# TABSAOND: A technique for developing agent-based simulation apps and online tools with nondeterministic decisions

Iván García-Magariño*[a], Guillermo Palacios-Navarro[b], Raquel Lacuesta[a]

[a] *Department of Computer Science and Engineering of Systems, University of Zaragoza, Escuela Universitaria Politécnica de Teruel, c/ Ciudad Escolar s/n, 44003 Teruel, Spain*
[b] *Department of Electronic Engineering and Communications, University of Zaragoza, Escuela Universitaria Politécnica de Teruel, c/ Ciudad Escolar s/n, 44003 Teruel, Spain*

**Abstract**

Agent-based simulators (ABSs) have successfully allowed practitioners to estimate the outcomes of certain input circumstances in several domains. Although some techniques and processes provide hints about the construction of these systems, there are still some aspects that are not discussed in the literature. In this context, the current approach presents a technique for developing ABSs. This technique especially discusses the simulation of the decision-making processes of agents and the different ways of evolving their attributes in nondeterministic scenarios. The current approach includes a framework for developing ABSs as mobile apps and online tools. This work illustrates the current approach with a case study in the field of health and welfare. This case study has also been implemented with the most similar technique from the literature for comparing both techniques. The presented technique improved the simulated outcomes in terms of similarity with the real ones, as proved with several statistical tests. The obtained ABS is more efficient and reliable for large amounts of agents (e.g. 10,000 - 400,000 agents). The development time was also lower. Both the framework and the implementation of the case study are freely distributed as open-source to ensure the reproducibility of the experiments and to assist practitioners in applying the current approach.

*Key words:* agent-based simulation, multi-agent system, agent-based modeling, app, mobile application

## 1. Introduction

Agent-based simulators (ABSs) are multi-agent systems (MASs) that are specifically aimed at simulating certain facets in particular problem domains. Thus, ABSs are especially useful for simulating the evolution of autonomous entities, which usually have social interactions among each other. ABSs have been successfully applied in different problem domains such as for example (a) the validation of marketing strategies in Twitter [26], (b) the evaluation

*Corresponding author, Telephone: +34 978645348, Fax: +34 978618104
*Email addresses:* ivangmg@unizar.es (Iván García-Magariño), guillermo.palacios@unizar.es (Guillermo Palacios-Navarro ), lacuesta@unizar.es (Raquel Lacuesta )

of social choice techniques [27], and (c) the evolution of sociograms according to the classification of students [10]. Thanks to these ABSs and others, specialists can obtain useful information that is estimated from what-if scenarios different from the known cases.

The literature reveals the interest of the research community in constructing ABSs. Some works adapt existing agent-oriented software engineering (AOSE) methodologies for being particularized to simulations. For instance, Gómez-Sanz et al. [14] introduce an adaptation of the Ingenias Agent Framework (IAF) using the Ingenias methodology for the model-driven development of ABSs. In addition, ELDAMeth [8] allows rapid prototyping of ABSs with a visual modeling language, and generates the programming code for JADE (the Java Agent DEvelopment framework). Nevertheless, to the best of authors' knowledge, the existing works about development techniques for ABSs lack some aspects. First, these works do not analyze the different ways of evolving agent attributes and their repercussions. Second, none of these works explicitly provides a framework for developing ABSs as both mobile applications (also referred as apps) and online tools.

In this context, the current work presents a technique for developing ABSs that is especially aimed at covering the aforementioned gaps detected in the literature. In particular, the current work presents a Technique for developing Agent-Based Simulation Apps and Online tools with Nondeterministic Decisions (TABSAOND). It complements our previous proposal PEABS (a Process for developing Efficient ABSs) [9]. The main difference of the current work is that it now mainly focuses on the simulation model concerning aspects such as the different ways of simulating decisions and evolving agent attributes. In addition, ABSs are now developed as mobile apps and online tools, which can reach to a wider audience in a larger amount of circumstances (e.g. meditation courses in which practitioners do not bring a laptop).

The current technique has been compared with the most similar one from the literature by means of two different developments for obtaining the same ABS with the same specifications. This comparison takes into account similarity between simulated and real outcomes, response times and development time.

The remaining of the article is organized as follows. The next section introduces some related works, and highlights the gaps of the literature covered by the current approach. Section 3 indicates the main steps of TABSAOND, and describes its most relevant characteristics. It also discusses the alternatives of some of the steps, and presents the framework of the current approach. Section 4 evaluates the current approach by developing the same ABS with the current technique and the most similar one. Finally, section 5 mentions the conclusions and depicts future lines of research.

## 2. Related work

### *2.1. General techniques for developing ABSs*

There are several general techniques for developing ABSs. For instance, Gómez-Sanz et al. [14] present an adaptation of the Ingenias Agent Framework (IAF) for the model-driven development of ABSs. This adapted framework can be used for graphically modeling the simulation goals, inputs and outputs. The simulation model is usually implemented with certain code components that are associated with tasks. These code components are associated with the corresponding programming code. It also provides elements for modeling the simulation environment and applications that can generate certain events.

Moreover, ELDAMeth [8] is a simulation-based methodology for the development of distributed agent systems. In particular, it supports rapid prototyping with visual programming. It also validates the model, and automatically generates the programming code for the JADE platform. This methodology was illustrated with an example in information retrieval. ELDAMeth is mainly focused on the graphical modeling language, the code generation, and its integration with other AOSE methodologies.

Furthermore, Molesini et al. [21] discuss the integration of simulation in the existing AOSE methodologies. More concretely, they use the SODA methodology for illustrating their approach. Their proposed process has five steps: modeling, pointwise simulation, exact verification, approximate verification and tuning. They use an activity diagram for indicating the order of these steps and the possible iterations. Basically, they focus on the development of ABSs and the comparison of their results with real ones. In case of failing the comparison, they propose either just tuning some internal variables or coming back to the beginning for modifying the model of the corresponding ABS. They also propose some improvements in the metamodel for enhancing the expressivity of the agent-based modeling languages for simulations.

Ronald et al. [24] propose a model for constructing ABSs based on both spatial and social information. For instance, they show the utility of modeling social aspects in the simulation of travels. Their approach is mainly aimed at integrating the information of social networks in physical space represented as a map. In this way, their approach can efficiently model simulations about face-to-face social activities. In addition, Hall and Kirsi [17] introduce a general development process for defining and implementing agent-based models. They mainly focus on the acquisition of domain-specific knowledge from experts who may not have any technological background. More concretely, they propose certain ways of expressing this knowledge with an intuitive use of some kinds of diagrams. Their approach takes spatial and social information into consideration. They exemplify their approach with an ABS about urban development including information such as houses with different types and parking places. Seven experts agreed that this approach facilitated the understanding of the agent-based model.

Bouanan et al. [2] present a general mechanism for modeling and simulating the spread of messages considering the interactions. Their approach considers the attributes of agents. They distinguish some of these as variables as these vary during the simulation. Examples of these are the opinions, satisfaction degrees and interests. In their simulations, these variables influence the way some messages are spread. Their approach use DEVS and Cell-DEVS tools. They illustrate their approach with a case study that simulates the influences on opinions. They provide a specific formula for evolving this variable attribute, but they do not discuss other alternatives for being applied generally.

Furthermore, Caballero et al. [3] introduce a general technique for developing ABSs with special emphasis in the cognitive processes of agents. For this purpose, they integrate the MASON and Jason environments. In this way, their approach simultaneously allows practitioners to (a) define the cognitive processes of agents with a specific language for behavioral rules, and to (b) use a graphical simulation environment for configuring and displaying simulations with a graphical interface. Their approach use both social interactions and spatial information. However, they do not discuss different ways of evolving agent attributes based on nondeterministic decisions.

Therefore, all these works tackle the same problem as the current approach, which is the development of ABSs in general. Nevertheless, none of these works provides guidelines about different alternatives for evolving the internal attributes of agents with nondeterministic decisions. In addition, none of these works supports the development of ABSs as apps or online tools.

### *2.2. Ad hoc techniques for constructing ABSs*

Some works use ad hoc techniques for developing ABSs in specific problem domains. For instance, Auld et al. [1] present a modeling and software development kit for constructing ABSs in the context of simulating travel demands and network operations. This development kit is especially aimed at achieving a high execution performance. In particular, their approach has a repository of open-source transportation algorithms. They focus on an efficient simulation-oriented memory allocator, and introduce a parallel engine for discrete events. In addition, they define two primary agents respectively related with people and planning of activities. There are also different sub-agents of these agents. They define composed simulation models by means of some diagrams that relate data sources, models and results.

Moreover, Serrano and Iglesias [26] present a method for building ABSs for simulating and validating marketing strategies through Twitter. They review the existing ABSs for rumor spreading comparing their main features. They propose a novel spread model based on two different rumor datasets. They also propose new strategies to control malicious gossips. They introduce a free and open-source simulator constructed with their method. Their method uses UML activity diagrams to determine diffusion models. These diagrams use properties of agents in the conditional bifurcations, and the agents can communicate with their neighbors influencing their states. These neighbor agents are treated recursively with the same diffusion model.

Resta [23] presents an ABS driven by different self-organizing maps. She developed an environment where the nodes of these maps are managed by agents. These agents simulate economic systems and their evolution over the time. She implicitly indicates the process that she followed to obtain the ABS. More concretely, first she modeled the individual level, and then she modeled the aggregate level. Finally, she managed the relations between the individual and aggregate levels. She mainly applied the evolution of the simulations with two different types of neurons in the corresponding neural network.

On the whole, all these works are more aimed at solving specific problem domains rather than proposing general development techniques for ABSs. In particular, none of these works provides a general mechanism for conforming simulation models with different alternatives for evolving agent attributes.

### *2.3. Techniques for developing MASs for mobile devices*

This section discusses several relevant techniques for developing MASs that can be executed from mobile devices. One of these techniques is general while the others are ad hoc for different domains.

To begin with, Zambonelli et al. [28] presented a general mechanism for modeling MASs that implement pervasive computing systems. Their approach is based on several nature-inspired coordination models. In fact, these models are taken from stigmergy, chemical coordination, physical coordination and biochemical coordination. Their approach is called SAPERE, and integrates an architecture with multiple users. It includes a middleware that supports the development of pervasive systems in mobile applications in devices such as smartphones and tablets. The coordination models are represented with eco-laws. Their approach is mainly aimed at developing MASs that coordinate several activities with some users through a distributed system that use several mobile devices. However, this approach is not aimed at performing simulations. Instead, this approach focuses on the problem of coordinating users. Thus, the scope of the systems developed with this approach is different from the ones developed with the current approach.

Moreover, Lin and Lin [20] developed a MAS that coordinates several users through an app. User can change their goals, by scheduling or re-scheduling their tasks. In particular, it solves the mobile user coordination problem. In addition, a model-driven approach was defined for developing ambient assisted-living MASs customized for Parkinson patients [12]. This approach implicitly defined the process, and introduced a language specifically designed for patients that suffer from Parkinson's Disease (PD). The workproducts of this implicit process are apps. In this line of research, PHAT [4] framework creates virtual living labs that reproduce realistic conditions of an application inside certain hardware that can run the Android operative system. Their approach runs an ABS that determines the concrete situations with the SociAALML modeling language. This language has been tested in the context of PD patients. Thus, these works described ad hoc techniques, each of which was only intended for one specific domain. These works do not provide general guidelines for developing ABSs. In addition, none of these works discusses different ways of evolving internal attributes of agents.

### *2.4. Comparison with survey questions*

In the spirit of the systematic review methods, the current work formulates several research questions for further analyzing the introduced related works. Some of these questions use the term MASs for covering more systems than strictly ABSs. These questions follow:

- Q1. Has this work used a technique for developing ABSs regardless whether the technique is ad hoc for a specific domain?
- Q2. Does this work explicitly define a general technique for developing different MASs?
- Q3. Is this work illustrated with at least an ABS?
- Q4. Does this work compare simulated and real results for an ABS as a case study?
- Q5. Does this work apply a statistical test for comparing real and simulated results?
- Q6. Does this work provide an open and free ABS as a case study?
- Q7. Does this work show the evolution of a metric in any simulation?
- Q8. Does this work discuss some different ways of evolving agent attributes?
- Q9. Has this work implemented a MAS as an app?
- Q10. Has this work implemented a MAS as an online tool?

Table 1 answers the questions for the most relevant related works. As one can observe, none of the analyzed works explicitly discusses several alternatives for evolving agent attributes. Thus, to the best of authors' knowledge, the current work is the first one that analyzes different ways of evolving agent attributes when defining a technique for developing ABSs.

| Work | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Gómez Sanz et al. | ✓ | ✓ | ✓ | ✓ |  | ✓ |  |  |  |  |
| ELDAMeth | ✓ | ✓ | ✓ |  |  |  | ✓ |  |  |  |
| Molesini et al. | ✓ | ✓ |  |  |  |  |  |  |  |  |
| Ronald et al. | ✓ | ✓ | ✓ |  |  |  |  |  |  |  |
| Hall and Kirsi | ✓ | ✓ | ✓ |  |  |  | ✓ |  |  |  |
| Bouanan et al. | ✓ | ✓ | ✓ |  |  |  |  |  |  |  |
| Caballero et al. | ✓ | ✓ | ✓ |  |  |  | ✓ |  |  |  |
| Auld et al. | ✓ |  | ✓ | ✓ |  |  | ✓ |  |  | ✓ |
| Serrano and Iglesias | ✓ |  | ✓ | ✓ |  | ✓ | ✓ |  |  |  |
| Resta | ✓ |  | ✓ | ✓ |  |  | ✓ |  |  |  |
| Zambonelli et al. |  | ✓ |  |  |  |  |  |  | ✓ |  |
| Lin and Lin | ✓ |  | ✓ | ✓ | ✓ |  | ✓ |  | ✓ |  |
| MASs for Parkinson |  |  |  |  |  |  |  |  | ✓ |  |
| PHAT | ✓ |  | ✓ |  |  | ✓ |  |  | ✓ |  |

Table 1: Comparison of related works with survey questions. Check mark: yes, empty space: no.

Moreover, considering only the works that explicitly define general techniques for developing different MASs, none of these contemplates providing a MAS as an online tool. In addition, most of these works neither implement their systems as apps. The only exception is the technique proposed by Zambonelli et al. [28], but it has not been specifically applied for constructing ABSs. Therefore, the current approach is the first general technique for specifically developing ABSs as apps and online tools.

Furthermore, most of these works about general development techniques neither compare real and simulated results for evaluating the technique nor apply a statistical test for this comparison. In particular, the only found exception is the general technique of Gómez et al. [14], which compared the results without applying a statistical test. In fact, this last technique has been considered as the most similar one to the current approach, so it has been used in the experimental comparison of the current work.

## 3. TABSAOND

TABSAOND allows practitioners to follow certain steps for modeling and constructing ABSs for particular problem domains. This technique mainly focuses on the decision-making process of agents specifically considering the nondeterministic decisions. In this context, it discusses different ways of evolving agent attributes. In addition, this technique is aimed at developing the simulators as mobile applications and online tools, so that these simulators can be available to a wider audience of users. The next subsection presents the main steps of this technique, and introduces the remaining subsections of this section.

### *3.1. Main steps*

The main steps of the current technique are presented in Figure 1. As one can observe, practitioners can come back from some steps, following an iterative approach. These main steps follow:

- *Definition of the agent types*: The agents of a system must be classified according to their internal and external behaviors. Agent types represent different agent roles. There can be one or several agents of each type.
- *Selection of the agent attributes*: Determine which features are relevant in each agent type as agent attributes. The current approach recommends that each agent attribute is represented with a real numeric value in a limited interval. In the simulators with spatial information, the locations of agents are also represented as agent attributes.
- *Modeling the social interactions*: In this step, the designer determines the social interactions indicating the agent types of the senders and the receivers. They also establish the influences of the interactions on the agent attributes. In particular, they indicate which information is transferred. In this step, the designer determines

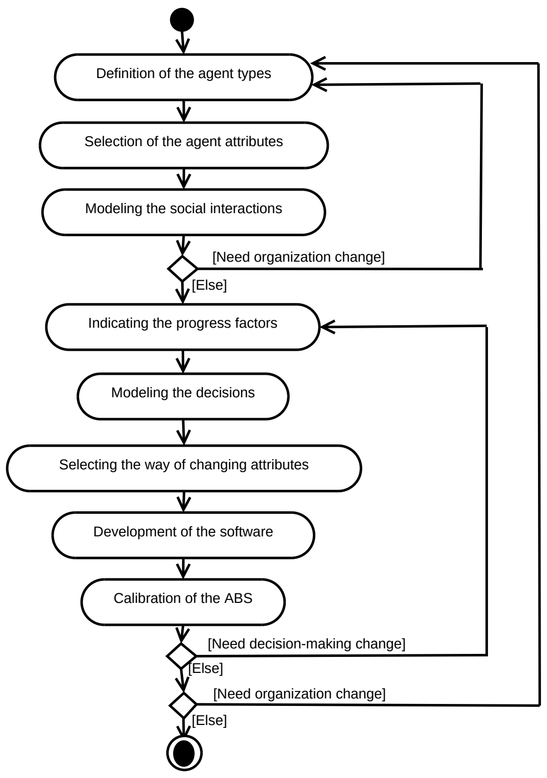

Definition of the agent types
Selection of the agent attributes
Modeling the social interactions
[Need organization change]
[Else]
Indicating the progress factors
Modeling the decisions
Selecting the way of changing attributes
Development of the software
Calibration of the ABS
[Need decision-making change]
[Else]
[Need organization change]
[Else]

Figure 1: Main steps of TABSAOND

whether the current design represents all the social interactions and transferred information. In some cases, the designer may need to change the organization of the ABS. In these cases, the designer comes back to the first step for redefining the agent types.

- *Indicating the progress factors*: This step simulates each ratio of agents that actually perform an action, simulate a repercussion or take a decision. This aspect is recommended to be simulated with a random number compared to the probability that represents the ratio, in a similar way to the next step.
- *Modeling the decisions*: The decisions are usually taken considering certain facts of the environments. In most cases, individuals do not decide always the same from the same input facts. This is advised to be simulated with probabilities. The decisions are modeled with the decision-making process presented in section 3.2. In addition, there are several ways of determining these probabilities, as it will be described later in section 3.3.
- *Selecting the way of changing attributes*: The agent attributes usually vary concerning certain events or actions. The designer must decide the proper mechanisms for simulating these variations, which can usually be either incremental or decremental. Section 3.4 discusses some alternatives for these variations. In the case of simulations with spatial information, these variations can be determined with a direction and a distance.
- *Development of the software*: The agents are recommended to be implemented with a particular framework. In this way, developers only need to implement the functioning of each agent type considering the update of its mental state (represented with attributes) and their social communications. The framework usually runs the whole simulation considering these implementations and certain numbers of instances for each agent type. This framework supports the current technique, and is presented in section 3.8. This framework is aimed at implementing ABSs as apps and/or online tools.
- *Calibration of the ABS*: The simulated results must be compared with real results. The developers normally need to tune certain parameters of the decisions or the variations, in order to achieve that the simulated outcomes are

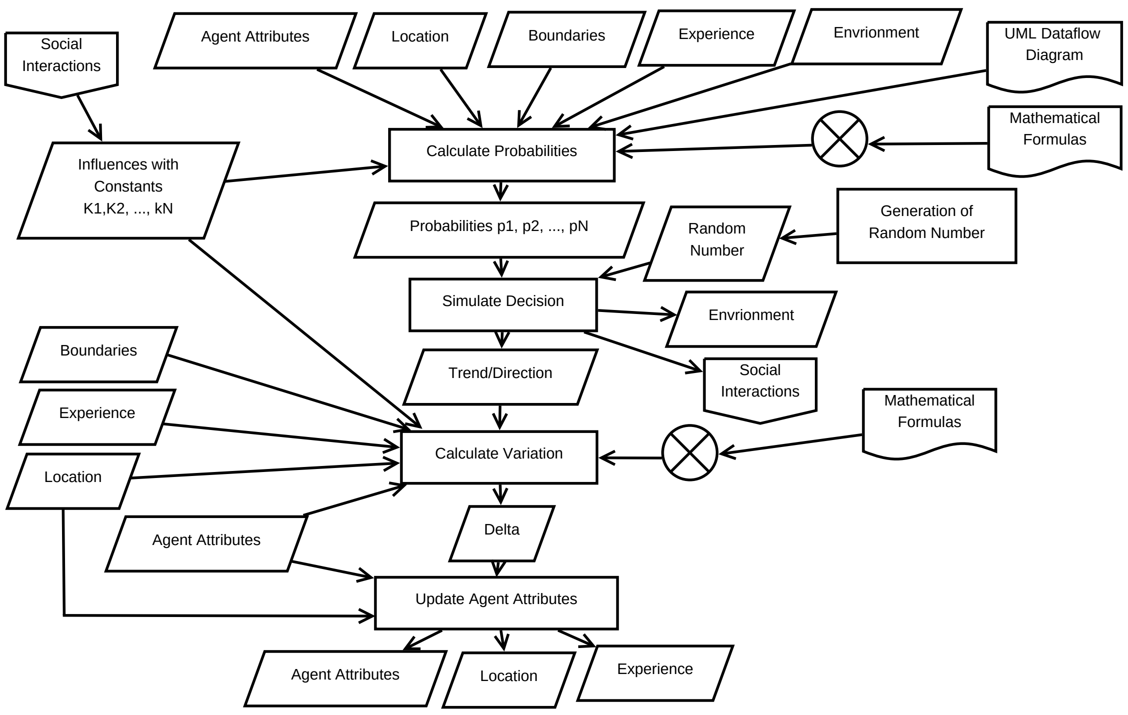


Figure 2: Dataflow diagram of the decision-making process in the mental states of agents

similar to the real ones. If the developers cannot properly calibrate the ABS, they can change the decision-making process by coming back to the step about the progress factor. If this is not enough, they can also come back to the first step for redefining the agent types, their features and their social interactions.

The next subsection proposes a generic decision-making process for agents and the way for particularizing it for specific domains. Section 3.3 determines different ways of determining the probabilities for simulating the decisions. Section 3.4 proposes several ways of calculating the variations of agent attributes according to the decisions. Section 3.5 discusses several possible combinations of (a) mechanisms for obtaining probabilities and (b) formulas for calculating variations. Section 3.6 discusses certain aspects of social interactions, and section 3.7 provides some hints for simulations with spatial information. Finally, section 3.8 presents the framework that supports the development of ABSs with the current approach.

### *3.2. Decision-making process*

Figure 2 shows the dataflow diagram with a general schema of the decision-making process of each agent in the current approach. In this approach, the decisions are taken in a nondeterministic way by means of probabilities. These probabilities condition all the decisions of a particular kind taken by a certain type of agents in different iterations.

One of the cornerstones of the current approach is the calculation of these probabilities, since these are used for simulating the decisions. The calculation of probabilities can receive input from agent attributes, their locations, the interval boundaries of agent attributes, the experience from previous iterations and other aspects from the environment. This calculation of probabilities can also be influenced by some information transferred from certain social interactions. Regarding the application domain, these calculations can be defined with UML dataflow diagrams (or activity diagrams) and/or mathematical formulas. The mathematical formulas can be combined with validated probability models [25]. Section 3.3 further discusses the calculation of these probabilities.

After selecting the probabilities, each decision is simulated by the generation of a random number, and its comparison with the accumulated values of these probabilities (see the explicit mechanism in section 3.3). In this manner, each choice has the corresponding probability of being taken. The choice of a decision can alter the environment or trigger one or several social interactions. Besides these effects, the selection of a choice usually influences the trend in an agent attribute or several ones. Each trend can be either positive, negative or neutral. In case of simulations with spatial information, the decision can also output a direction, expressed either with an angle $\alpha$ or with a vector of several dimensions.

Another cornerstone of the current approach is the calculation of the variation of each corresponding agent attribute from the trend or direction. The calculation of these variations can be defined as a combination of certain mathematical formulas as further discussed in section 3.4. These formulas can receive information from the constants transferred in social interactions, the agent attributes, their boundaries, the previous experience and the location.

Finally the agent attributes and/or the location are updated with the outputted variation (referred as "Delta"). The experience can also be recorded to be used in the decisions of later simulation iterations.

## 3.3. *Probabilities of decisions*

The decisions of agents can usually be simulated with certain probabilities. In particular, a decision of two options can be simulated by the generation of a random number $r$, and then comparing it with a certain probability $p$ assigned to the positive option. In this case, the decision $d$ can be simulated with the following equation:

$$d = \begin{cases} \text{yes}, & \text{if } r \leq p \\ \text{no}, & \text{otherwise} \end{cases}$$

This can be generalized for a decision $d'$ with more than two options with the following equation:

$$d' = \begin{cases} o_1, & \text{if } r \leq p_1 \\ o_2, & \text{if } p_1 < r \leq p_2 \\ o_i, & \text{if } p_{i-1} < r \leq p_i \\ o_N, & \text{otherwise} \end{cases}$$

where $p_1$,$p_2$, ... $p_{N-1}$ are respectively the probabilities of options $o_1$, $o_2$, ...,$o_{N-1}$, and the probability of $o_N$ is one minus the sum of all the other probabilities.

The current approach proposes several mechanisms for calculating the probabilities. Each of these mechanisms can be configured in different ways regarding the adjustment of some of its internal constant values. These mechanisms follow:

- *Constant probability*: In this mechanism, the decisions of each kind are simulated with a fixed probability. This mechanism is mainly recommended to be used in the step about the progress factor. This mechanism can also be useful in the next step about modeling decisions. For instance, the constant value can be transferred from an interaction. In addition, in some cases, the decisions can be categorized by some types of events. In these cases, one can define a list of different probability constants for the different types. This mechanism can also be useful for combining it with the other proposed mechanism for calculating probabilities and/or some mechanisms for obtaining non-constant variations in agent attributes.

- *Consider the integration of a normal distribution*: In this case, the designer must properly establish the way of setting the mean and the standard deviation of the normal distribution. In each iteration, the probability is obtained from the integration of the normal distribution between the current attribute value and one of the attribute interval limits.

Regarding the second option, a normal distribution is defined with the following equation:

$$\Phi(x) = f(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{1}$$

where $\mu$ is the mean and $\sigma$ is the standard deviation (SD).

In this option, the probability is reduced when reaching unusual values of the agent attribute. In particular, for increasing trends, it calculates the integral of the normal distribution $\Phi(x)$ from the current value to the upper limit.

This probability is multiplied by a constant called $I_i$, which is recommended to be set to two or lower values. Oppositely, the current approach uses the integral of $\Phi(x)$ from the lower limit to the current agent attribute for decreasing trends. This probability is multiplied by the $I_d$ constant which is recommended to have any positive value up to two. In particular for increasing trends, the probability is calculated in the following manner:

$$p_{ii} = min\left(1,\ \ I_i \cdot \int_c^{1_{max}} \Phi(x)\,\mathrm{d}x\right) \tag{2}$$

where c is the current agent attribute, and $l_{max}$ is its upper limit.

In addition, for decreasing trends, the probability is obtained with the following formula:

$$p_{id} = min\left(1,\ \ I_d \cdot \int_{l_{min}}^{c} \Phi(x)\,\mathrm{d}x\right) \tag{3}$$

where $l_{min}$ is the lower limit.

The way of establishing the mean $\mu$ and the SD $\sigma$ in the normal distribution $\Phi(x)$ influences the way the decision probabilities evolve. This $\mu$ value represents the common average in the corresponding domain. In particular, the current approach advises developers to automatically set $\mu$ to the average of the corresponding attribute of all the agents at the beginning of the simulation, because this value is usually a representative average in the domain. In this way, the simulation uses the previous experience as it is conditioned with an initial value of the simulation. The $\sigma$ value commonly influences in the final global variations of the whole system. Although some value can be initially assigned, $\sigma$ is usually adjusted after in the final calibration step of the present technique, in order to obtain similarity between simulated and real outcomes.

A simulation can combine both equations about the integral of a normal distribution for respectively increasing and decreasing trends. In this case, the current work recommends that the sum of the $I_i$ and $I_d$ constants does not surpasses two. In this manner, the system can simulate a decision with three options for respectively increasing, decreasing and neutral trends. The last option is selected when the others are not selected. The values of the $I_i$ and $I_d$ constants are also usually tuned in the final calibration step.

Sometimes, a task is performed after two or more decisions take place together. In these cases, the decisions can usually be combined as a product of the probabilities, which is supported by a validated probability model [25].

After the probabilities are simulated, the decision output can be represented as an integer value that represents the selected option. For example, a possible range of values can be $\{-1, 0, 1\}$ for respectively determining decreasing, neutral and increasing trends.

### *3.4. Calculation of the variations*

In the calculation of the variations of agent properties, this technique advises practitioners to consider boundaries of the corresponding intervals. It is worth reminding that in general this work recommends establishing boundaries for the properties of all the agent types. There are two ways of considering boundaries in the current approach. First, developers can consider any function $f$ of evolving agent attributes, and then apply another function $f'$ that forces that any value outside the allowed limits is changed to nearest limit. The second way is to establish the function of attribute evolution considering the distances to the limits in order to avoid surpassing any limit. This section firstly introduces the first way of keeping agent attributes within some limits. Then, it presents some ways of evolving agent attributes. One of these considers the distance to the limits, so it can keep the attributes within the allowed intervals.

Regarding the first option about the attribute limits, if the variation of an attribute value makes this surpass any boundary, then this attribute is finally set to that boundary. This is represented with the following equation:

$$f'(x) = \begin{cases} l_{max}, & \text{if } f(x) \geq l_{max} \\ l_{min}, & \text{if } f(x) \leq l_{min} \\ f(x), & \text{otherwise} \end{cases}$$

where $l_{min}$ and $l_{max}$ are respectively the minimum and maximum limits, and $f(x)$ represents the function that updates the $x$ value of a particular agent attribute considering any variation.

If the function f is known to be monotonically increasing, f' can be simplified with the following equation:

$$f'(x) = min(f(x), l_{max}) \tag{4}$$

Likewise if the function f can be assumed to be monotonically decreasing, f' can be simplified with the following equation.

$$f'(x) = max(f(x), l_{min}) \tag{5}$$

In general, all the formulas of evolving agent attributes must consider the aforementioned mechanism for not surpassing limits, except the formula that considers the distance to the limits.

This section now describes the different formulas for evolving agent attributes from scratch. To begin with, the evolution function f(x) can be decomposed with the following equation that explicitly mentions the variation as $\Delta x$:

$$f(x) = x + \Delta x \tag{6}$$

One option is that $\Delta x$ is established to fixed values, depending certain surroundings facts of the environment. In order to illustrate the current approach, this paper uses the constants $D_i$ for increasing trends and $D_d$ for decreasing trends.

As another option, one can consider the boundaries in a function that calculates $\Delta x$. For instance, this can be achieved by calculating a $\Delta_k(x)$ variation as a portion of the distance to the approaching limit with the following equation:

$$\Delta_k(x) = \begin{cases} K_i * (l_{max} - x), & \text{if } d > 0 \\ -K_d * (x - l_{min}), & \text{if } d < 0 \\ 0, & \text{otherwise} \end{cases}$$

where $K_i$ and $K_d$ represents the ratio constants for respectively the increasing and decreasing trends, and $d$ is the decision.

Figure 3 shows the evolution of an agent attribute that always evolves with the variations calculated with this mechanism. The inferior and upper bounds are respectively set to -1 and 1 for this example and the remaining ones.

Another alternative for calculating the variation is to consider the position of the simulation iteration (i.e. a counter that increases in each iteration). This alternative is mainly aimed at reducing the fluctuations for reaching stability. Besides the constants $D_i$ and $D_d$, it also uses the constant $L_c$ referring for the long duration aspect and $L_e$ referring to the exponent of this aspect. From the $d$ decision, in this alternative the $\Delta_L(x)$ variation is calculated in the following way:

$$\Delta_L(x) = \begin{cases} D_i \cdot min(1, L_c/i)^{L_e} & \text{if } d > 0 \\ -D_d \cdot min(1, L_c/i)^{L_e}, & \text{if } d < 0 \\ 0, & \text{otherwise} \end{cases}$$

where $i$ refers to the position of the iteration.

In particular, $L_c$ is recommended to be set considering the number of iterations that are expected to be necessary for reaching stability in the specific domain. In addition, $L_e$ is suggested to be set to one or two for smooth losses of fluctuations, and is recommended to be set to higher values for quicker losses of fluctuations.

In this approach, some of the aforementioned alternatives can be combined to calculate the variations in a more appropriate way according to the specific domain. For example, one can calculate two possible increments in different ways like $\Delta_1$ and $\Delta_2$, and obtain either the minimum, the maximum or the average of these two values, respectively with the following equations:

$$\Delta_c x = min(\Delta_1, \Delta_2) \tag{7}$$

$$\Delta'_c x = max(\Delta_1, \Delta_2) \tag{8}$$

$$\Delta''_c x = (\Delta_1 + \Delta_2)/2 \tag{9}$$

In addition, similar equations could be applied for the combinations of more than two formulas. The minimum combination is recommended to be used in increasing trends in which one of the formulas increases more steeply

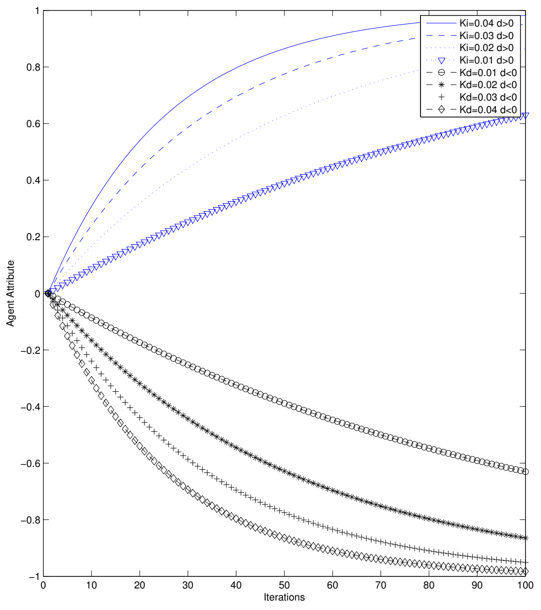


Figure 3: Evolution of an agent attribute being $\Delta_k(x)$ a portion of the distance to the approaching limit.

than appropriate in some iterations. Oppositely, the maximum combination is recommended to be applied in decreasing trends when one of the equations decreases more steeply than appropriate in some iterations. Notice that the maximum of several negative numbers is the one with the lowest absolute value. The average of several formulas is usually recommended for the designers that want to avoid inappropriate evolutions in the points that correspond to the intersection of the original formulas when using the minimum or maximum combinations.

### *3.5. Combination of decisions and calculations of variations*

Normally, in the construction of ABSs, developers combine mechanisms for taking decisions with particular manners of calculating variations of agent attributes. The goal of this section is to illustrate some useful combinations, so that developers can know the implications of these combinations. In this manner, they can select the combination that better fits each particular domain.

To begin with, Figure 4 shows an example of combining a decision with a constant probability with a variation $\Delta_k(x)$ calculated as a portion of the distance to the approaching limit. This chart shows different evolutions considering several values of the probability of the decision.

Moreover, one can apply decisions with more than two options including not only increasing trends but also decreasing ones, and combines this with a $\Delta_k(x)$ as a portion of the distance to the approaching limit. Figure 5 shows an example of this case. This figure represents the evolutions of an agent attribute with different probabilities for the decisions. The probability of increasing is denoted as $p_i$, while the probability of decreasing is denoted as $p_d$. The remaining implicit probability determines when the agent attribute maintains the same value. This combination can achieve different relative stable values regarding the different pairs of values for respectively $p_i$ and $p_d$. In particular, this example is shown with a large number of iterations (i.e. 10000 iterations) to show that the results of this combination reach relative stable solutions without converging to any of the interval limits.

The stability is achieved because there is both increasing and decreasing decisions. For instance, the increasing variations are lower when approaching the upper limit, while at the same time decreasing variations are larger since the attribute value is getting far from the lower limit. In this case, although the probability of increasing $p_i$ is greater than the probability of decreasing $p_d$, this difference is compensated with the difference of variations at some positive

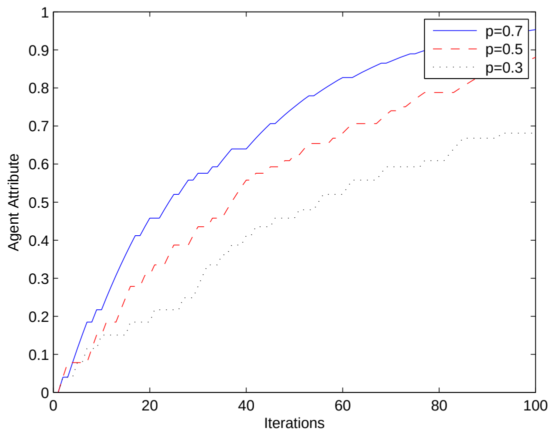


Figure 4: Evolution of an agent attribute combining a decision with a constant probability (referred as “p” in the chart) and a portion of the distance to the approaching limit. $K_d$ is set to 0.04 for all the evolutions of this chart.

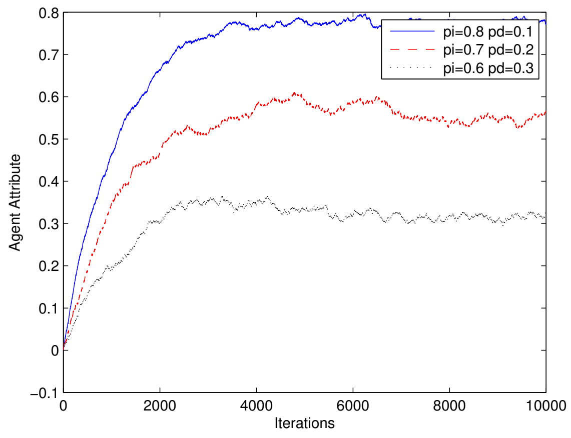
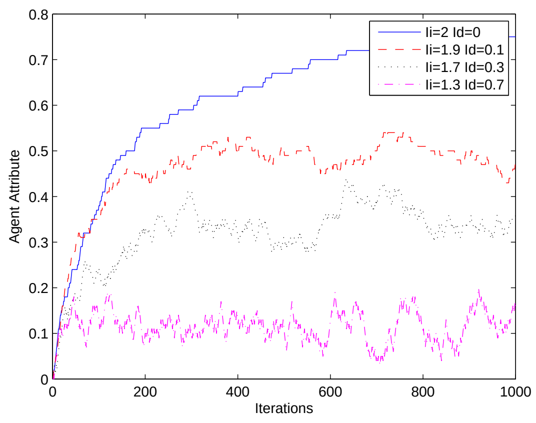
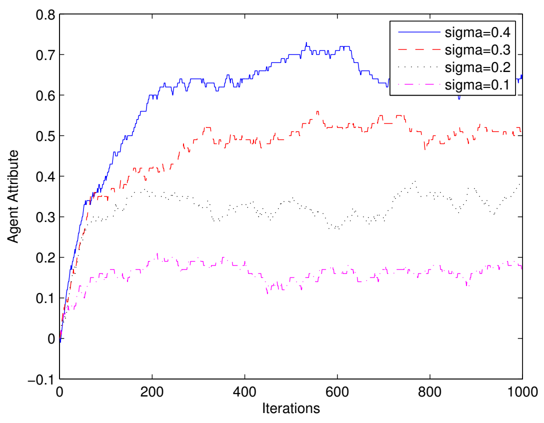


Figure 5: Evolution of an agent attribute combining a decision of three options with a portion of the distance to the approaching limit. $K_d$ and $K_i$ are both set to 0.001 for all the evolutions of this chart.



(a) Variation of $I_i$ and $I_d$



(b) Variation of $\sigma$

Figure 6: Combination of decisions based on the integral of a normal distribution and fixed variations.

value. In this way, the simulation usually reaches to an agent attribute value that tends to be the same, despite some slight fluctuations from the underlying nondeterministic random mechanism for simulating decisions.

Furthermore, another recommended combination is to use (a) the probabilities from the integration of a normal distribution for decisions and (b) fixed values $D_i$ and $D_d$ for the variations controlling that the limits are not surpassed. Figure 6 shows an example of this combination. In particular, Figure 6a uses the values $\mu = 0$, $\sigma = 0.3$ for the normal distribution and $D_i = 0.01$, $D_d = 0.01$ for the variations. The figure represents the results of different combinations of $I_i$ and $I_d$. This combination achieves relative stability, as one can observe in the evolution of 1000 iterations. This stability is reached because the probabilities of increasing and decreasing usually vary in opposite directions until these meet the same value. However, the fluctuations are high, because the variations are not reduced.

In addition, Figure 6b shows the same combination of decisions and variations. However, this chart uses different values of $\sigma$, the SD of the normal distribution used for the decisions, while the other values are fixed. In particular, this example uses the values $I_i = 1.9$ and $I_d = 0.1$. One can observe that the lower $\sigma$ is, the nearer to the mean of the normal distribution (in this example $\mu = 0$) the stability is reached.

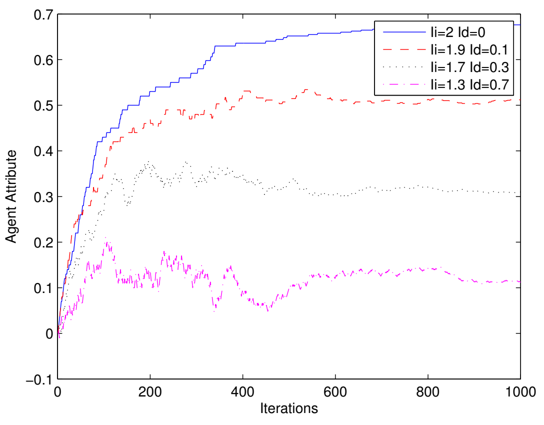
The decisions from the integral of the normal distribution can be usefully combined with the variations $\Delta_L(x)$ that are reduced when there are long durations. Figure 7 presents several examples for illustrating this combination. In particular, this figure represents simulations with the same values of $I_i$, $I_d$, $\mu$, $\sigma$, $D_i$ and $D_d$ as in the previous Figure 6a, but it now considers the variations $\Delta_L(x)$ based on the long durations. Both subfigures of Figure 7 use $L_c = 300$. The difference is that Figure 7a has applied $L_e = 2$, while Figure 7b shows the results when $L_e = 4$. As one can observe, the former reduces the fluctuations with a smoother transition, while the latter reduces its fluctuations more rapidly.

Considering the analyzed combinations, the last one achieves the lowest fluctuations in simulations with long durations. Thus, this last combination is probably the most appropriate choice when the simulation is intended to achieve stability with very low fluctuations.

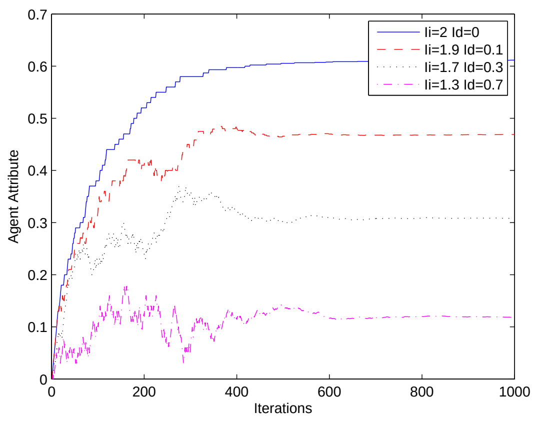
It is worth mentioning that another combination with low fluctuations was the one that used a distance portion to the approaching limit, when the attribute tends to be one of the attribute interval limits. However, the fluctuations are not so low when the stability is reached far from the limits. Hence, the current approach recommends using the variations based on long durations when the stability should be reached far from the limits. In particular, the higher the $L_e$ constant is, the quicker the fluctuations disappear.

### 3.6. *Discussions about social interactions*

The agents of ABSs usually communicate with each other conforming social interactions. Hence, in most ABSs these interactions may influence the evolution of the internal agent attributes. In the current technique, the social



(a) $L_e = 2$



(b) $L_e = 4$

Figure 7: Combination of decisions based on the integral and the variations considering the long durations, with $L_c = 300$.

interactions can be considered as inputs in the decisions. The decisions can also receive input from other aspects such as certain environmental information. Furthermore, the outputs of some decisions can imply that certain agents initiate some social interactions.

In social interactions, the influence of some agents on others can be modeled by the communication of certain constants of the previously introduced formulas. For instance, an agent $a_s$ sends a message to an agent $a_r$. Then, the receiver agent $a_r$ takes some decisions based on certain formulas for both (a) calculating the probabilities for simulating certain decisions and (b) obtaining the variations of some internal attributes. These formulas can use several constants $K_1, K_2, ..., K_N$. Some of these can be transferred in the communication from $a_s$ to $a_r$.

In social interactions, an important decision of each sender agent $a_s$ is the selection of the receiver agent $a_r$. Regarding the kind of communication, the receiver agent $a_r$ is usually selected from the existing ones of a specific type. There can be several agents of each type. An unbalanced selection mechanism can imply bias in the functioning of the whole system, especially when some agents are overloaded. This fact has been further discussed and proved in our previous work [15]. Thus, this technique recommends equiprobable distributions for selecting receiver agents from a particular type when possible.

Nevertheless, it is not always possible to select receiver agents with equiprobable choices for each type. In some cases, the agents are selected considering some of their properties. For example, the selection can consider the locations of agents. A common way is to select the nearest agent to the sender agent, only considering the agents of the corresponding type. This selection mechanism does not usually produce overloading patterns, for common heterogeneous location distributions of agents.

In other cases, agents are aimed at maximizing or minimizing some attribute of the receiver agent. If the selection mechanism is deterministic in all the sender agents (e.g. selecting the agent with the maximum or the minimum value in a particular attribute), then it will lead to an overloading pattern. The current technique recommends that this decision is selected with the equation for the decisions with more than two options introduced in section 3.3. The probabilities $p_1$, $p_2$, ..., $p_N$ can be established considering the values of the relevant attribute of the agent. In this way, although the agents with higher values of the attribute are more probable to be selected, the other agents can also be selected. The selection is still close to the reality, since the relevant attribute influences the distribution in comparison to the equiprobable decision. At the same time, it alleviates the possible negative effects of the overloading patterns.

Furthermore, the selection of the receiver agent could also be selected considering several agent attributes. In this case, the presented technique also recommends taking decisions based on probabilities considering the corresponding attributes. If possible, different sender agent types are recommended to use different weighted combinations of the relevant attributes, alleviating even more the negative effects of the overloading patterns [16].

## 3.7. *Discussions about spatial information*

In ABSs, the spatial information is used in several domains such as ecosystems, land manager and traffic simulations. In the current approach, the locations are represented as agent attributes, although normally these are recommended to be treated in a different way.

Locations can be considered as inputs and/or outputs of the decisions. Considering the inputs, the location can determine different aspects. For instance, it can be the initial spot for a path finding problem. It can also be used to determine the distances or relative positions from certain environmental elements such as the limits of the allowed space, certain obstacles of the environment, other agents representing any kind of individuals and the ball in some sport simulations. Regarding the outputs, the decisions can influence on the location of the corresponding agent. For instance, some common decisions are to move in certain directions.

Locations are recommended to be decomposed in its dimensions for being represented as agent attributes, e.g. $x$ and $y$ for two dimensions (2D). Besides the location point, the agents can also represent their orientation with an angle $\alpha$. In this manner, some of the presented formulas can be adapted for being used with a local polar coordinate system.

In each iteration, the decision can determine the agent orientation and whether the agent moves. In case of spaces with graphs, the direction of each agent usually follows a searched path. However, in ecosystem-like scenarios, the direction is not usually restricted to an edge of a given graph map, and directions are usually selected for getting to certain targets in the space and/or avoiding certain obstacles.

Considering the variation of the location, a variation distance $\Delta d$ can be calculated with some of the formulas previously introduced in section 3.4. This distance is added to the coordinates of a specific agent considering its orientation. For example, in the case of 2D, the new coordinates can be calculated with the following equations:

$$x = x + cos(\alpha) \cdot \Delta d \tag{10}$$

$$y = y + sin(\alpha) \cdot \Delta d \tag{11}$$

In a similar way, simulations can be modeled in three dimensions (3D), where the three coordinates will be updated with $\Delta d$ and a direction in 3D. This direction could be represented either with two angles or a vector of three components. This vector should be normalized for letting the velocity be determined by $\Delta d$.

Following the equations of the current technique, some agents may reach stable positions. For example, assume that an ABS simulates the mobility of the population of a continent. This system can simulate the fact that older people usually change their place of residence with shorter distances and less frequently. The simulator could use a similar function as the one previously introduced for long durations in section 3.4 for simulating the displacements. In particular, the distance $\Delta d$ of the variations of locations can be calculated with the following equation:

$$\Delta d = K_v * min(1, L_c/a)^{L_e} \tag{12}$$

where $K_v$ is a constant with the maximum velocity per simulation iteration, and $a$ is the age of each individual (similar to the iteration position but it can be different for each agent). $L_c$ and $L_e$ have similar meanings as before for the originally proposed equation for long durations.

In order to reduce the frequency of the changes in places of residence, a similar equation can be applied for calculating the probability in the decision with the following formula:

$$p_c = min(1, L_c/a)^{L_e} \tag{13}$$

This probability is normally combined with other probabilities that receive input from some environment circumstances and other agent attributes. The direction $\alpha$ can be decided considering the objectives of each agent and other factors. The new location would be calculated from the previous one with the corresponding $\alpha$ and $\Delta d$ by means of equations 10 and 11.

## 3.8. *Framework for agent-based simulation apps and online tools*

The framework of TABSAOND contains a Matlab script with several functions and procedures for assisting practitioners in selecting the appropriate choices for the simulation models. This script contains all the agent decision mechanisms presented in the current article. It also includes all the necessary functions for calculating the variations

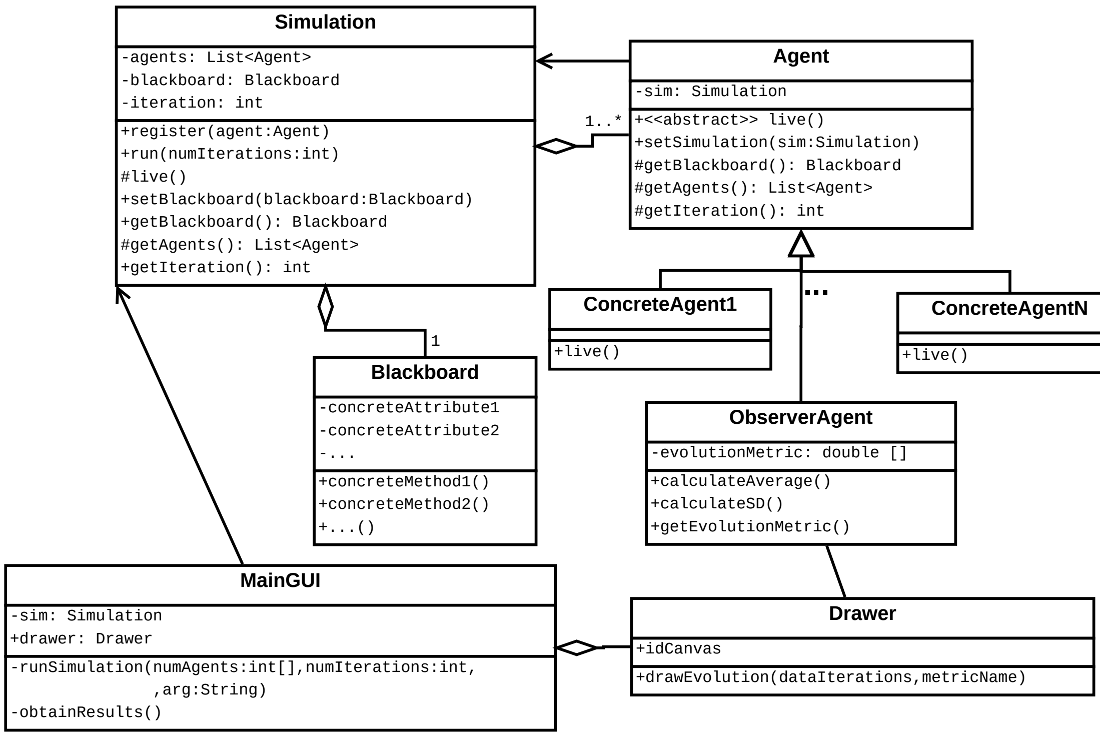


Figure 8: Cass diagram about the definition of the simulation with its agents and the user interface

of agent attributes as described in the current approach. It also has a high order function that allows developers to combine the different decision mechanisms with the different variation mechanisms. In this manner, they can rapidly design and test some simulation model excerpts before actually selecting one and implementing the complete ABS as an app and an online tool. This script also has some examples for plotting charts with several combinations of decision mechanisms and ways of calculating variations.

TABSAOND also has a software framework for developing ABSs constituting mobile apps and online tools. In particular, this framework has been developed with the JavaScript programming language. In this way, an app can be automatically generated with the Apache Cordova toolkit, as one can observe in other developments with this technology [7]. In addition, the ABSs developed with the current framework can also be deployed as online tools, which can be usually executed in the browsers of users, as JavaScript supports Web client programming. For this purpose, the browsers of users need to enable JavaScript with all its functionalities. Both the Matlab script and the reusable part of the JavaScript software framework are available from the TABSAOND website [13].

Figure 8 shows the main excerpt of the class diagram of the software framework of TABSAOND. The framework includes the “Simulation” class. The simulation can contain different types of agents represented with the hierarchy of the “Agent” class. All the agents have a “Live” method that is called periodically in all the iterations of the simulation. In this way, each agent type is represented as a subclass of Agent, and can determine its own behavior by overwriting the Live method. The concrete agent types depend on the specific domain of the ABS, and are defined by the developer. The current approach recommends having the “Observer Agent” besides all the necessary domain-specific agents. The observer agent records the average value of the relevant attribute in all the agents for each iteration. In this manner, it records the evolution of this average value in the whole simulation. At the end of the simulation, it also calculates the SD of the corresponding attribute.

In this framework, all the agents have access to the simulation, and consequently to all the other agents and to the blackboard. In this way, each agent can explicitly communicate with any other agent. In addition, agents can implicitly communicate with each other by changing some information in the “Blackboard”. Its concrete attributes depend on the specific domain.

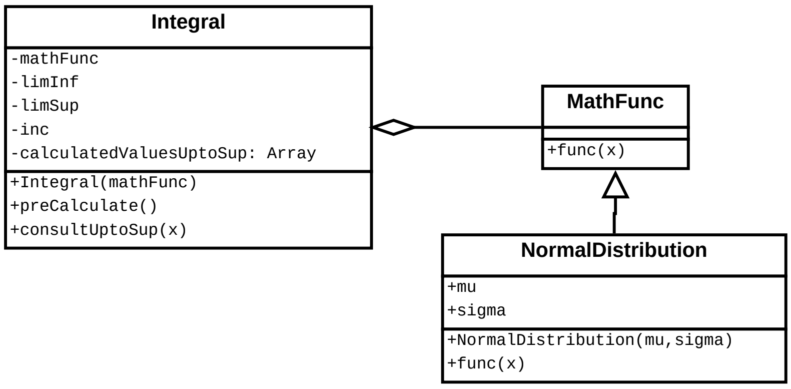


Figure 9: Class diagram concerning the pre-calculation of the integral of a normal distribution

The web-based interface of TABSAOND uses the “Main GUI” class for configuring the simulations with the parameters indicated by the user. The “Drawer” class shows a chart of the evolution of the average of the agent attribute collected by the observer agent. The interface is defined with a web page called “index.html”. This page uses JQuery Mobile so that its style can be properly observed from mobile devices. The charts are represented with the open-source “Chart.js” library.

The software framework of TABSAOND has some math utilities for performing some of the operations proposed in the current approach in an efficient way. For example, Figure 9 shows the class diagram of an excerpt of the framework for pre-calculating the integral of a normal distribution from any x value of the allowed agent attribute interval to the upper limit, considering a finite number of values separated with a certain distance. In this way, the integral is calculated only once from each x value to the upper limit. This framework saves execution time, since each integral value can be requested several times by different agents. In addition, this pre-calculation uses an accumulative numerical integration approach. The pre-calculation of the set of all the considered attribute values only has a global linear computational cost $O(n)$ considering the number of these values, since the integration of each x value is calculated with a constant cost from the integration result of the previous x’ value.

## 4. Evaluation

The current technique has been applied to develop an ABS as a case study. The same ABS has been developed with one of the most similar techniques, which was presented by Gómez et al. [14]. These versions of the ABS have been executed in a data set based on published works [19, 22]. The current evaluation compares the quality of the simulated outputs, the execution times and the development times. Section 4.1 presents the case study implemented with the current technique. Section 4.2 introduces the same ABS developed with the alternative technique, and section 4.3 compares all the aforementioned features between the two versions of the ABS.

### *4.1. Case study developed with TABSAOND*

The current approach has been applied for developing an ABS in the context of heart rate variability (HRV). More concretely, this ABS simulates different mindfulness training programs, and evolves the HRV of some simulated meditators. It simulates the HRV that would be obtained during meditation. The development followed the steps of TABSAOND. The most relevant aspects are described below for each step:

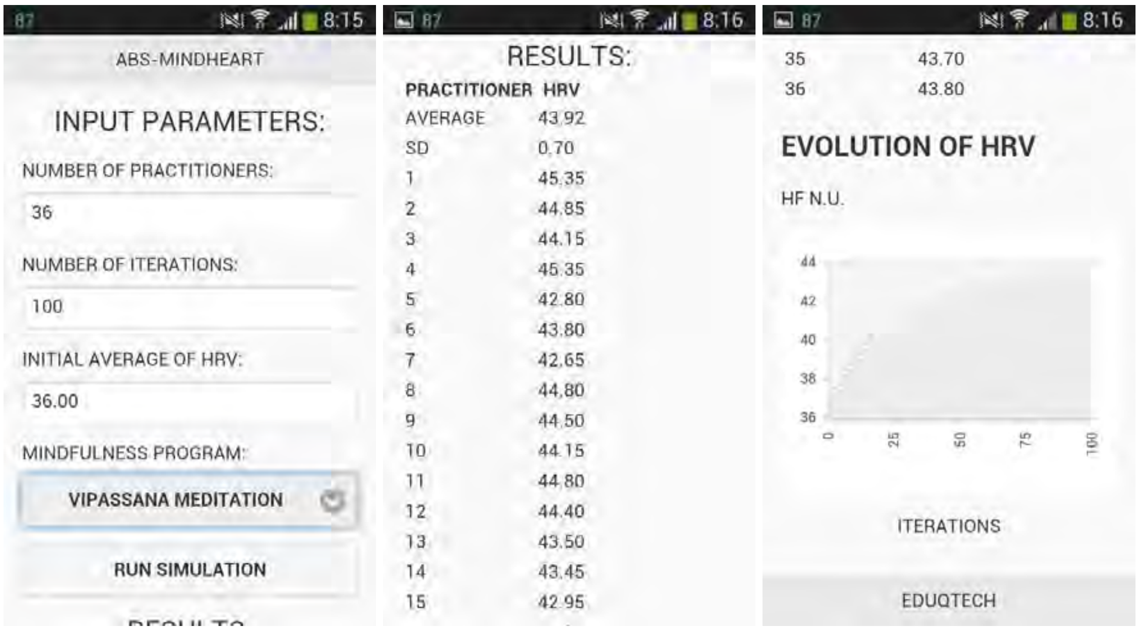
1. *Definition of the agent types*: The developer decided to have two different agent types besides the recommended observer agent. First, the “Practitioner agents” simulate meditators of mindfulness with the corresponding repercussions in their HRV. Second, each “Instructor agent” simulates an instructor that determines a mindfulness program and guides meditators with this program. In particular, the generic instructor agent type is easily extensible with particular instructor agent subtypes. Each of these subtypes determines a specific mindfulness program by implementing the corresponding Live method.

2. *Selection of the agent attributes*: The main feature of practitioner agents is their HRV during meditation. Specifically, the current work has selected the Normalized High Frequency Power (HF n.u.) metric. Among other reasons, this metric has been selected because it is bounded to a specific interval, which is [0, 1]. Remind that agent attributes are recommended to be bounded in TABSAOND. The HF n.u. metric is presented with its percentage.
   The main attributes of the instructor agent type relate to information about the different mindfulness exercises and their influences on HRV. The most relevant feature of each instructor agent subtype is its program for guiding a mindfulness course. It usually indicates which mindfulness exercises are practiced in each iteration by programming code. In this way, developers can combine programming structures such as multiple-selection statements, conditional statements and remainder operations to simplify the definition of mindfulness programs with repetitive schedules.
3. *Modeling the social interactions*: The instructor agent communicates with all the practitioner agents for guiding these in a particular mindfulness program. Specifically it guides each mindful activity of the program in the particular simulation iteration according to its schedule. In each social interaction, the instructor agent transfers a particular mindful activity with some constants that determine its common influence on HRV. These constants are used in both the simulated decisions of the practitioner agents and in the calculation of the agent attribute variations.
4. *Indicating the progress factors*: In this ABS, the progress factor determines whether each practitioner is focused enough to experience the influence of each mindfulness exercise. This is simulated with a constant probability.
5. *Modeling the decisions*: The decisions were modeled for determining the influence of each mindfulness exercise on the HRV of each practitioner agent. Specifically, each practitioner agent is only influenced if the results of two events are consecutively positive. The first one is the progress factor mentioned in the previous step. The second one is the decision that simulates the probability obtained from the integral of a normal distribution considering the current HRV of the practitioner agent by means of equation 2. This ABS uses the initial HRV average of all the practitioners as the mean $\mu$ of the normal distribution as recommended in TABSAOND. Thus, the probability of the decision decreases when getting far from the initial average. In this way, the current ABS considers the experience of the previous influences from mindful exercises. The other constants were adjusted later in the calibration step.
6. *Selecting the way of changing attributes*: In the case of the positive results of both the progress factor and the decision described in the previous steps for a particular agent, the ABS calculates the variation of its HRV attribute. This system uses a fixed variation, in which the final attribute value is checked to avoid surpassing the limits. The combination of decisions based on the integral of a normal variation with fixed variations is one of the recommended options of TABSAOND (remind Figure 6 for instance). The practitioner agent receives the constant for the increasing fixed variation $D_i$ from the transferred information of the social interaction from the instructor agent. In particular, the instructor agent sends a different increasing constant for each mindfulness exercise. Since all the influences have increasing trends, the restriction of limits is checked with the simplified equation 4 of the current approach.
7. *Development of the software*: This ABS was constructed using the JavaScript framework of TABSAOND using an object-oriented approach. This simulator is called ABS-MindHeart, and was developed as an app and as an online tool, both of which are available from the website of the current approach [13]. The app is also available from the Google Play store for Android.
8. *Calibration of the ABS*: This ABS was calibrated by implementing well-known mindfulness programs such as the Vipassana meditation and the Mindfulness Based Stress Reduction (MBSR), and comparing the simulated results with the real ones reported in the literature [19, 22]. This ABS was calibrated by tuning (a) the probability $p$ of the progress factor, (b) the $\sigma$ value in the used normal distribution of equation 1, and (c) the different $D_i$ values for the corresponding mindful activities. Specifically, some of the adjusted values were $p = 0.99$ and $\sigma = 0.04$. In addition, the $D_i$ constants were calibrated with the values presented in table 2 for the different mindful activities.

Figure 10 shows an example of execution of the ABS-MindHeart app. The developer defined each new subtype of the Instructor agent by extending the class representing this agent and implementing the Live method. In this context,

|  | Body Scan | Group Sitting | Instructor Discourse | Long Meditation | Mindful Breathing | Mindful Eating | Mindful Movement | Questions and Answers | Walking Meditation |
|---|---|---|---|---|---|---|---|---|---|
| $D_i$ | 0.80 | 0.70 | 0.00 | 0.30 | 0.75 | 0.40 | 3.30 | 0.00 | 0.30 |

Table 2: The values of $D_i$ for the different mindful activities



ABS-MINDHEART

INPUT PARAMETERS:

NUMBER OF PRACTITIONERS:

36

NUMBER OF ITERATIONS:

100

INITIAL AVERAGE OF HRV:

36.00

MINDFULNESS PROGRAM:

VIPASSANA MEDITATION

RUN SIMULATION

RESULTS:

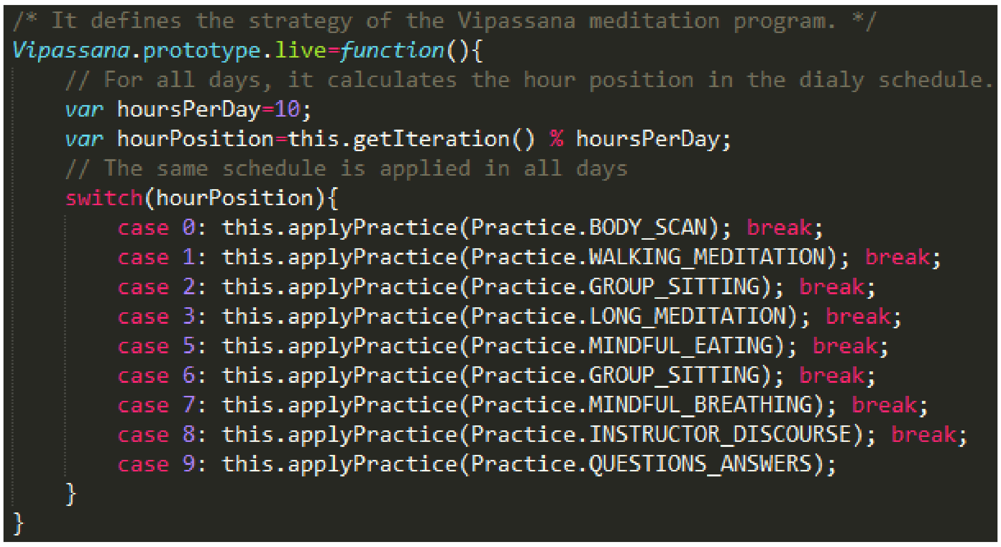
```
PRACTITIONER  HRV
AVERAGE       43.92
SD            0.70
1             45.35
2             44.85
3             44.15
4             45.35
5             42.80
6             43.80
7             42.65
8             44.80
9             44.50
10            44.15
11            44.80
12            44.40
13            43.50
14            43.45
15            42.95
```

```
35            43.70
36            43.80
```

EVOLUTION OF HRV

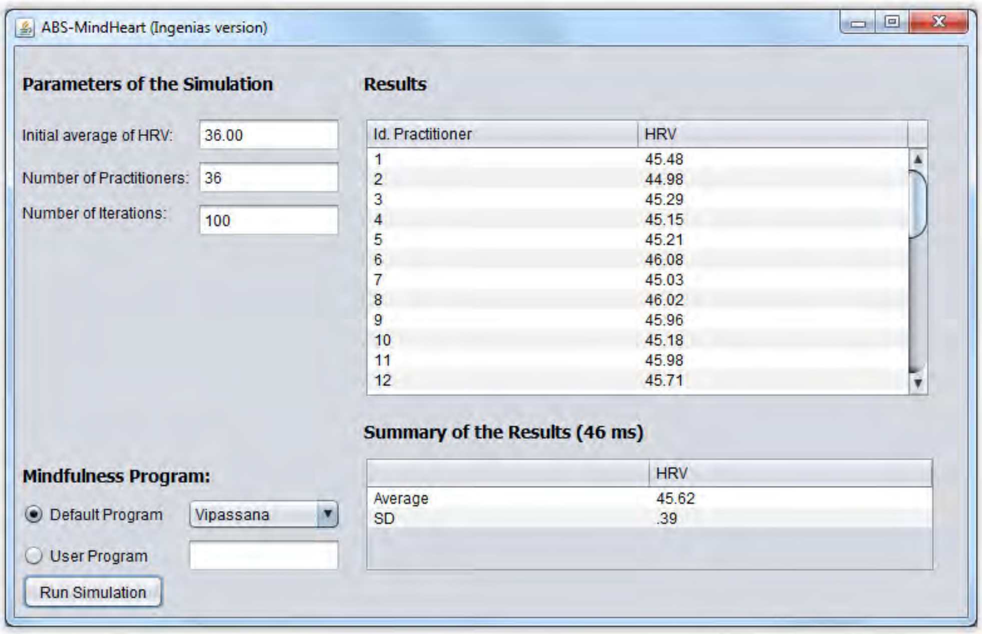
(a) Input of the ABS (b) Results (c) Evolution with a chart

Figure 10: ABS-MindHeart app



```
/* It defines the strategy of the Vipassana meditation program. */
Vipassana.prototype.live=function(){
    // For all days, it calculates the hour position in the dialy schedule.
    var hoursPerDay=10;
    var hourPosition=this.getIteration() % hoursPerDay;
    // The same schedule is applied in all days
    switch(hourPosition){
        case 0: this.applyPractice(Practice.BODY_SCAN); break;
        case 1: this.applyPractice(Practice.WALKING_MEDITATION); break;
        case 2: this.applyPractice(Practice.GROUP_SITTING); break;
        case 3: this.applyPractice(Practice.LONG_MEDITATION); break;
        case 5: this.applyPractice(Practice.MINDFUL_EATING); break;
        case 6: this.applyPractice(Practice.GROUP_SITTING); break;
        case 7: this.applyPractice(Practice.MINDFUL_BREATHING); break;
        case 8: this.applyPractice(Practice.INSTRUCTOR_DISCOURSE); break;
        case 9: this.applyPractice(Practice.QUESTIONS_ANSWERS);
    }
}
```

Figure 11: Definition of the strategy of the Vipassana meditation program


ABS-MindHeart (Ingenias version)

**Parameters of the Simulation**

Initial average of HRV: 36.00

Number of Practitioners: 36

Number of Iterations: 100

**Mindfulness Program:**

[✓ Default Program] Vipassana

User Program

**Results**

| Id. Practitioner | HRV |
|---|---|
| 1 | 45.48 |
| 2 | 44.98 |
| 3 | 45.29 |
| 4 | 45.15 |
| 5 | 45.21 |
| 6 | 46.08 |
| 7 | 45.03 |
| 8 | 46.02 |
| 9 | 45.96 |
| 10 | 45.18 |
| 11 | 45.98 |
| 12 | 45.71 |

**Summary of the Results (46 ms)**

| | HRV |
|---|---|
| Average | 45.62 |
| SD | .39 |

Figure 12: Control version of ABS-MindHeart

Figure 11 shows an example of the implementation of a Live method. In particular, this example defines the strategy of the Vipassana meditation program [19].

### *4.2. ABS developed with the control mechanism*

In order to compare the current approach with another similar one, the same ABS of the previous case study has been developed with the approach proposed by Gómez et al. [14]. In the current experimentation, this version of the ABS is also referred as the control version. The ABS was firstly modeled with the Ingenias modeling language considering the indications of that approach. In order to improve the performance of the simulations, the ABS was developed with the framework of PEABS. Our previous work showed that this framework improves the performance of the Ingenias auto-generated programming code in ABSs [9], so the current paper omits the explanation and proof of this fact for the sake of brevity. The ABS was trained with the same input data as in the previous case study. In particular, the ABS used the Deck random generators, since this kind of generators usually facilitates the training of the ABSs as shown in our previous work about ATABS (a technique for Automatically Training ABSs) [11]. As an example, Figure 12 shows the control version of ABS-MindHeart running the Vipassana meditation program.

### *4.3. Comparison*

The current work compares both versions of ABS-MindHeart in two scenarios: the Vipassana meditation program and the MBSR program. In particular, this work uses a data set extracted from the published studies of respectively Krygier et al. [19] and Nijjar et al. [22].

The simulation inputs were established with the same values as the published data set, and the simulation outputs were compared with the real outcomes according to the data set. Table 3 shows the input circumstances and the outcomes in the real case and the two versions of the ABS. The input circumstances mainly are the number of practitioners and their average pre-program HRV, measured with the HF n.u. metric. In addition, the number of iterations represents the number of meditation hours of the mindfulness program. The main outcome of each simulation is the average post-program HF n.u. of participants.

Since this ABS is a nondeterministic system, the simulation was executed several times to reduce the possible bias because of the output fluctuations. In particular, each version of the ABS was executed 100 times for each mindfulness program in order to obtain an adequate statistical power [6].

| Program | Real or ABS | Practitioners | Iterations | Pre-Program HF n.u. (%) | Post-Program HF n.u. (%) | Mean difference from real |
|---|---|---|---|---|---|---|
| Vipassana | Real | 36 | 100 | 36.00 | 44.00 | |
| | TABSAOND ABS | 36 | 100 | 36.00 | 44.03 (SD 0.15) | 0.03 |
| | Control | 36 | 100 | 36.00 | 45.60 (SD 0.04) | 1.60 |
| MBSR | Real | 18 | 24 | 24.50 | 32.10 | |
| | TABSAOND ABS | 18 | 24 | 24.50 | 32.08 (SD 0.44) | 0.02 |
| | Control ABS | 18 | 24 | 24.50 | 30.99 (SD 0.09) | 1.11 |

Table 3: Input and outcomes of the real cases and the simulations of the two versions of ABS-MindHeart considering 100 simulations for each mindfulness program and ABS

| Program | TABSAOND error (%) | Control error (%) | TABSAOND error / control error (%) |
|---|---|---|---|
| Vipassana | 0.27 (SD 0.21) | 3.63 (SD 0.08) | 7.44 |
| MBSR | 1.09 (SD 0.81) | 3.44 (SD 0.28) | 31.71 |

Table 4: Average errors considering 100 simulations for each program and ABS

Table 3 presents the mean and the standard deviation (SD) of the outputs of the 100 simulations for each program and ABS in the post-program HF n.u. column. As one can observe in the mean differences, the ABS developed with TABSAOND obtains mean results that are quite closer to the real values than the ABS developed with the control mechanism.

In each simulation, the error was calculated as the percentage of the absolute difference between the simulated and real outcomes divided by the value of the real outcome. Table 4 shows the averages and SDs of the error percentages for each ABS and program, considering the 100 simulations in each case. This table also compares the average errors by dividing the ones from TABSAOND ABS by the ones from the control ABS. The average errors of the current approach were only the 7% and 32% of the ones from the control mechanism, respectively for the two mindfulness programs.

The distributions of the outputs and errors of each set of simulations have been analyzed with the Shapiro-Wilk test of normality, and table 5 shows the corresponding results. The HF n.u. outputs of simulations follow a normal distribution for each technique and mindfulness program. This can be considered as an adequate property of the ABS outputs since these are intended to match common real situations, which usually follow normal distributions. However, the output errors do not follow a normal distribution in some cases. This is also reasonable because the errors are calculated considering absolute values of differences, which can include both negative and positive values. Thus, the normal distribution of differences can be regarded as trunked in the zero value when calculating the absolute values.

This work statistically analyzes whether the differences between the outcomes of TABSAOND simulations and the control ones are significant by means of a parametric test, since these values follow a normal distribution. In particular, this work uses the Welch's t-test. This test is an adaptation of the well-known Student's t-test that is more reliable for populations with unequal variances. Table 6 shows the results of this test, and one can observe that the differences are significant with low p-values below not only the common significance level ($\alpha$) of 0.05 but also a significance level of 0.01. This table also includes the effect sizes measured with the Cohen's d and the Partial Eta Squared considering a significance level of 0.05, and in both scenarios the effect sizes were considered to be large according to the guidelines

| | | HF n.u. | | | Error | | |
|---|---|---|---|---|---|---|---|
| | | Statistic | df | Sig. | Statistic | df | Sig. |
| Vipassana | TABSAOND | 0.977 | 100 | 0.082 | 0.896 | 100 | 0.000 |
| | Control | 0.982 | 100 | 0.206 | 0.981 | 100 | 0.161 |
| MBSR | TABSAOND | 0.977 | 100 | 0.078 | 0.906 | 100 | 0.000 |
| | Control | 0.987 | 100 | 0.457 | 0.987 | 100 | 0.441 |

Table 5: Results of the Shapiro-Wilk test of normality

|  | Welch's t-test |  |  |  | Effect Size |  | Power |
|---|---|---|---|---|---|---|---|
|  | Statistic | df1 | df2 | Sig. | Cohen's d | Partial Eta Squared |  |
| Vipassana | 10616.640 | 1 | 111.709 | 0.000 | -14.572 | 0.991 | 1.000 |
| MBSR | 592.818 | 1 | 107.474 | 0.000 | 3.443 | 0.750 | 1.000 |

Table 6: Comparison of HF n.u. with Welch's t-test, the effect size and the statistical power. The statistical power is computed using $\alpha = 0.05$.

|  | Mann Whitney U Test |  | Effect Size |  | Power |
|---|---|---|---|---|---|
|  | Sig. | Decision | Cohen's d | Partial Eta Squared |  |
| Vipassana | 0.000 | Reject the null hypothesis | -20.945 | 0.991 | 1.000 |
| MBSR | 0.000 | Reject the null hypothesis | -3.899 | 0.793 | 1.000 |

Table 7: Comparison of errors of ABSs with the Mann-Whitney U Test, the effect size and the statistical power. The power is computed using $\alpha = 0.05$.

of Cohen [5]. In addition, the statistical power is high with a 1.00 value considering $\alpha = 0.05$, substantially surpassing the convention of 0.80 proposed by Cohen for general use [6].

In order to further determine whether the TABSAOND simulations actually outperform the control ones, the errors of the simulations have been compared with a statistical test. More concretely, this work uses the non-parametric Mann-Whitney U Test, since the distributions of errors do not follow a normal distribution. Table 7 shows the results of this test alongside the effect sizes and the statistical powers. One can observe that the ABS developed with TABSAOND obtains errors that are significant different from the ones obtained in the ABS developed with the control mechanism, with large effect sizes and adequate statistical powers in both scenarios according to the Cohen's guidelines.

Moreover, this work has measured and compared the execution times of the ABSs developed with the two different approaches as shown in table 8. In small amounts of practitioner agents like the ones presented in the previous experiments, both ABSs were properly responsive, i.e. with response times below 0.2 s that are perceived as real-time [18]. In this analysis, we have also considered larger amounts of agents to determine whether the ABS developed with TABSAOND is more efficient and reliable than the control mechanism. More specifically, the number of practitioners were multiplied by powers of ten until $10^4$ in the presented scenarios, leaving all the other input parameters to the same values. One can observe that the ratio of response times between TABSAOND and the control mechanism decreases when the number of agents increments. In particular, this ratio reaches values between 2% and 8% for more than 10,000 agents. Furthermore, for simulations over 100,000 agents, TABSAOND provided the results in only 30 s or less, while the control mechanism lasted more than 18 min until the ABS started overloading the PC without providing any result in both scenarios and the user needed to stop the simulator with the task manager of the Windows operative system. This experimentation used the online version of TABSAOND so that both ABSs were able to be executed in the same hardware and operative system to obtain comparable results. More concretely, both ABSs were run in a PC with an Intel(R) Core(TM) i7 CPU 920 2.67 GHz processor with 8 cores, 6 GB RAM, and the Windows 7 Service Pack 1 (6.1.7601) 64 bit operative system. Therefore, this experimentation advocates that the current approach may

|  | Participants | TABSAOND (s) | Control (s) | TABSAOND / Control (%) |
|---|---|---|---|---|
| Vipassana | 36 | 0.044 | 0.046 | 95.65 |
|  | 360 | 0.058 | 0.139 | 41.73 |
|  | 3600 | 0.761 | 2.330 | 32.66 |
|  | 36000 | 3.182 | 132.088 | 2.41 |
|  | 360000 | 30.304 | NaN |  |
| MBSR | 18 | 0.020 | 0.018 | 111.11 |
|  | 180 | 0.046 | 0.071 | 64.79 |
|  | 1800 | 0.491 | 0.959 | 51.20 |
|  | 18000 | 2.150 | 26.310 | 8.17 |
|  | 180000 | 15.042 | NaN |  |

Table 8: Comparison of execution times. Not a Number (NaN) determines the simulations that did not finish properly.

allow developing ABSs that are more efficient for large amount of agents than the control mechanism. These ABSs may also be able to simulate higher amounts of agents without crashing or overloading the system than the control mechanism.

Furthermore, the development time has also been measured. The development of ABS-MindHeart took 42 h with TABSAOND, while it took 51 h in the case of the control mechanism. Both developments benefited from reusable components offered respectively by the underlying frameworks of the two techniques. TABSAOND reduced the development time according to these experiments. However, this time reduction is not conclusive since only one development has been conducted for each approach and the development time can have been influenced by the background experience of the developer.

## 5. Conclusions and future work

The current work has presented a technique for modeling and constructing ABSs in which agents take decisions in a nondeterministic way. One of its main novelties is the presentation and discussion about different alternatives for simulating the evolution of the internal agent attributes. In addition, the current approach is also the first general technique for developing specifically ABSs as both mobile apps and online tools.

In the current approach, the evolution of agent attributes considers nondeterministic scenarios where (a) agents take decisions based on certain inputs with some mechanisms, and (b) the attribute variations can be calculated with different formulas. The decision inputs can be the agent attributes, the information received from social interactions, the previous experience, some locations and the environment. In this approach, some probabilities are determined from these inputs, and then the corresponding decision is simulated considering these probabilities. Regarding the agent attribute variations, this work shows the repercussions of the proposed variation formulas in combination with the decision mechanisms. This technique is supported with a framework that allows developers to construct ABSs as both apps and online tools.

The current approach has been compared with the method proposed by Gómez et al. [14], which is one of the most similar ones in purpose. The experimentation shows that the current approach allows one to obtain ABSs that are more accurate in terms of similarity between simulated and real results. The final product is also more efficient and reliable for large amounts of agents. The results also advocate that development time is lower with TABSAOND than with the analyzed alternative.

The current approach is planned to be extended in several manners. The current approach will be experienced with more case studies to further determine the generality of the current approach in different domains and kinds of simulations. In particular, TABSAOND will be applied to develop ABSs with spatial information. In this manner, the current approach will include more details to overcome certain problems of this kind of simulations, especially in the ones with excessive numbers of individuals. For instance, some hints will be provided for preventing spatial simulations from getting into deadlocks. The current approach will also be compared with more alternative techniques. All this future experimentation will be used to further assess the current findings about the similarity between simulated and real outcomes, the execution times and the development time.

*Acknowledgments*

The current work has been supported by the “Universidad de Zaragoza”, “Fundación Bancaria Ibercaja” and “Fundación CAI” in the “Programa Ibercaja-CAI de Estancias de Investigación” in the context of the project “Sistema basado en agentes para simular y trabajar las emociones en los programas de mindfulness” with reference IT6/15. This work has also been done in the context of the project “Soluciones tecnológicas para facilitar la práctica de mindfulness: caminando hacia mHealth” with reference TEC2013-50049-EXP and supported by the Spanish Ministry of Economy and Competitiveness. Furthermore, we acknowledge the “Fondo Social Europeo” and the “Departamento de Tecnología y Universidad del Gobierno de Aragón” for their joint support with grant number Ref-T81.

## References

[1] Auld, J., Hope, M., Ley, H., Sokolov, V., Xu, B., Zhang, K., 2016. POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations. Transportation Research Part C: Emerging Technologies 64, 101–116.

[2] Bouanan, Y., Zacharewicz, G., Vallespir, B., 2016. DEVS modelling and simulation of human social interaction and influence. Engineering Applications of Artificial Intelligence 50, 83–92.

[3] Caballero, A., Botía, J., Gómez-Skarmeta, A., 2011. Using cognitive agents in social simulations. Engineering Applications of Artificial Intelligence 24 (7), 1098–1109.

[4] Campillo-Sanchez, P., Gómez-Sanz, J. J., 2014. Agent based simulation for creating ambient assisted living solutions. In: Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection. Vol. 8473 of Lecture Notes in Computer Science. Springer, pp. 319–322.

[5] Cohen, J., 1988. Statistical power analysis for the behavioral sciences (2nd ed.). Hillsdale, NJ: Lawrence Earlbaum Associates.

[6] Cohen, J., 1992. A power primer. Psychological bulletin 112 (1), 155–159.

[7] de Abreu Freire, C., Painho, M., 2014. Development of a mobile mapping solution for spatial data collection using open-source technologies. Procedia Technology 16, 481–490.

[8] Fortino, G., Russo, W., 2012. ELDAMeth: An agent-oriented methodology for simulation-based prototyping of distributed agent systems. Information and Software Technology 54 (6), 608–624.

[9] García-Magariño, I., Gómez-Rodríguez, A., González-Moreno, J. C., Palacios-Navarro, G., 2015. PEABS: A Process for developing Efficient Agent-Based Simulators. Engineering Applications of Artificial Intelligence 46, 104–112.

[10] García-Magariño, I., Medrano, C., Lombas, A. S., Barrasa, A., 2016. A hybrid approach with agent-based simulation and clustering for sociograms. Information Sciences 345, 81–95.

[11] García-Magariño, I., Palacios-Navarro, G., 2016. Atabs: A technique for automatically training agent-based simulators. Simulation Modelling Practice and Theory 66, 174–192.

[12] García-Magariño, I., Palacios-Navarro, G., 2016. A model-driven approach for constructing ambient assisted-living multi-agent systems customized for parkinson patients. Journal of Systems and Software 111, 34–48.

[13] García-Magariño, I., Palacios-Navarro, G., 2016. TABSAOND website. Last accessed on August 27, 2016.
URL `http://webdiis.unizar.es/~ivangmg/tabsaond/`

[14] Gómez-Sanz, J. J., Fernández, C. R., Arroyo, J., 2010. Model driven development and simulations with the INGENIAS agent framework. Simulation Modelling Practice and Theory 18 (10), 1468–1482.

[15] Gutierrez, C., García-Magariño, I., 2011. Revealing bullying patterns in multi-agent systems. Journal of Systems and Software 84 (9), 1563–1575.

[16] Gutiérrez, C., García-Magariño, I., 2013. A measurement approach for overcoming unbalanced overwork in multi-agent systems. International Journal on Artificial Intelligence Tools 22 (04), 1350026.

[17] Hall, A., Virrantaus, K., 2016. Visualizing the workings of agent-based models: Diagrams as a tool for communication and knowledge acquisition. Computers, Environment and Urban Systems 58, 1–11.

[18] Jacko, J. A., 2012. Human computer interaction handbook: Fundamentals, evolving technologies, and emerging applications. Boca Raton, FL: CRC press Taylor & Francis Group.

[19] Krygier, J. R., Heathers, J. A., Shahrestani, S., Abbott, M., Gross, J. J., Kemp, A. H., 2013. Mindfulness meditation, well-being, and heart rate variability: a preliminary investigation into the impact of intensive Vipassana meditation. International Journal of Psychophysiology 89 (3), 305–313.

[20] Lin, S.-m., Lin, F.-r., 2016. Coordinating mobile activities by integrating simulated and physical software agents. Information Sciences 348, 49–67.

[21] Molesini, A., Casadei, M., Omicini, A., Viroli, M., 2013. Simulation in agent-oriented software engineering: The SODA case study. Science of Computer Programming 78 (6), 705–714.

[22] Nijjar, P. S., Puppala, V. K., Dickinson, O., Duval, S., Duprez, D., Kreitzer, M. J., Benditt, D. G., 2014. Modulation of the autonomic nervous system assessed through heart rate variability by a mindfulness based stress reduction program. International journal of cardiology 2 (177), 557–559.

[23] Resta, M., 2015. An agent-based simulator driven by variants of self-organizing maps. Neurocomputing 147, 207–224.

[24] Ronald, N., Dignum, V., Jonker, C., Arentze, T., Timmermans, H., 2012. On the engineering of agent-based simulations of social activities with social networks. Information and Software Technology 54 (6), 625–638.

[25] Ross, S. M., 2014. Introduction to probability models. San Diego, CA: Elsevier Academic Press.

[26] Serrano, E., Iglesias, C. A., 2016. Validating viral marketing strategies in twitter via agent-based social simulation. Expert Systems with Applications 50, 140–150.

[27] Serrano, E., Moncada, P., Garijo, M., Iglesias, C. A., 2014. Evaluating social choice techniques into intelligent environments by agent based social simulation. Information Sciences 286, 102–124.

[28] Zambonelli, F., Omicini, A., Anzengruber, B., Castelli, G., De Angelis, F. L., Serugendo, G. D. M., Dobson, S., Fernandez-Marquez, J. L., Ferscha, A., Mamei, M., et al., 2015. Developing pervasive multi-agent systems with nature-inspired coordination. Pervasive and Mobile Computing 17, 236–252.