# Modeling Artificial Life Through Multi-Agent Based Simulation

Terry Lima Ruas, Maria das Graças Bruno Marietto, André Filipe de Moraes Batista, Robson dos Santos França, Alexandre Heideker, Emerson Aguiar Noronha and Fábio Aragão da Silva
*Center of Mathematics, Computation and Cognition, Federal University of ABC*
*Brazil*

## 1. Introduction

The Multi-Agent Based Simulation (MABS) area is placed at the intersection of two distinct areas: Distributed Artificial Intelligence (DAI) and Computational Simulation. This field of research provides a proper infrastructure for modeling and understanding the processes related to social interactions such as coordination, cooperation, training and coalition of groups and resolutions of conflicts, among others. Such understanding is made possible because of the relationship established between local and global behavior, which leads to leading to explicit chains of cause and effect of how internal agent components affect the agents behavior, how this behavior affects the agency and, dialectically, how the agency affect its agents components. Multi-agent simulation models are based on the concept of the individual-program relationship, which allows the simulation of artificial worlds where each entity (interactive computing entity) is represented as an agent that maps a single entity (or a group of them) in the target system. Since the infrastructure of technical and theoretical areas of simulation allows researchers to mimic the essential elements of a target system without having to work directly with the target system itself, it becomes handful when dealing with phenomena such as the spread of fire without hazarding the integrity of the environment and its living beings.
Artificial Life in Computers yields the creation of a laboratory capable of providing the necessary means to study, research, reproduce and maximize the simulations on a specific subject. As stated previously, a simulation model is a particular type of model that aims to represent a given system. However, it differs from classical models in the sense that it facilitates a) the study of how the modeled systems behave under certain conditions, and b) the examination, in varying degrees of detail, the consequences of changing internal behaviors of the system, and vice versa.
The results obtained in a simulation might be of great help in the decision-making process, in the evaluation of systems and in reducing implementation time and costs.
In (Ferber, 1996; Gilbert & Troitzsch, 1999) some simulation goals are presented, namely:

- Discover and formalize new theories and models;
- Develop a better understanding of some features of the real system;
- Test hypotheses of the modeled system;

- Predict future actions and behaviors.

More specifically, (Ferber, 1996) defines that an agent-based simulation model relates to the idea that a system is comprised of all relationships of its inner parts, and in that sense, it is possible to simulate an artificial world based on the relationships of its entities.
The simulation occurs when there is a transposition of the population of a target system [1] to a *conceptual model* equivalent, followed by the encoding of this model to a *computational model*. In this case, an agent (or actor) equates to a real world entity or a group of them. Such actors can be of different natures and with various granularities, such as humans, robots, computer programs, inanimate objects and organizations.
After the establishment of the multi-agent paradigm in the computer science, the role of multi-agent based simulation has been acquiring relevance in a variety of scientific disciplines. In particular, the sources of analogy between agent-based technologies and models of actual social systems, and the efforts towards dealing with such complex systems through simulation models have created this intense interdisciplinary effort that provided ground for the advent of a new scientific field, named Multi-Agent Based Simulation (MABS). As a result, research interfaces were created across various disciplines under the umbrella of a multidisciplinary area that involves researchers from as diverse fields of study as Psychology, Sociology, Economy and Computer Science.
Considering the relatively recent advent of MABS, its multidisciplinary aspect might also pose as one of the biggest challenges to be overcome by researchers, since it requires cutting across traditional boundaries of school of thoughts, mixing different theories, methodologies, techniques and point of views. In this chapter, the principles of multi-agent based simulation are presented, as well as some simulations that exemplify the integration of MABS and artificial life. To accomplish that, the chapter is divided in three main parts: the first part focus on the presentation of MABS concepts and techniques. The second part presents some of the main simulation platforms and frameworks available today and also analyses and compares two of them. The third and final part displays a set of models that aim to simulate artificial life though the use of MABS techniques.

## 2. Principles of Multi-Agent Based Simulation

The main goal of the Multi-Agent Based Simulation (MABS) researchers is to develop and study simulation models taking into consideration a theoretical-technical framework based on the Distributed Artificial Intelligence field. The general relevance of simulation, and more specifically agent-based simulation becomes so clear that some authors have gone far enough to consider it as a third way of doing science, along with traditional deduction and induction reasoning (Axelrod, 1998). It could be stated that simulation distinguishes from standard deduction and induction in its implementation and also in its goals. A simulation starts with a set of explicit assumptions (as in deduction) but not generally providing any theorems, producing data which are suitable for analysis by induction that come from a strictly set of assumptions.
Following that perspective, a simulation model is a kind of model that represents a specific target system. What makes this model distinct from the others is (i) the chance of studying the global behavior of the modeled system in certain conditions and (ii) the possibility to inspect the consequences of changes in the internal components of the system. An important aspect to be considered in simulation systems is the assurance that both conceptual and

[1] The target system is equivalent to the simulation domain and can be real or theoretical.

computational models accurately represent the target system, and that can be achieved by using two processes: validation and verification.
The validation process aims to certify that the conceptual model represents the target system in an acceptable degree of adherence. Thus, the validation processes fundamentally addresses a specific question: Does the simulation outcomes correspond to those from the target system? On the other hand, the verification process' main purpose is to assure that the conceptual model was correctly translated to the computational environment. Specifically, a multi-agent simulation model is based on the concept that it is feasible to simulate an artificial world inhabited by interactive computational entities. Such simulation can be achieved by transposing the population from a target system to its artificial counterpart. In that sense, an agent is similar to an entity or a group of entities of the target system. Moreover, agents can be of distinct natures and granularities, such as human beings, robots, computer algorithms, inanimate objects and organizations.
In a multi-agent based simulation, the relationship between agents can be specified in several ways, ranging from reactive to deliberative attitude approaches. In both cases, agents must be able to decide and perform their actions autonomously. Nevertheless, to ensure a proper execution of the simulation, agents actions must be synchronized through the scheduling of a minimum set of events, and their behavior can be either time-stepped scheduled - performed within each discrete time step - or event-driven scheduled, in which agent actions are scheduled by other agents' actions and/or events.
The MABS area provides a suitable infrastructure to model, study and understand the processes related to complex social interactions such as coordination, collaboration, group formation, evolution dynamics of norms and conventions, free will and conflict resolution, among others. That can be achieved by relating local and global behavior and analyzing how agents can affect the environment and other agents (and vice-versa, leading to explicit chains of cause and effect), how internal agent components affect the agent's behavior, how this behavior affects the agency and, dialectically, how the agency affects its agents components (Gilbert & Troitzsch, 1999).

### 2.1 Multi-agent modeling

Multi-agent simulations require the development of multi-agent models, which aims to model complex real-world systems as dynamical systems comprised of interacting autonomous decision-making entities called agents. Traditional analytical methods might not be suitable to deal with complex phenomena that are simply too complicated to be analytically tractable, especially when involving non-linear relationships. Multi-agent models have then emerged as an alternative for these types of problems. In recent scientific literature, many denominations for agent-based modeling can be found, such as: Individual-Based Modeling (IBM), Agent-Based Systems (ABS), among others.
An agent-based model is essentially a population of heterogeneous agents, which represents autonomous entities that interacts between themselves and with their environment, allowing the formation of a social system where aggregated structures (patterns) emerge from those interactions. The fundamental principle of an agent-based model is the emergence of social structures and groups of behaviors from the interactions of individual agents. These agents operate in artificial environments and under specific rules that are valid only when taking into account the limitations of each agent regarding their own computational and memory capabilities.

In Table 1 a comparison between a traditional and agent-based modeling is presented (each of the aspects is explored in the subsequent sections of this chapter).

| **Traditional** | **Agent-Based** |
|---|---|
| Focus on continuous time | Focus on discrete time |
| Mathematical language (equations) | Descriptive model |
| Aggregate level granularity | Individual level granularity |
| Top-down (macro-to-micro) approach | Bottom-up (micro-to-macro) approach |
| Pre-defined behavior | Emergent behavior |
| Global control | Local control |

Table 1. Comparison Between Traditional and Agent-Based Modeling.

## 2.2 Main aspects of multi-agent models

Simulation models based on multi-agents are comprised of a number of heterogeneous agents, relationships between these agents and an environment capable of simulating the behavior and interactions of such agents. Also, there is no central authority in charge, as agents are modeled to behave autonomously in a self-organized model based on simple local rules of interactions between agents and the environment. The ultimate goal of such model is to allow the emergence of system-level phenomena resulting from these local interactions between agents themselves and the environment.

A more specific definition of agent would be of a discreet entity with its own objectives and behaviors. Each agent contains internal states and behavior rules, allowing them to interact with other agents and the surrounding environment. Agents are also autonomous and display some degree of initiative, allowing them to behave as object-oriented entities. They are modeled to execute the vast majority of their actions without any direct interference from either humans or other computational agents. Examples of agents include people, groups, organizations, social insects, swarms, robots, and so forth.

### 2.2.1 Ascending (bottom-up) modeling

Agent-based models are built from agents that have very simple rules defined for their behavior. The interactions between these agents create collective structures in an ascending approach instead of a descending one, where the macro structures and behavior of a system would be modeled and then used to explain micro interactions of its components. Modeling a complex system using a top-down approach would prove much too complex and not appropriate as a complex system behavior is the result of a large number of interactions. An analytical/reductionist approach is also not adequate for modeling complex systems as it assumes that the system behavior can be understood by analyzing its parts separately.

So a bottom-up model is therefore more suitable for complex systems such as the ones applied to artificial life simulation, as the bottom-up approach focus instead on simple rules of behavior for small parts of a system - its agents - and how they interact with each other, making use of computational power to simulate a large number of those agents and their interactions, allowing emergent patterns to be observed and studied. The model can then be easily manipulated in terms of addition or removal of its micro-level individual properties and how these changes might affect the macro-level social phenomena. For instance, a bottom-up model for an ant colony would describe ants in a micro-level and in terms of their behavior as individuals in the colony and how they communicate to each other. A simulation tool would then be used to mimic the colony environment where several individual ants are put to communicate and perform tasks, allowing an observer to study the emergence of colony-level social phenomena.

### 2.2.2 Complex systems

The word "complexity" has roots in two Latin words: "complexus" which means "totality" and "completere" which means "to embrace". So complex system are formed by two or more interlinked components that creates a network of objects interacting with each other, displaying a dynamic and aggregated behavior. In this context, the complex adjective is not to be confused with complicated. Moreover, in a complex system the action of a single element might affect the actions of other objects in the network, making the famous paradigm 'The whole is more than the sum of its parts' even more true.
In fact, complex systems are made of several simple behavioral units that influence each other mutually in an intricate network of connections that ultimately generates a global complex behavior. As a result of such a systematic behavior, many properties of a complex system can only be observed during its collective behavior and cannot be identified in any of its fundamental units. One example of a complex system is the fire propagation phenomenon. An adequate (non-analytical) approach to treat complex systems, more specifically the fire propagation phenomenon, is to use simulation techniques based on Cellular Automata (CA). Simulating complex systems allows researchers to (a) propose new structures or alternatives to treat social systems, studying and understanding their existence and operation; (b) have a better understanding of the social, anthropological, psychological aspects, etc. used to describe and explain the analyzed phenomena and (c) to use existing theoretical models already proven effective when dealing with institutional and social processes.

### 2.2.3 Unpredictable systems

Unpredictable systems are complex systems with a high degree of instability and unpredictability in the decision-making process, and such aspects need to be treated in a dynamic manner. According to (Lempert, 2002), agent-based models are often useful under conditions of deep uncertainty where reliable prediction of the future is not possible by either in a best estimate or probabilistic approaches (such as the ones in traditional simulation models).
In his work, (Lempert, 2002) argues that agent based models are useful at describing the behavior of inherently unpredictable systems. According to him, the predictive policy analysis is an example of application of agent-based simulations, as police simulators may be effective in situations where the standard methods of predictive policy analysis are least effective. Also, in dynamic and unpredictable systems, agents must be modeled in a way that their deliberation and responsiveness are balanced so that they act appropriately. This must be done to avoid long deliberations that might impact the performance of the simulation but also to avoid agents to become too reactive to choose the best action to execute.

### 2.2.4 Emergent behavior

According to (Axelrod, 1998), 'emergent properties' of a system can be described as the large-scale effects of locally interacting agents, noticed as non self-evident, stable macroscopic patters arising from individual agent's local rules of interaction. Below is a non-exhaustive list of situations when agent-based models are useful for capturing emergent behavior:

1. The interactions between agents are discontinued, nonlinear. This can be particularly useful when describing complex individual behavioral. Discontinuity proves much too complex by using traditional analytical methods (for instance, differential equations);
2. There is a significant necessity of designing a heterogeneous population of agents. The heterogeneity allows agents with clearly distinct rationality and behavior to be modeled;
3. The topology of the agent's interactions is complex and heterogeneous. This can be particularly useful when modeling social processes, specially the inherent complexity of physical and social networks.

Emergent phenomena can also be formalized as requiring new categories to describe them, which are not necessary to describe the behavior of the model's underlying components (i.e. agents) (Gilbert & Terna, 2000). In some models, the emergent properties can be formally deduced, but they can also be unpredictable and unexpected, as anticipating the consequences of even simple local interactions sometimes proves to be a hard task. Also, according to (Axelrod, 1998), an example of emergent phenomenon can be seen in a model where agents represent consumers and have local behavior rules that allow them to choose and buy brands of video tapes according to the availability of machines on which to play it. Only by analyzing the agent's local rules, one would not intuitively notice that the simulation model is most likely to lead one format to completely overcome the other.
Moreover, mathematical analysis might be limited in its ability to derive the dynamic consequences in models where, for instance, agents have an adaptive behavior influenced by their past experience. For this type of situation, a simulation model is usually the only feasible method.

### 2.2.5 Open systems and self-organization

Self-organization is a process where the organization of a system is not guided or managed by an outside source. Self-organizing systems normally represent open systems and might typically display emergent properties. Open Systems in turn can be described as a system with high environmental adaptability through quick incorporation of new elements, information and ideas. On the other hand, a closed system resists the incorporation of new ideas and risks atrophy, ceasing to properly serve the environment it lives in.
Self-organization is considered an effective approach for modeling the complexity in modern systems, allowing the development of systems with complex dynamics and adaptable to environmental perturbations without complete knowledge of future conditions. According to (Gardelli et al., 2008), "*The self-organization approach promotes the development of simple entities that, by locally interacting with others sharing the same environment, collectively produce the target global patterns and dynamics by emergence. Many biological systems can be modeled using a self-organization approach*".
Some examples of self-organizing environments include food foraging in ant colonies, nest building in termite societies, the comb pattern in honeybees, brood sorting in ants, decentralized coordination for automated guided vehicles, congestion avoidance in circuit-switched telecommunication networks, manufacturing scheduling and control for vehicle painting and self-organizing peer-to-peer infrastructures, among others.

### 2.2.6 Local and global control

Governing laws of behavior for individual agents in a multi-agent simulation model can be implemented as either local, global or a combination of both. The decision depends on the type of simulation being modeled and the constraints imposed by the problem being solved.

According to (Parker, 1992), emergence might occur solely based on the interaction of the local control laws of the individual agents, which might not be aware of any global goals. However, this approach might not be sufficient to model some simulations where agents are expected to cooperate towards a global goal, and a hybrid model with both local and global control might offer a solution. Still according to (Parker, 1992), the key difficulty when designing control laws governing the behavior of individual agents is to find the proper balance between local and global control and how to design such controls: as global goals designed into the agents, as global or local knowledge or through a behavioral analysis method.

## 3. Multi-agent based simulation platforms

This section presents two frameworks that aid in the building and execution of SMA's: Swarm (Group, 2009) and Mason (Cioffi-Revilla & Rouleau, 2010). In this context, the main aspects of each platform are covered in order to establish a comparative overview among these platforms. Although the scope of this section is limited to these two platforms, it is worth emphasizing that there are many others multi-agent frameworks being used by scientists, for example Repast, NetLogo, and etc.

### 3.1 The Swarm Platform

The Swarm Platform was created in 1994 at the *Santa Fe Institute, USA* by Christopher Langton with the help of other researchers. It was written in *Objective C*, but later a *Java* interface was also developed. The Swarm Platform offers multi-agent researchers a good variety of resources such as memory management, action scheduling, graph generation, real-time simulation updating/interference, etc.
A Swarm can be described as a type of animal behavior characterized by the reunion of many similar entities that together seem to behave as a bigger, single organism, such as a school of fishes swimming at the sea or a swarm of bees flying in the sky. This type of behavior displays a noticeable degree of flexibility (a swarm of insects adapting to environmental changes such as the wind, rain, smoke, etc.), robustness (a global objective will still be pursued - and most likely achieved - even if some of the members of the swarm are lost during the execution of the task), decentralization (there is no central control as in a fish shoal) and self-organization (insects in a swarm will organize themselves to achieve a global objective).
Following that philosophy, the Swarm Platform was developed to allow the mimic of such features and concepts, modeling the agents with reactive features and actions. A second feature provided by this platform is the creation of hierarchical models. In other words, it could be possible to design multi-agent simulations which the agents are composed by other agents, forming a multi-agent simulation by itself, or a simulation of nested simulations. This allows the formation of systems with a high level of complexity.

#### 3.1.1 The Swarm Platform architecture

The basic component that organizes agents in the Swarm platform is called SWARM. A SWARM can be described as a collection of agents under a schedule of events and represents the entire model, as it contains all agents within then model as well as the representation of time. The basic architecture of a swarm simulation is comprised of a MODELSWARM, an OBSERVERSWARM and, optionally, simulation PROBES. Figure 1 displays the basic architecture of a simulation in the Swarm Platform.
The MODELSWARM contains the conceptual model implementation, and is comprised by a SWARM and optional sub-swarms. In this architecture, active and passive agents are defined

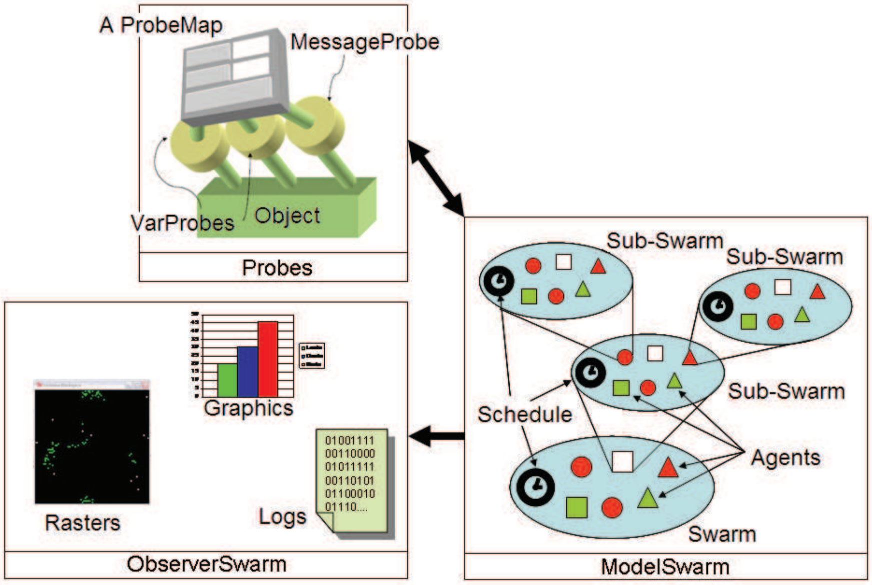


Fig. 1. Basic Architecture of a Simulation in the Swarm Platform.

along with their structures, features and their influential environment. The OBSERVERSWARM is responsible for collecting the simulation data and the consequent presentation of such data through charts, animations, files, etc. Finally, optional PROBES can be implemented so the user can interact with the simulation in real time by observing and changing variables, agent states and so forth.
Another essential element in a Swarm simulation is the SCHEDULER, which is responsible for synchronizing the actions of agents of each swarm. Only one scheduler is allowed for each swarm in a Swarm simulation

### 3.2 The Mason platform

The "Multi-Agent Simulator Of Neighborhoods" (MASON) Platform is a multi-agent simulation framework written in Java, comprised of a model library and a set of 2D and 3D visualization tools. It is a result of a coordinated effort between The Evolutionary Computation Laboratory and The Center for Social Complexity at George Mason University. Among its characteristics, it is worth mentioning:

- Models are completely independent from their visualization;
- Portability between different platforms allowing the production of identical results;
- Native generation of simulation snapshots and movies from the simulation data.

Figure 2 shows the basic architecture of the Mason platform.
Just as the Swarm Platform, the Mason Platform contains a scheduler that allows the simulation of discrete-time events. However, a significant difference between the two schedulers is that in Mason the scheduler schedules agents instead of events, while in Swarm the events themselves are the ones scheduled by the scheduler.

and general concepts, (ii) a computation model which, after being implemented, could be analyzed empirically in a safe and controlled environment, (iii) tools for improving and revising the theories used for building the model, (iv) new ways of looking into the social theories using, for example, artificial worlds or a distinct set of rules for a real case scenario that would never be feasible in a physical experiment.
Thanks to the multi-agent based simulations, a new family of multidisciplinary projects is coming. Social theorists, mathematicians, computing specialists, biologists, linguists and many other professionals are working together to model and to improve the social simulations.

### 4.1 Artificial life and MABS in practice

At the following sections, a general overview of the research work of the Artificial and Social Intelligence Group at Center of Mathematics, Computation and Cognition, Federal University of ABC (UFABC) is presented to exemplify how multi-agent simulations can be used to model artificial life systems. The applications are classified into groups according to the following: Group I (basic applications), Group II (focus on information exchange) and Group III (advanced models). Such division is performed to provide a better understanding of the MABS' potential.

#### 4.1.1 Group I: Basic applications

Two basic applications are well-known in the MABS area: Conway's Game of Life and the Prey-Predator Model. Such applications provide the basis for several other MABS applications, as they successfully represent most basic MABS concepts, such as emergency, self-organization, bottom-up modeling, etc.

##### 4.1.1.1 Conway's game of life

A Cellular Automata (CA) is a set of cells disposed in a bi-dimensional grid where each cell might assume a certain state, with the number of possible states being finite. All of the Cellular Automata cells will simultaneously evolve to the next generation according to the same set of evolution rules.
The Game of Life was created in 1970 by a British mathematician called John Conway, which extended the work of John von Neumann. Having simplicity as the guiding rule for his work, Conway achieved impressive results. Contrary to the work of John von Neumann, which was comprised by a large number of rules, Conway's Game of Life is comprised by few simple rules. In an orthogonal grid, each cell can assume two states, alive or dead. During each turn (or time-step), the automata determine whether each cell will be alive or dead based on four simple rules, as follows:

- any live cell with fewer than two live neighbors dies by loneliness;
- any live cell with more than three live neighbors dies by overpopulation;
- any dead cell with exactly three live neighbors becomes alive;
- any live cell with two or three live neighbors survives.

Figure 3(A) shows the execution of the first rule, where isolated cells die by loneliness. Figure 3(B) shows the execution of the second rule, where a cell with more than three live neighbors dies by overpopulation. Third rule can be seen in Figure 3(C), where each cell with exactly three live neighbors becomes alive.

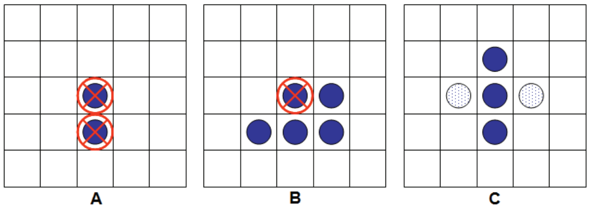


Fig. 3. Examples of rules execution in a Conway's Game of Life.

The results that turned the Game of Life an example of complex system are still a subject of research. These four simple rules are capable of generating a broad range of patterns, from static ones to complex arrangements that will look and behave as "creatures" that seem to walk through the grid, many times destroying other clusters of live cells or originating new creatures. Figure 4 shows an example of static pattern generated by such rules. Figure 5 shows a classic example of a formation known as "Pulsar", where the cluster of cells will alternate between the three states shown on each step of the simulation.

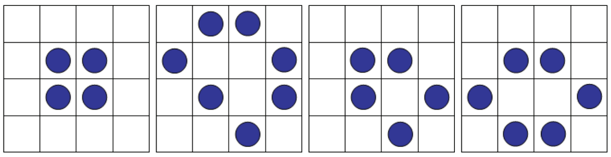


Fig. 4. An Example of a Static Pattern.

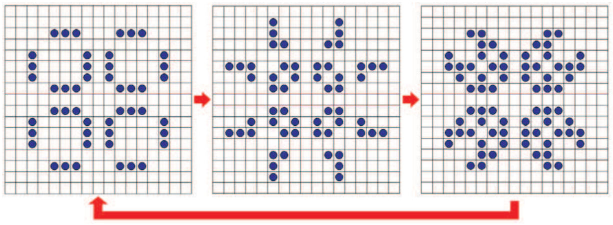


Fig. 5. An Example of a Pulsar Creature and the Corresponding States.

### 4.1.1.2 The Prey-Predator model

There are many examples of self-sustained, complex systems in nature. The Food Chain is one of such systems, and it can be described as a hierarchy of consumers. An herbivore eats grass and, in turn, a carnivore eats herbivores. Should something happen to this chain, there might be a minor shift in one of the chain members' population which could lead to a major shift in higher level chain members. Therefore, the Food Chain system can be understood as a complex system since the overall behavior of that system relies on how each element interact with each other, and not by the individual behaviors alone (Engbert & Drepper, 1994).
The Predator-Prey Model - modeled after the Food Chain system - is a prime example of a heterogeneous system: there are populations (groups) of distinct species. Some of the behaviors found in these species are the same, such as moving and escaping. However, some species must deal with the hunt of preys and the escape from predators. The interaction between predators and preys makes this system complex, in spite of the simple behaviors

displayed by each element. The purpose of the simulation is to model an environment where there are preys that attempt to run away from their predators while looking for food, and predators that are hunting for these preys. This kind of simulation addresses both populations and hazardous elements that could decrease them, as well as special elements (such as Cheese) that could increase them.

**Environment settings**

The environment is described as a bi-dimensional grid with 50 rows by 50 columns, and each cell can hold a numeric value. "0" stands for an empty space. "4" indicates a slice of cheese. Values 1,2 and 3 represent a dog, a cat and a rat respectively.

**Agents**

At first, each agent fights for its survival, hunting for preys and running away from predators, and performs the following actions at each simulation step:

**Dog** The DOG tries to notice a CAT. If there are no CATS close to it, a random movement is performed. If there is a CAT nearby, the next step is to establish whether the DOG hit the CAT. If it did, the CAT is killed. Otherwise, the newfound CAT is pursued by the DOG.

**Cat** The CAT follows a similar behavior to the DOG. In other words, the CAT looks for a RAT nearby and, if a RAT is noticed, the CAT starts pursuing it. Likewise, if the RAT is hit, it is killed. However, a runway element is added to the CAT: if a DOG is nearby, the CAT behaves like a pray, running away from the DOG. If there is neither a DOG nor a RAT nearby, a random movement is performed.

**Rat**

Essentially, the RAT runs away from the CAT and tries to follow the biggest Cheese gradient. If a Cheese is hit, the RAT eats it, and in the place where the Cheese was found, a new RAT is born. If there is neither a clear major Cheese gradient (one that represents the RAT's direction change) nor a CAT nearby, the RAT will perform a random movement.

**Standard actions**

There are some standard actions for all agents. For these actions, the agents' viewing area is a two-cell Von Neumann's neighborhood, and they can be described as follows:

**Notice Agent** To search a certain agent.

**Follow Agent** If a target-agent is close enough, the agent will pursuit it.

**Run Agent** If a predator is detected, the agent will run in the opposite direction of this dangerous agent.

**Hit Agent** To identify an agent in a one-cell Von Neumann neighborhood.

**Kill Agent** If a target-agent has been hit, it must be killed, which means that its cell must be clean (get a zero value).

**Random Movement** If there is no target-agent nor predator is noticed or hit, the agent will move randomly in one of the eight possible directions (N,NE,NW,S,SE,SW,E,W).

**Follow Major Gradient** Only available for the rats, it represents a linear search where the largest concentration of cheese in the environment can be found.

The Figure 6 shows the execution of the predator-prey model using the Swarm Framework. The yellow dots represent the cheese, the blue ones the dog agents, the red ones the cats and the white ones are the rats.

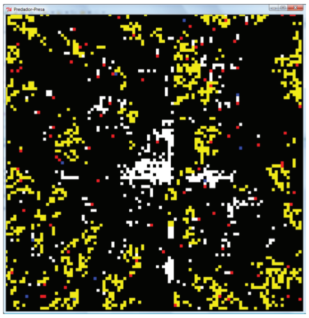


Fig. 6. The Predator-Prey Model Simulation Running.

#### 4.1.2 Group II: Applications focused on information exchange and coordinated actions

This section presents simulations that are focused on the communication and exchange of messages among agents in order to obtain a common objective.

4.1.2.1 Exploring Robots

The resolution of search problems represents a vast research field in AI, influencing the Robotics area as well. Movement planning, execution and the recognition of an unknown territory allows autonomous devices to successfully perform such tasks in highly dynamic environments. In face of that, let's consider an unknown environment (such as a maze or a room with walls) where a group of robots share information while creating a collective memory, in order to map the environment so that an exit can be found. Besides detecting and mapping walls, the robots are supposed to navigate thought the environment avoiding collisions and using routes not visited before while determining the best escape path. Below, an overview of the model and the simulation that represent the aforementioned situation is presented.

**Conceptual model**

The model of a simulated exploring robot will be implemented using the Swarm platform, where several independent agents build their own maps using a simulated computational vision system. The idea is to make use of technologies such as computational vision, path finding, maps creation, etc. The basis for this type of implementation is an already discretized model, containing walls and at least one exit that will be the agent's main goal. The premises for the creation of the conceptual model are as follows:

- The agent simulates a robot, thus is important to consider its characteristics and limitations;
- The robot is equipped with four sensors which are capable of detecting any front, lateral or rear obstacle;
- Walls detected by such sensors are automatically added to the map;
- A single map is shared among all robots, which means that robots collaborate on building a collective representation of the environment;
- The map can be trusted, which means that the correctness of the information provided by the robots is assured;
- Routes already mapped must be avoided as unknown routes are prioritized;

- Whenever enough information to find the exit is available, the most efficient algorithms must be used.

**The map and the environment**

Figure 7 shows an example of environment used in the simulation proposed by this study. At the beginning of the simulation, the environment is loaded by the simulator and robots are scattered randomly.

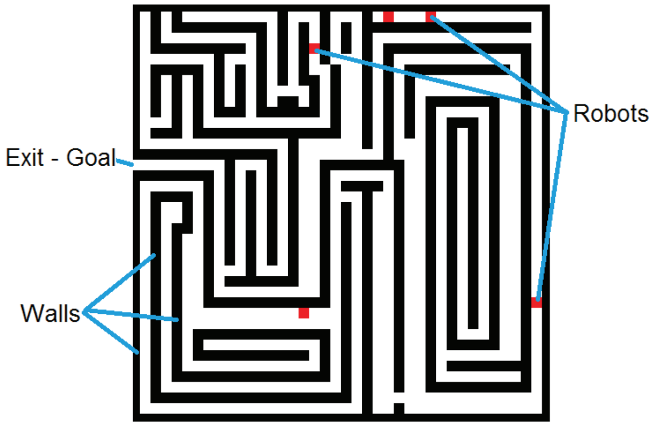
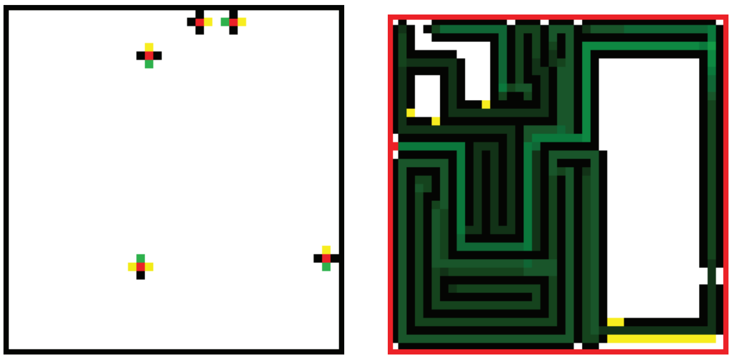


Fig. 7. Example of a Simulation Environment for Exploring Robots.

The map is a bi-dimensional representation implemented in Swarm using a structure known as Discrete2dImpl, a type of matrix where each position holds an integer. The meaning of such number is given by the following pattern: 0 - no information, 1 - identified wall, 2 - identified free pathway, 3 to 15 - path already taken (incremented in each step) and 50 - exit.
Figure 8(a) shows the map at the beginning of the simulation. Figure 8(b) shows the map at the end of the simulation, when robots have found the exit.



(a) Map at the Beginning of the Simulation.

(b) Map at the End of the Simulation.

Fig. 8. Map at Both Beginning (a) and End (b) of the Simulation.

**The robots**
The agent that represents a robot is capable of moving towards four directions (left, right, forward and backward) as well as detecting obstacles in any of these directions. On each movement, the agent evaluates its sensor's information as well as its last movement to determine its next step. First, the agent will check to which directions a movement is possible through the detection of walls and other obstacles. Such information is added to the map, which is updated with information regarding the location of walls and free spaces. In possession of that data, the agent will try to infer the following:

- Any position occupied by a wall is immediately discarded from the list of possible movements;
- Any position occupied by another agent is also immediately discarded from the list of possible movements;

The choice of a new position among those available will be based on the priorities below, according to the following order:

- Exit the environment;
- Movement's continuity, which means that the agent will prefer to follow the same direction as the one taken in its last movement, as changing the direction will most likely incur cost;
- Never visited locations;
- Least visited locations;

At the moment an agent occupies a new position, the related point in the map is incremented indicating that a new visit was made. In the next inference, such position will be despised by all agents.

**Search algorithm**
Considering the heuristics discussed so far, the search algorithm used might be classified as a blind search algorithm, which is fairly inefficient (Russell & Norvig, 2004). However, as the map doesn't provide broad range information (the robot is capable of detecting only whatever lies in its surroundings), it is impossible to make use of more sophisticated algorithms. On the other hand, exploring algorithms greatly favor the exploration and mapping of the environment.
Whenever enough information is gathered by one or many agents to link their position to the map's exit, the $A^*$ (read as Star) algorithm is used to get the shortest path to the exit. The $A^*$ search algorithm is one of the most common solutions to finding the shortest path in a graph (Russell & Norvig, 2004).
In the context of an exploring robot simulation, the graph used is obtained by discretizing the environment. Figure 9 shows how such discretization is performed and the resulting graph. The weight of the edges is 1, and the total weight of the path from the robot's current position to the exit is the same as the number of cells (or vertexes) visited during the walk.

4.1.2.2 Autonomous Robot Navigation Through Fuzzy Logic

The example presented in 4.1.2.1 shows the navigation in a discretized Cartesian environment where well-defined geometric basis are established. The autonomous navigation in real environments finds application from automated machine movement in factories to the creation of partially or completely automated robots used in outer space exploration. Such

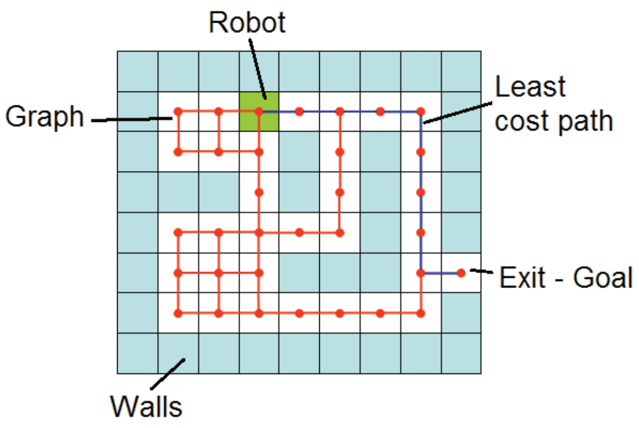


Fig. 9. Discretized Environment and the Resulting Graph.

broad range of applications also includes automating vehicles navigation, which is the focus of the study presented below. In this context, it is necessary to take into consideration the following:

- How inhospitable is the environment where the vehicle navigates;
- Which variables are involved;
- How to represent such variables and how they influence the movement.

The use of Fuzzy Logic offers a better interpretation of unpredictable environments, where the knowledge of the environment is required on each step for making as much of a natural decision as possible. This type of decision involves concepts like close, far away, fast, slow, slight turn, very slight turn, etc. Such concepts cannot be easily modeled by using classical logic, making Fuzzy Logic a very attractive choice.
Created in the 60's by Lotfi Zadeh, Fuzzy Logic is derived from Classical Logic. Zadeh considered that dealing with ambiguity (as the one caused by imprecise information) is an innate human ability. The application of Fuzzy Logic occurs in three stages:

- Fuzzification (to transform values into linguistic variables);
- Logical Inference (to apply logical rules);
- Defuzzification (to transform linguistic variables into values).

Fuzzification is the process of mapping a problem's input values into a function called pertinence function, which represents the parameter's variation and its linguistic counterpart. For instance, if 1.60 meters is considered short and 1.80 meters is considered tall, 1.70 meters might be considered as in-between (not short nor tall, and at the same time kind of short and kind of tall). On the other hand, Defuzzification is the inverse process. The result of logical inference is a value that needs to be translated to a linguistic value. For instance, is a 30-year-old a young, adult or old person?
Using Fuzzy Logic to solve the autonomous navigation problem allows a better modeling of the system's rules, as the ambiguity found when determining if something is far away, close, moving fast or moving slow is much better described by the tools offered by Fuzzy Logic.

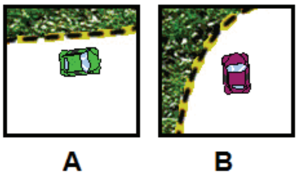
traffic regulation. This is achieved by modeling the functions considering "near" in the left side as being "nearer" than in the right side.
Besides the sensor's reading, the vehicle's current velocity is also evaluated using a pertinence function. After the fuzzification of these variables, the inference process occurs following the set of rules shown in Table 2.

| RULE | IF | | | | | | THEN | |
|---|---|---|---|---|---|---|---|---|
| | LEFT | LEFT CORNER | FRONT | RIGHT CORNER | RIGHT | SPEED | ANGLE | ACCELERATE |
| 1 | NEAR | | | | | | RIGHT | KEEP |
| 2 | | NEAR | | | | | STRONG RIGHT | KEEP |
| 3 | | | | | NEAR | | LEFT | KEEP |
| 4 | | | | NEAR | | | STRONG LEFT | REDUCE |
| 5 | | | | MEAN OR FAR | MEAN OR FAR | | RIGHT | KEEP |
| 6 | | | | FAR | FAR | | STRONG RIGHT | KEEP |
| 7 | | | FAR | | | SLOW OR MEAN | KEEP | ACCELERATE |
| 8 | | NEAR OR FAR | NEAR OR FAR | MEAN OR FAR | | MEAN OR QUICK | KEEP | REDUCE |

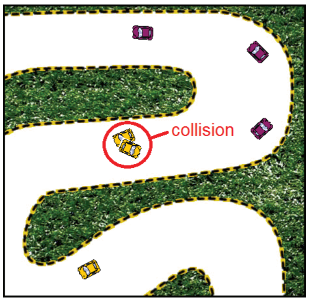
Table 2. Table of Rules Used in the Model.

The idea of using simple rules and in reduced number also favor the observation of emergent behavior. These rules are applied in each simulation step and the results are submitted to the defuzzification process.

**Implementation of the simulation**
Figure 11 shows the simulation running, where vehicles are seen navigating through an unknown route. The expected behavior is observed, as vehicles reduce their velocity during turns while keeping the distance to vehicles navigating in front of them or in the opposite direction.
An example of association between the rules and the agent's behavior is shown in Figure 11(a). Figure 11(a)-A shows an example of Rule 1 (see Table 2) executed as a result of the vehicle being near the left side of the road without a velocity's reduction needed. On the other hand, Figure 11(a)-B shows a hazardous situation that triggers Rule 2, which in turn changes the vehicles' trajectory and reduces its velocity drastically.



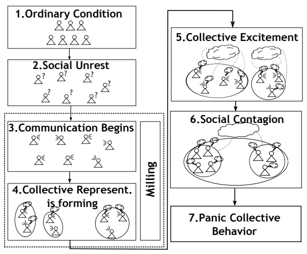
(a) Situations Associated to the Triggering of Rules.



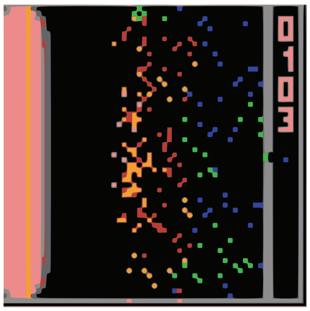
(b) Occurrence of Accidents During the Simulation.

Fig. 11. Implementation of the Simulation - Situations Associated to the Triggering of Rules and Occurrence of Accidents.

The velocity range that the vehicle operates on is another parameter that can be informed during the creation of a new agent. In one simulation instance, some of the agents were instructed to navigate in above-average velocity. Figure 11(b) show the execution of such



(a) Possible Steps in an Interactionist Approach.



(b) A Panic Simulation Screenshot.

Fig. 12. Panic in Crowds - Model and Simulation.

When the individuals share a similar representation of the current situation, they choose a line of action and they act in order to re-establish the previous condition or to save themselves from the imminent danger, in a typical collective panic behavior. These steps are linear just for didactical purposes, since the individuals may jump or redo some of these steps when they see fit (dos Santos França et al., 2009).

The transition from a theoretical model to a computation model could be a challenge. However, if the transition occurs, the implemented model might be used to analyze the social phenomena in a privileged seat. The data, behaviors and communications shown during the simulation can be compared to the theory and even validate it. Figure 12(b) shows a panic simulation screenshot.

#### 4.1.3.2 An Agent-Based Model for the spread of the Dengue Fever

The dengue fever is today the most spread arbovirosis in Brazil. Transmitted only by the female *Aedes aegypti* mosquito, it reaches its peak during the hot and humid Brazilian summer season. While there are many approaches to analyze the spread of the dengue fever, most of them focus on developing a mathematical model to represent that process. One disadvantage of such approach is to neglect the importance of micro-level behavior, focusing instead on the macro-level aspects of the system.

(Jacintho et al., 2010) developed an agent based simulation model for the spread of the Dengue Fever in the city of Rio de Janeiro, Brazil. Such model achieved similar results to the models currently being used, with the advantage of using just one set of agents and their interactions. The virus is transmitted to mosquitoes when they feed on the blood of a person already infected with the dengue virus. After an incubation period of eight (8) to twelve (12) days, the mosquito is then ready to propagate the disease. In humans, the incubation period might last from three (3) to fifteen (15) days, and symptoms are noticeable only after this period. Most importantly, there is no transmission through direct patient contact (including secretions) with a healthy person. The virus is not transmitted through water or food as well. To better understand and simulate the features observed in the real world, a transposition was made in order to build a model to be executed in a controlled environment. Rules were established for building a model as close to reality as possible, according to the scope of the project.

Below, a description of the simulation model based on (Otero et al., 2005; Santos et al., 2009) is presented, as well as the behavioral rules transposed to the computational model implemented in the Swarm platform.

**(A) Mosquito Agent Behavior**

As in the real world, this agent is modeled to display four (04) distinct stages: egg, larva, pupa and the land form, which corresponds to the adult mosquito. During simulation, each stage is represented internally in the mosquito agent, with no graphical/visual representation being used to differentiate distinct stages. The mosquito agent evolves according to the simulation progress and its behavior is internally adjusted according to its current life cycle stage.

**(A.1) Egg Agent Behavior**

Egg agents cannot move or feed and have an ideal temperature higher than 20 ºC, with an ideal humidity higher than 70%. Their outbreak will normally take place in about three (03) days.

**(A.2) Larva Agent Behavior**

These agents move only within their birthplace water spot, feeding on microorganisms and on their own egg remains. Their ideal temperature is between 25 ºC and 29 ºC, and their ideal humidity is higher than 70%. Under such conditions, this stage will take between three (03) to five (05) days to complete.

**(A.3) Pupa Agent Behavior**

Just like eggs, pupa agents cannot move nor feed. Their ideal temperature and humidity is around 20 ºC e 70% respectively, and they will have an 83% chance to become adult mosquitoes within three (03) days approximately.

**(A.4) Adult Mosquito Agent Behavior**

In this stage, agents are able to move freely through the environment up to 100m from their birthplace. Only females are capable of transmitting the disease, and that rarely happens at temperatures below 16 ºC, normally taking place under temperatures above 20 ºC. The mosquitoes proliferate at an estimated temperature between 16 ºC and 29 ºC, and have an average egg positivity of four (04) during their lifetime. Females will lay about 300 eggs on clean water with a 40% survival rate and 60% chance of being capable of transmitting the disease, i.e., other females. That means about 72 eggs will be considered in the simulation.
The mosquitoes can be killed by either exterminator agents or traps in the environment. They have an incubation period of about 8 to 11 days, by the time at which they become infectious and remain so for the rest of their life. Each infected female mosquito can propagate the disease to healthy humans by only a simple bite.

**(B) Human Agent Behavior**

As in the real world, this agent represents a human being, which might or might not become infected by the disease. Humans can move freely through the environment. After being bitten, it takes three (03) to six (06) days for the symptoms to become apparent. The Dengue Fever might last from three (03) to fifteen (15) days, with an average of five (05) to six (06) days. After being infected, the human agent can transmit the virus to others non-infected mosquitoes by blood contact during a mosquito's bite. This can occur one day before the first symptom appears and continues up to the last day of the disease. The death rates on multiple infections (also called the hemorrhagic dengue) are: 0.5% when infected twice; 10% when infected three times; 15% when infected four times and 25% when infected more than four times.

**(C) Exterminator Agent Behavior**

The exterminator agent moves freely through the environment based on the mosquitoes' gradient, attracted by areas of high density of mosquitoes in the map. This represents the public health organizations that map and notify all risk areas when planning control actions. Their role in the simulation exterminate adult mosquito agents.

**(D) The Environment**

The environment is not modeled as an agent by itself, but influences the agents' behaviors. Environmental factors such as temperature, food rate (probability of finding food) and humidity are globally defined as average values for the entire simulation, simplifying the simulation model and allowing the study of scenarios with different average values. There will be two states presented in the scenario: clean water and trap. Clean water servers as the place where mosquitoes will lay their eggs. Traps, on the other hand, are placed by exterminators to eliminate mosquitoes.

The conceptual model is transposed to a computational model that was later implemented in the Swarm simulation platform.

The environment interacts with all agents offering food for the mosquitoes, water for their reproduction and traps with substances to inhibit their proliferation. The results of such interactions between agents and the environment can be visualized in a 2D raster provided by the Swarm platform. According to Dantas et al. (Dantas et al., 2007) and Dibo et al.(Dibo et al., 2008), meteorological aspects such as temperature, humidity and precipitation can be used as predictors for Dengue incidence. In that sense, evaluating different climatic seasons allows a better understanding of the spread of the disease.

In this work, a tropical wet and dry climate region (Aw) is considered, according to the Köppen-Geiger climate classification (McKnight & Hess, 2000; Peel et al., 2007), as this is the predominant climate for most of Brazil. The weather in Brazil is characterized by high average annual temperatures and by a pluviometric regime that separates two distinct seasons: a rainy summer and a dry winter season.

**Scenario I: Winter season**

To simulate the Brazilian winter season, the Rio de Janeiro's climate information was used, with an average temperature of 18 ºC e average humidity of 45%. As the winter in Brazil is characterized by high dryness, 20 water spots were considered, with 5% of them set as traps. As the winter is a season with historically low dengue fever occurrence, only 10 exterminators were made available in this simulation scenario. The simulation started with 100 human agents, with 8% already infected with the disease, and 50 mosquitoes, with 60% already infected by the disease.

After 180 simulation cycles (60 days), no occurrences of hemorrhagic dengue (a human agent being infected more than once) were noticed, and the infection rate actually dropped to 7%. The number of mosquitoes in the environment also dropped from 50 to 30, with a 100% infection rate and with 111 mosquitoes in non-adult stages of their life cycles. Figure 13 (a) shows the simulation screen after 60 days. The color gradation shows the density of infected mosquitoes, and clearly displays few concentration spots. Figure 13 (b) shows the chart of mosquitoes in non-adult stages against adult mosquitoes. It was observed that even though there were many non-adult mosquitoes, several adults were unable to survive due to the harsh winter conditions (temperature, humidity and lack of water spots). Figure 13 (c) shows the