm is useful to provide insights at a global level, where the aim is to observe the behavior of groups or even a population [Eisinger and Thulke, 2008].

Additionally, ABSs are highly scalable and modular, which means that models can be easily adapted and extended to include new elements or details [Macal, 2016]. This makes this approach especially useful for simulating dynamic and evolving systems, such as ecosystems, financial markets, or social networks.

2.2 GAMA platform

Agent-Based Modeling (ABM) platforms, like Cormas and GAMA, are specialized tools for creating agent-based simulations (ABSs), particularly focusing on explicit spatial representation. Cormas, developed in 1998, stands as one of the pioneering ABM platforms [Bousquet *et al.*, 1998]. Similarly, GAMA, built in Java, is a prominent open-source ABM platform. These platforms aim to offer a scientific approach for modeling a wide range of scenarios, making them accessible to both scientists and non-scientists alike [Taillandier *et al.*, 2019].

GAMA provide a programming language called Gama Modeling Language (GAML). Through GAML, it is possible to develop simple or complex simulations, including 2D and 3D models that provide control over aspects such as lighting, texture, and cameras. Additionally, the platform prioritizes the development of intuitive interfaces, enabling the provision of various customizable displays for the same model, dynamically updated to visualize the simulation [Taillandier *et al.*, 2019].

ABSs for studying various phenomena have been recently developed in GAMA. Gaudou *et al.* [2020] present the COMOKIT, an ABS developed in GAMA to analyze and compare interventions to deal with the Covid-19 epidemic at the scale of a city. Daudé *et al.* [2019] present the ESCAPE ABS, to study mass evacuation strategies in crisis situations. These simulations highlight the potential of the GAMA platform.

2.3 Sugarscape

The Sugarscape is a model of an artificial society proposed by Epstein and Axtell [1996]. It is widely used in the ABS community as an example of a simulation that exhibits emergent phenomena. In essence, the Sugarscape model simulates a population that depends on limited resources existing in the environment. The population consists of *ants* searching for food (sugar) present in the environment, which is composed of different regions, some rich and others poor in sugar. Each ant in the simulation is represented by an artificial agent with distinct attributes, such as vision, energy, and metabolism. These agents have the ability to move in the environment and collect sugar. The objective of each ant is to identify nearby regions with a higher quantity of sugar and that are free (without another ant). When a region is identified, the ant moves there. When an ant interacts with a sugar region, it consumes the resource, and this sugar may or may not have an immediate growth. The variation in resources allows for exploring population dynamics and their concentration in different parts of the environment.

In the model created by Epstein and Axtell [1996], the environment consists of a set of 2500 cells, representing the regions, organized in a grid of dimensions 50 x 50. Each cell has a certain amount of sugar and a maximum storage capacity. The initial amount of sugar in each cell is defined to form two sugar peaks, in the northeast and southeast of the grid. An external file provides the initial amount of sugar

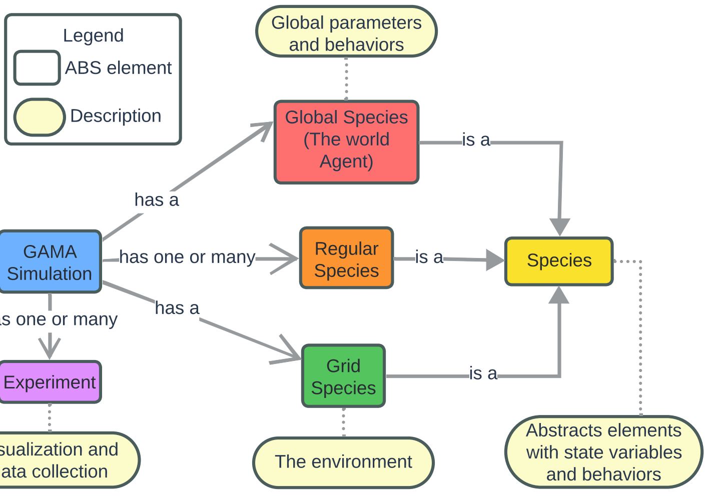


Figure 1. Overview of the essential elements of an ABS in GAMA

for each cell. As the simulation progresses, the sugar consumed by the agents is replenished. Two replenishment rules are considered: *immediate replenishment*, where the amount of sugar consumed by an agent is fully replenished at each simulation step, and *partial replenishment*, where only a percentage of the initial amount of sugar is replenished at each simulation step.

Although simple, the Sugarscape simulation highlights, for example, the emergent ecological phenomenon of carrying capacity for a species—according to which a given environment can only support a finite population [Epstein and Axtell, 1996]. This ABS was chosen in this work due to its simplicity and popularity. The NetLogo platform, for example, provides well-documented implementation of Sugarscape with the two mentioned sugar replenishment rules [Li and Wilensky, 2009a,b].

3 Development of the Sugarscape ABS in GAMA

This section presents a structured report on the development of ABS with GAMA. To investigate the GAMA platform and identify the essential elements for ABS implementation, we conducted a case study in which the Sugarscape simulation was developed. In the following we report these essential elements.

Figure 1 presents an overview of the essential elements required to develop an ABS in GAMA, in particular those ABSs similar to the Sugarscape model. A GAMA simulation is often composed of a global species (which plays the role of the world agent, or the simulation controller), one or many regular species, a grid species (to represent the environment where agents are situated) and one or many experiments to collect data and to visualize the simulation output. The diagram was abstracted from both GAMA's documentation[3] and simulations available in the GAMA platform. In the following sections we detail these simulation elements.

It is worth noting that the diagram in Figure 1 and those presented in the following sections are contributions of this paper. Their simplified content, in comparison to the GAMA documentation, intends to produce a straightforward view of an ABS structure, making it easier to understand. The GAMA documentation provides a learning tutorial that only describes the general structure of the GAML language metamodel, not offering a concrete view of the elements required in the simulation and their components. GAML is still under development, which may explain its documentation. This can create difficulties for beginner developers to understand the structure of ABSs on the platform.

In the following sections we describe the essential elements of ABSs in GAMA and how they were implemented in Sugarscape. Examples included in the following diagrams were either created by the authors or adapted from examples available in the platform's documentation. The goal of these diagrams is to serve as a guide to the foundational structure of ABSs in GAMA, bridging the gap between theoretical concepts and practical application. The complete ABS implementation is provided online.[2]

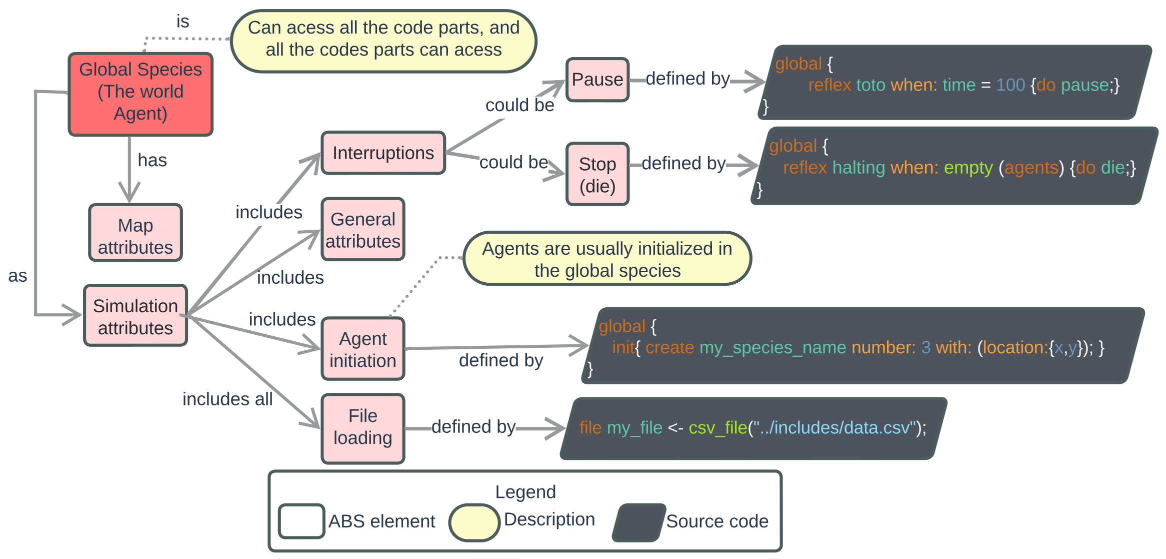
3.1 Global Species

The global species plays a fundamental role in the simulation, being the main entity that defines the essential characteristics of the environment and agents. Taillandier *et al.* [2019] define it as a *master species*. Figure 2 shows the GAMA essential elements related to the global species. Only one instance of the global species can exist, and it is where characteristics such as values and methods accessible to all other species and the experiment are defined.

The global species incorporates attributes that define important aspects of the simulation, such as the size of the map, as well as fundamental attributes like simulation interruptions, which can pause or terminate the simulation. These simulation attributes can be accessed and modified by other species and the experiment, acting as simulation parameters that provide a flexible basis for configuring the simulation (e.g., with user-defined data) Taillandier *et al.* [2019].

In Sugarscape, the global species plays a fundamental role in defining key parameters that shape the simulation. These parameters, specified as simulation attributes, are implemented in the global section of the source code. Listing 1 presents the source code of the global species from the the Sugarscape simulation. As we can see, the Sugarscape simulation specifies parameters to constraint the

[3]`https://gama-platform.org/wiki/LearnGAMLStepByStep`



```
global {
    reflex toto when: time = 100 {do pause;}
}
```

```
global {
  reflex halting when: empty (agents) {do die;}
}
```

```
global {
  init{ create my_species_name number: 3 with: (location:{x,y}); }
}
```

```
file my_file <- csv_file("../includes/data.csv");
```

Figure 2. GAMA elements related to the global species

maximum vision range of agents (vision), the initial number of agents (nb_initial_ant), and the metabolism rate of agents (metabolism). These values set the limits for the behavior of agents in the simulation. An additional parameter, the maximum value of energy (max_energy), ensures compatibility with graphical elements such as histograms. The number of live ants is tracked in nb_ant, which is updated dynamically based on the current state of the simulation.

```
 1  global {
 2       // Customize the environmnent extent
 3     geometry shape <− rectangle(50#cm,50#cm);
 4     // Maximum values for some ant attributes
 5       int vision <− 6;
 6     int nb_initial_ant <− 400;
 7     int metabolism <−4;
 8       // Maximum energy value to make the histogram graph work
 9     float max_energy <− 200.0;
10       // Counts the number of live ants
11     int nb_ant −> {length(ant)};
12
13     // Calculates the averages of the specific characteristics of living agents
14     float average_vision <− 6.0 update : calculate_average_vision();
15     float average_metabolism <− 5.0 update : calculate_average_metabolism();
16
17     float calculate_average_vision {
18         float totalVision <− 0.0;
19         ask ant {
20             totalVision <− totalVision + vision_ant;
21         }
22         return totalVision / nb_ant;
23     }
24
25     float calculate_average_metabolism {
26         float totalmetabolism <− 0.0;
27         ask ant {
28             totalmetabolism <− totalmetabolism + metabolism_ant;
29         }
30         return totalmetabolism / nb_ant;
31     }
32
33       // Open and configure a CSV file
34       csv_file arquivo <− csv_file("../includes/map.csv");
35
36     init {
37         matrix data <− matrix(arquivo);
38         ask cell {
39             grid_value <− float(data[grid_x, grid_y]);
40             sugar <− grid_value;
41             max_sugar <− grid_value;
42         }
43         create ant number: nb_initial_ant;
44     }
45  }
```

Listing 1: Global Species source code

Whenever the required data aggregate values from various agents, its gathering must be implemented in the global species, which has access to all simulation elements. This is done by implementing functions to perform calculations and collect data from the population of agents. In the case of the Sugarscape simulation, we collect the average vision and metabolism of agents, as implemented by Li and Wilensky [2009a]. Listing 1 lines 17–31 presents the calculate_average_vision and calculate_average_metabolism functions, which iterate through all living ants to compute the averages. These functions are used to update global attributes average_vision and average_metabolism, as shown in Listing 1 lines 14–15. That way it is possible to analyze the overall trends and monitor changes in the population's behavior over time.

In GAMA, the simulation environment is a cartesian space that adopts a coordinate system to locate elements situated on it and may have its extent customized. GAMA allows defining a particular environment topology to ease the location of elements. Two topologies are supported: grid and graph. The grid topology resembles a matrix formed by cells, where it is possible to define specific values for particular regions of the environment (details of the grid topology are detailed later in section 3.3). The graph topology organizes the environment elements as nodes and edges. The attributes of cells, nodes, or edges can be defined manually or initialized from files. When the simulation depends on external files to initialize the environment, it is in the global species that such files should be loaded. In the case of Sugarscape, we chose to initialize the environment from a CSV file, which provides the initial amount of sugar and the maximum storage capacity for each cell.[4]. Listing 1 shows the source code that loads the CSV file (map.csv, line 34) and initialize the simulation environment with sugar levels to each cell based on data read from the file (lines 37–42). The initialization process implemented in the global species ensures the simulation is set up before agents start to interact.

[4]The CSV file used was adapted from Li and Wilensky [2009a]

3.2 Regular Species

The regular species within the GAMA framework are considered a complement to the master species, each composed of specific characteristics and values that serve as fundamental parameters for the creation and manipulation of the necessary instances during the simulation. Analogous to object-oriented languages, we can understand regular species as classes that define a set of properties and behaviors to be shared by their agents Taillandier *et al.* [2019]. This organization is essential in GAMA to define and structure the entities that compose the simulation, providing them with identity and functionality. Figure 3 presents the GAMA essential elements related to the regular species.

When creating a species in GAMA, some attributes are automatically integrated, such as the name (by default, the same name by which the species was defined), location, and shape. Attributes related to the context of the ABS under development must be explicitly defined, such as the actions and behavioral characteristics of each species. Actions, in relation to object orientation, can be compared to methods, and they enable the interaction and behavior of the agents.

The instantiation of species (agents) for the simulation execution, as recommended by the language documentation, should be implemented in the init block of the global species Taillandier *et al.* [2019], as shown in Listing 1, line 43, which creates a number of agents. However, a regular species can also implement an init block to initialize particular attributes, such as their locations.

As a platform aimed at facilitating the creation of ABSs, GAMA offers additional resources not found in traditional object-oriented languages. This includes the availability of ready-to-use abilities, such as the movement ability, which provides predefined commands for agents to move and can be adjusted to change the speed and direction of agents as necessary. To implement interaction between agents, GAMA provides the ask command. This command allows an agent to interact with instances of its own species or other species. For example, the ask command can be used to check which agent is the closest in a specific situation. This flexibility allows for greater adaptation and customization of simulations according to the specific needs of each scenario.

In Sugarscape, the agents are ants. Therefore, it is necessary to define a species ant to instantiate the agents in the simulation execution. Listing 2 shows the source code of the ant regular species from the Sugarscape simulation. The ant species has three essential attributes: vision_ant, which determines the number of accessible regions for sugar checking; metabolism_ant, which symbolizes the energy expended to keep the ant alive; and initial_energy, which defines the amount of energy with which the ant is born Epstein and Axtell [1996]. In the ABS developed in this paper, minimum and maximum limits for initializing these attributes were specified, as previously described in the global species. Upon creation, a random value is chosen between these limits for each ant characteristic using the rnd() command in GAMA. Additionally, the ant species defines a my_cell attribute to store its position on the grid environment (detailed later in section 3.3), which is initialized with a random cell. An energy attribute represents the resource that keeps the agent alive. This attribute is updated considering the agent metabolism and the amount of sugar gathered.

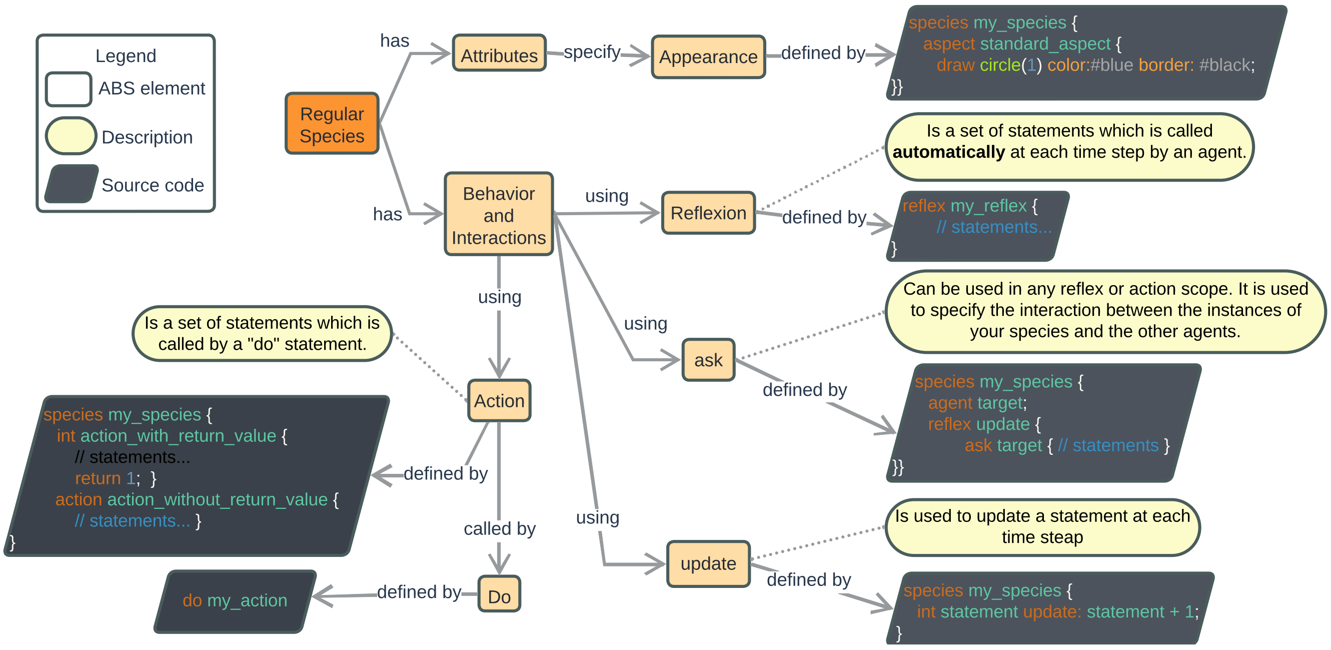
```
1  species ant {
2      // Sets the random values of agent's attributes
3      int vision_ant min: 1 <- rnd(vision);
4      int metabolism_ant min: 1 <- rnd(metabolism);
5      float initial_energy min: 5.0 <- rnd(max_energy);
6      cell my_cell <- one_of(cell);
7
8      // Updates energy quantity based on metabolic rates
9      float energy min : 0.0 <- initial_energy update : energy − metabolism_ant +
             my_cell.sugar max : max_energy;
10
11     init {
12         // Initialize attributes used to move
13         location <- my_cell.location;
14         my_cell <- choose_cell();
15     }
16
17     // Function to choose which cell the agent wants to move
18     cell choose_cell {
19         list <cell> available_cells <- my_cell.neighbours[vision_ant] where
                 (empty(ant inside (each)));
20         cell cell_with_max_sugar <- available_cells with_max_of (each.sugar);
21         if (cell_with_max_sugar.sugar <= my_cell.sugar) {
22             return my_cell;
23         } else {
24             return cell_with_max_sugar;
25         }
26     }
27
28     aspect default { // Appearance of agents
29         draw circle(1.0) color : #darkred;
30     }
31
32     reflex end_of_life when : (energy <= 0) {
33         do die; // The end of agents' life condition
34     }
35 }
```

Listing 2: Regular species source code

The ant species also implements the actions that the agents must perform during the simulation. In GAMA, these actions can be implemented in the following ways: through the Reflexion command, which calls a method automatically at each cycle; through the action command, a method that can be activated from a do command; through the ask command, which refers to an interaction between agents; and finally, the update command, which updates an attribute at each cycle. The reflexion and do commands are used in Sugarscape to define the condition that might cause the agents to die, as shown in Listing 2 lines 32–33. The update command is used to update the agent energy, as shown in Listing 2 line 9. In addition to agents' actions, the regular species allows specifying the appearance of agents by means of the aspect primitive, shown in lines 28–30.

In the Sugarscape simulation, ant agents move on the environment looking for places with sugar. The movement is obtained by updating the current position of agents, as shown in Listing 2 line 14. The choose_cell function finds and returns the position with maximum sugar among the neighboring positions that are not occupied with ants (lines 18–26).

Figure 4 presents a portion of the *Sugarscape* simulation implemented in the GAMA platform. In the left section, we find the *Users Models*, which allows developers to create customized ABSs, while a green button initiates the simulation. In the right section, there is the code where agent attributes can be configured, such as the random values for vision and metabolism of each agent, illustrating the use of the <- operator for value assignment. Additionally, the use of commands like update and reflex is evident, as mentioned earlier. Furthermore, there is the definition of agents appearance and the termination of their life.


Legend
ABS element
Description
Source code
Regular Species
has
Attributes
specify
Appearance
defined by
has
Behavior and Interactions
using
Reflexion
defined by
Is a set of statements which is called **automatically** at each time step by an agent.
using
ask
defined by
Can be used in any reflex or action scope. It is used to specify the interaction between the instances of your species and the other agents.
using
Action
Is a set of statements which is called by a "do" statement.
defined by
called by
Do
defined by
using
update
defined by
Is used to update a statement at each time steap

```
species my_species {
  aspect standard_aspect {
    draw circle(1) color:#blue border: #black;
}}
```

```
reflex my_reflex {
    // statements...
}
```

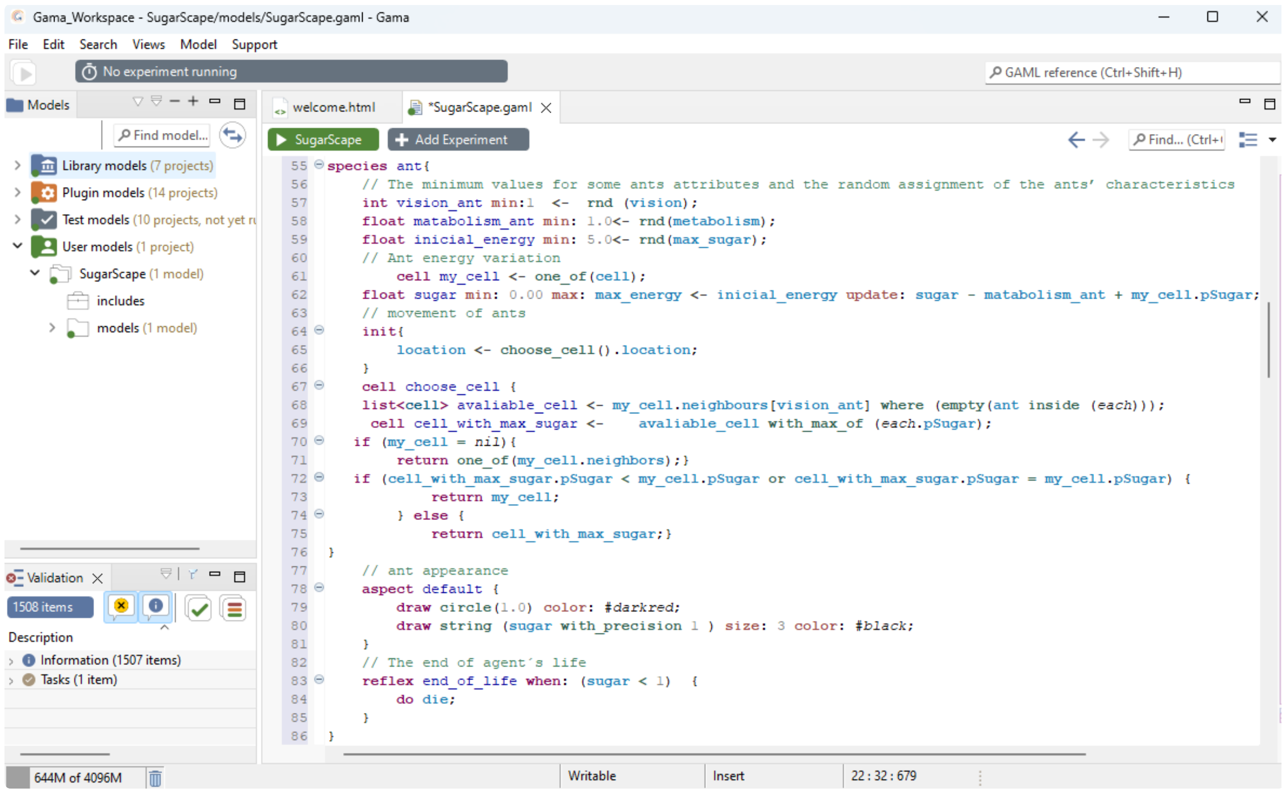
```
species my_species {
  agent target;
  reflex update {
      ask target { // statements }
}}
```

```
species my_species {
  int action_with_return_value {
      // statements...
      return 1; }
  action action_without_return_value {
      // statements... }
}
```

```
do my_action
```

```
species my_species {
  int statement update: statement + 1;
}
```

Figure 3. GAMA elements related to the regular species



```
species ant{
    // The minimum values for some ants attributes and the random assignment of the ants’ characteristics
    int vision_ant min:1  <-  rnd (vision);
    float matabolism_ant min: 1.0<- rnd(metabolism);
    float inicial_energy min: 5.0<- rnd(max_sugar);
    // Ant energy variation
        cell my_cell <- one_of(cell);
    float sugar min: 0.00 max: max_energy <- inicial_energy update: sugar - matabolism_ant + my_cell.pSugar;
    // movement of ants
    init{
        location <- choose_cell().location;
    }
    cell choose_cell {
    list<cell> avaliable_cell <- my_cell.neighbours[vision_ant] where (empty(ant inside (each)));
     cell cell_with_max_sugar <-    avaliable_cell with_max_of (each.pSugar);
   if (my_cell = nil){
        return one_of(my_cell.neighbors);}
   if (cell_with_max_sugar.pSugar < my_cell.pSugar or cell_with_max_sugar.pSugar = my_cell.pSugar) {
            return my_cell;
        } else {
            return cell_with_max_sugar;}
}
    // ant appearance
    aspect default {
        draw circle(1.0) color: #darkred;
        draw string (sugar with_precision 1 ) size: 3 color: #black;
    }
    // The end of agent´s life
    reflex end_of_life when: (sugar < 1)  {
        do die;
    }
}
```

Figure 4. The SugarScape code on the GAMA platform

3.3 Grid Species

The *Grid* species, whose essential elements are shown in Figure 5, is unique compared to other species, as it has unique attributes and behaviors that are essential for defining the grid environment topology where the Sugarscape simulation takes place. Unlike conventional species, it is automatically generated and is present in all simulations. The *Grid* is responsible for organizing the map as a structure that represents a grid or spatial mesh on which agents can move and interact in a simulation. This grid is composed of cells, and each cell can contain different attributes, such as the amount of available resources, the presence of obstacles, or other relevant information to the ABS.

By default, the *Grid* species has a series of attributes, such as the number of rows and columns, as well as the predefined color. However, all these characteristics can be modified if we define the species in the code. When manipulating the *Grid*, we are altering a matrix that, by default, has dimensions of 100x100. In other words, all changes will permeate every value of the matrix. Additionally, when defining the

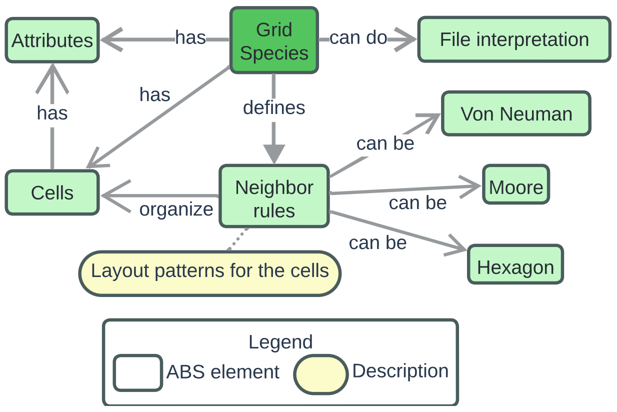


Figure 5. GAMA elements related to the Grid species

Grid species, it is necessary to specify which neighbor rule is adopted for the cells. GAMA supports the following neighbor rules: *Von Neumann*, where each cell perceives 4 neighbor cells (horizontally and vertically); *Hexagon*, perceiving 6 neighbor cells; and *Moore*, perceiving 8 neighbor cells.

In Sugarscape, the *grid* species is defined with dimensions of 50x50, as shown in Listing 3. Each cell has attributes representing the current amount of sugar and the maximum amount of sugar it can hold. A sugar growth rate can be added as a cell attribute, which varies based on both consumption and progressive recovery. This feature is implemented using the `update` command, providing greater dynamism to the simulation. The maximum sugar amount is initialized from the CSV file loaded in the global species. A value of zero indicates the absence of sugar in the cell. The adopted neighborhood rule is the *Von Neumann*, where 4 neighboring cells are considered by the agent when searching for cells with sugar to move to.

```
1  grid cell width: 50 height: 50 neighbors: 4 {
2      float max_sugar; // Read from the csv file
3      float sugar_growth_rate <- 1.0;
4      float sugar update: sugar + sugar_growth_rate max: max_sugar;
5      map<int, list<cell>> neighbours;
6
7      init { // A way to deal with neighbours bug
8          loop i from: 1 to: vision {
9              neighbours[i] <- self neighbors_at i;
10         }
11     }
12
13     reflex updateColor {
14         if (sugar = 1) {
15             color <- rgb(250, 250, 210);
16         } else if (sugar = 2) {
17             color <- rgb(247, 246, 167);
18         } else if (sugar = 3) {
19             color <- rgb(243, 242, 126);
20         } else if (sugar = 4) {
21             color <- rgb(240, 241, 50);
22         } else if (sugar = 0) {
23             color <- rgb(254, 254, 251);
24         }
25     }
26 }
```

Listing 3: Grid species source code

The visual aspect of the simulated grid environment is also defined in the Grid species, where it is possible to implement which colors will be used for its cells, as shown in Listing 3 lines 13–25. In Sugarscape, darker shades of yellow were defined to represent cells with a higher concentration of sugar and lighter ones to represent cells with a low concentration of sugar. This visual representation facilitates the interpretation of simulation results and provides important insights into the behavior of agents and the environment.

3.4 Experiment

For an ABS to be effective and representative, its visual should encompass all the characteristics specified in the various species in the simulation. This requires a precise definition of the experiment where the user can configure inputs, outputs, and behaviors as needed for the simulation. Figure 6 presents the essential element of experiments. In GAMA, four main types of experiments can be implemented to meet different needs and contexts Taillandier *et al.* [2019].

- *GUI*: It offers a graphical user interface that allows interaction with the simulation. This facilitates parameter manipulation and real-time observation of results, providing a more intuitive and immersive experience.
- *Batch*: It allows running the simulation multiple times, with the possibility of pre-changing parameter values for each run. This approach is useful for comparative analyses and evaluating results in different scenarios, providing insights into the model's behavior under various conditions.
- *Test*: It focuses on performing utility tests to ensure the quality and integrity of the simulation. This approach is essential for validating the accuracy and robustness of the model, identifying any potential flaws or inconsistencies that could compromise the results.
- *Memorize*: It maintains a detailed record of each step taken during the simulation. This allows the user to review and modify any stage of the process as needed, providing greater control and flexibility in conducting the simulation.

When defining the experiment, it is necessary to specify the type of simulation adopted. The visual and interactive configurations of the simulation should be also specified. This includes determining how relevant information will be presented through graphs and monitors, which play a role in interpreting the data and making informed decisions based on the results obtained.

Listing 4 presents the experiment source code for the Sugarscape simulation. As we can see, the GUI simulation type is adopted. Visual interface components can be assigned to global parameters, as shown in lines 2–3. That way users

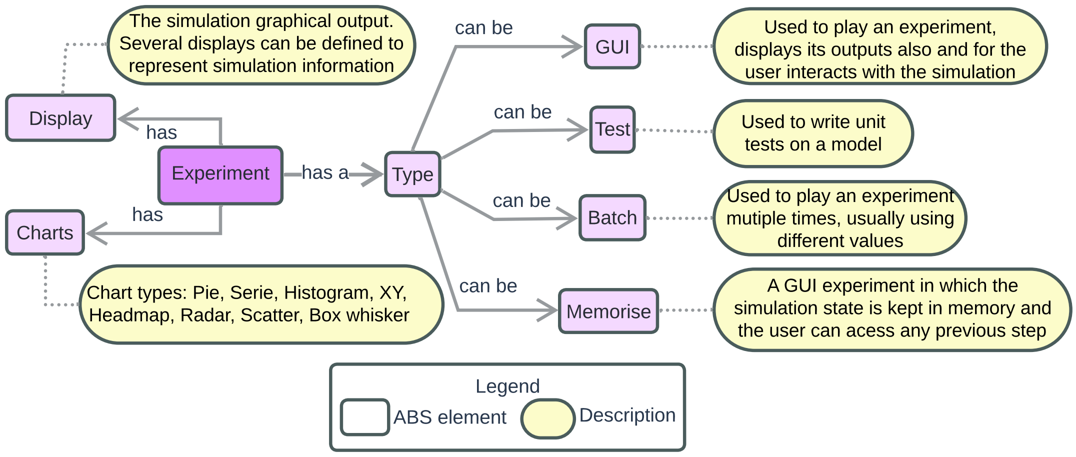

The simulation graphical output. Several displays can be defined to represent simulation information
Display
has
Experiment
has a
Type
has
Charts
Chart types: Pie, Serie, Histogram, XY, Headmap, Radar, Scatter, Box whisker
can be
GUI
Used to play an experiment, displays its outputs also and for the user interacts with the simulation
can be
Test
Used to write unit tests on a model
can be
Batch
Used to play an experiment mutiple times, usually using different values
can be
Memorise
A GUI experiment in which the simulation state is kept in memory and the user can acess any previous step
Legend
ABS element
Description

Figure 6. GAMA elements related to experiments

can easily set up parameters such as the initial number of ants or energy limits. This flexibility streamlines the setup process, enabling diverse simulation scenarios and fostering deeper insights into the complex behaviors of agents in the Sugarscape environment.

```
1  experiment simulation type: gui {
2    parameter ”Max values of vision: ” var : vision min : 1 max: 25 category :
         ”Ant”;
3    parameter ”Initial number of ants: ” var : nb_initial_ant min : 1 max : 2500
         category : ”Ant”;
4    output {
5      display display_grid {
6        grid cell;
7        species ant aspect : default;
8      }
9      display Population refresh : every(5 #cycles) type : 2d {
10       chart ”Population” type : series size : {1, 0.5} position : {0, 0} {
11         data ”Number of ants” value : nb_ant color : #black;
12       }
13     }
14     display Vision refresh : every(5 #cycles) type : 2d {
15       chart ”Vision” type : series size : {1, 0.5} position : {0, 0} {
16         data ”Vision average” value : average_vision color : #black;
17       }
18     }
19     display Metabolism refresh : every(5 #cycles) type : 2d {
20       chart ”Metabolism” type : series size : {1, 0.5} position : {0, 0} {
21         data ”Metabolism average” value : average_metabolism color :
               #black;
22       }
23     }
24
25     monitor ”number of ant” value : nb_ant;
26
27     display ”my_display” {
28       chart ”my_chart” type : histogram {
29         datalist (distribution_of(ant collect each.energy, 10, 0, 200) at
               ”legend”) value : (distribution_of(ant collect each.energy,
               10, 0, 200) at ”values”);
30       }
31     }
32   }
33 }
```

Listing 4: Experiment source code

With the GUI generated by GAMA, it is possible to explore and analyze the behavior of agents in the environment. This choice allows for direct interaction with the simulation through display elements. Displays, which are graphical representations of simulation elements, facilitate understanding and experimentation with different configurations to investigate various aspects of the phenomenon under study.

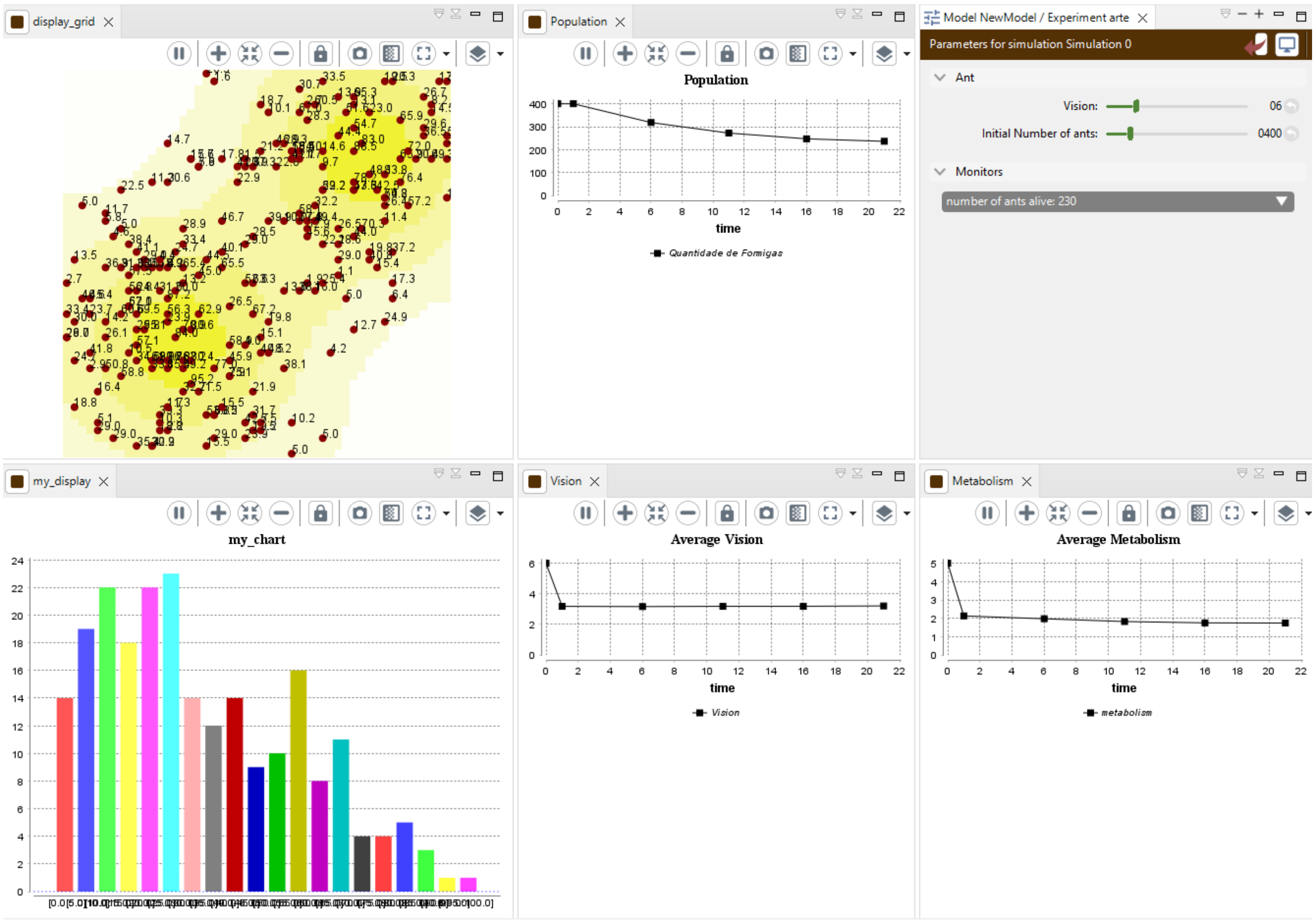
In the experiment species of Sugarscape, four display elements are defined, shown in Listing 4 lines 5–23. The first display, called display_grid, is a visual representation of the simulation's grid, showing cells and agents. The other three displays specify plot elements that display aggregated data from the various agents collected through the definition in the global species: Population is a line plot that shows the number of ants in the simulation over time; Vision is a line plot showing the average vision of the ants over time; and Metabolism is also a line plot, displaying the average metabolism of ants.

Besides display elements, the experiment species also allows the definition of monitoring elements. These elements show values throughout the execution of the simulation, providing the user with a quick and convenient way to monitor the simulation's progress and make decisions based on the presented data. In Sugarscape, a monitor is defined to display the number of ants in the simulation (line 25). Finally, a histogram display is used in the Sugarscape simulation to show the distribution of agents according to their energy levels (lines 27–31).

Figure 7 shows how the visual representation of the ongoing simulation is presented, comprising four distinct monitors. The first monitor, labeled *Display grid*, graphically displays the distribution of ants on the map grid, accompanied by a histogram that assesses the energy distribution among live ants. Subsequent monitors, *Population*, *Average Metabolism*, and *Average Vision*, provide the variation in the population of live agents, average metabolism, and average vision of ants over time. Additionally, the simulation view includes interactive parameters allowing users to adjust the agents' vision range and modify the initial quantity of ants, alongside a real-time monitor showing the number of live ants in the simulation, providing an instant view of the current population status.

4 Conclusion

This work presented a structured report on ABS development with the GAMA platform. To reveal the essential elements for implementing ABS in GAMA, a case study was conducted, in which Sugarscape, a simple and widely known ABS, was developed. A set of diagrams was created to highlight the essential structure of an ABS in GAMA. The complete Sugarscape implementation is available online. Together, the structured report and implementation provide an introductory guide for beginners on the GAMA platform.



Vision: 06
Initial Number of ants: 0400
number of ants alive: 230

Figure 7. SugarScape simulation made on GAMA

Developing on the GAMA platform can be a significant challenge, even for those familiar with ABS. This is partly because the platform is still under development and may exhibit errors in executing its commands. Some issues in GAMA require workarounds. One such issue involves the neighbours command, which was supposed to return the neighboring cells to the agent's current position. However, this command did not worked as expected, ignoring the predefined configuration specified for the grid. This issue is reported to the GAMA developers[5], through the platform's mailing list. The GAMA developers acknowledge the problem with the command and suggest an alternative method to achieve the desired execution. An issue with the histogram graph is also observed[6], which prevents configuring the histogram to adapt to the constantly varying energy levels of the agents.

The available documentation can be confusing for beginners, lacking in information and organization, which contributes to an inadequate understanding for the developer. Therefore, a recommendation for beginners on the GAMA platform is to interact and post questions on the mailing list. However, it is important to note that responses from the GAMA developers often take a significant amount of time, and the answers may be insufficient for problem resolution.

To conclude, the GAMA platform facilitates the development of ABMs, especially when compared to object-oriented languages. However, it does require dedication and study from the developer to understand its elements and perform basic operations. The absence of materials that assist the beginner developer in GAMA contributes to the difficulty of accessibility of the platform for those developing their first ABM. Therefore, works like this are relevant to contribute to the popularization, accessibility, and development of GAMA.

Declarations

Funding

This research was partially supported by the Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina—FAPESC (TO2021TR882 and TO2023TR246).

Authors' Contributions

ARS is the main contributor ot this manuscript, and contributed to the software development and manuscript writing. FS contributed to the conception, research supervision, manuscript writing and revision. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests

Availability of data and materials

The softwares generated and during the current study are available in `https://github.com/agentbasedsimulations/sugarscape-gama-simulation`.

[5] https://groups.google.com/g/gama-platform/c/jOFr9ju7sS4/m/LekjUGQnAAAJ
[6] https://groups.google.com/g/gama-platform/c/fSDZzeQjs_E/m/wSdQZUVHAgAJ

References

Bousquet, F. c., Bakam, I., Proton, H., and Le Page, C. (1998). Cormas: Common-pool resources and multi-agent systems. In Pasqual del Pobil, A., Mira, J., and Ali, M., editors, *Tasks and Methods in Applied Artificial Intelligence*, pages 826–837, Berlin. Springer. DOI: 10.1007/3-540-64574-8_469.

Costa, A. C. R. (2023). The Tupinambá: An Exercise in Societal Modeling. In *Anais do XVII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações (WESAAC 2023)*, pages 44–54. DOI: 10.5753/wesaac.2023.33436.

Creswell, J. W. and Creswell, J. D. (2018). *Research design: Qualitative, quantitative and mixed methods approaches*. SAGE Publications, 5 edition.

Daudé, E., Chapuis, K., Taillandier, P., Tranouez, P., Caron, C., Drogoul, A., Gaudou, B., Rey-Coyrehourcq, S., Saval, A., and Zucker, J.-D. (2019). ESCAPE: exploring by simulation cities awareness on population evacuation. In *Proceedings of the 16th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2019)*, pages 76–93. Available at `https://shs.hal.science/halshs-02144058v1`.

Eisinger, D. and Thulke, H.-H. (2008). Spatial pattern formation facilitates eradication of infectious diseases. *Journal of Applied Ecology*, 45(2):415–423. DOI: 10.1111/j.1365-2664.2007.01439.x.

Epstein, J. M. and Axtell, R. L. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. MIT Press.

Gaudou, B., Huynh, N. Q., Philippon, D., Brugière, A., Chapuis, K., Taillandier, P., Larmande, P., and Drogoul, A. (2020). COMOKIT: A Modeling Kit to Understand, Analyze, and Compare the Impacts of Mitigation Policies Against the COVID-19 Epidemic at the Scale of a City. *Frontiers in Public Health*, 8. DOI: 10.3389/fpubh.2020.563247.

Kleiboer, M. (1997). Simulation Methodology for Crisis Management Support. *Journal of Contingencies and Crisis Management*, 5(4):198–206. DOI: 10.1111/1468-5973.00057.

Klügl, F. (2008). A validation methodology for agent-based simulations. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 39–43. ACM. DOI: 10.1145/1363686.1363696.

Klügl, F. and Bazzan, A. L. C. (2012). Agent-Based Modeling and Simulation. *AI Magazine*, 33(3):29–40. DOI: 10.1609/aimag.v33i3.2425.

Li, J. and Wilensky, U. (2009a). NetLogo Sugarscape 1 Immediate Growback model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. Available at: `http://ccl.northwestern.edu/netlogo/models/Sugarscape1ImmediateGrowback`.

Li, J. and Wilensky, U. (2009b). NetLogo Sugarscape 2 Constant Growback model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. Available at: `http://ccl.northwestern.edu/netlogo/models/Sugarscape2ConstantGrowback`.

Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10(2):144–156. DOI: 10.1057/jos.2016.7.

Pereira, A. H., Nardin, L. G., and Sichman, J. S. a. (2011). Coordination of Agents in the RoboCup Rescue: A Partial Global Approach. In *2011 Workshop and School of Agent Systems, their Environment and Applications*, pages 45–50. DOI: 10.1109/WESAAC.2011.10.

Santos, F., Nunes, I., and Bazzan, A. L. C. (2017). Model-driven engineering in agent-based modeling and simulation: a case study in the traffic signal control domain. In Das, S., Durfee, E., Larson, K., and Winikoff, M., editors, *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, pages 1725–1727, São Paulo. IFAAMAS. Available at: `https://dl.acm.org/doi/abs/10.5555/3091125.3091418`.

Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.-N., Marilleau, N., Caillou, P., Philippon, D., and Drogoul, A. (2019). Building, composing and experimenting complex spatial models with the GAMA platform. *GeoInformatica*, 23:299–322. DOI: 10.1007/s10707-018-00339-6.

Teixeira, L. and Santos, F. (2020). Uma Simulação com Agentes para Estudar a Propagação da COVID-19 em Ibirama(SC). In *Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional*, pages 176–187, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/eniac.2020.12127.

Wilensky, U. (1999). NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. Available at: `http://ccl.northwestern.edu/netlogo/`.

Wilensky, U. and Rand, W. (2015). *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. Mit Press.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2024). *Experimentation in Software Engineering*. Springer, 2 edition.