A Simple-to-use BDI architecture for Agent-based Modeling and Simulation

Philippe Caillou[1], Benoit Gaudou[2], Arnaud Grignard[3,4], Chi Quang Truong[3,5], and Patrick Taillandier[4]

[1] UMR 8623 LRI, University of Paris Sud, Paris, France
[2] UMR 5505 IRIT, University of Toulouse, Toulouse, France
[3] UMI UMMISCO, University Pierre and Marie Curie/IRD, Paris, France
[4] UMR 6266 IDEES, University of Rouen, Rouen, France
[5] CENRES, DREAM Team, Can Tho University, Can Tho, Vietnam
caillou@lri.fr, benoit.gaudou@ut-capitole.fr,
agrignard@gmail.com, tcquang@ctu.edu.vn,
patrick.taillandier@gmail.com

Abstract. With the increase of computing power and the development of user-friendly multi-agent simulation frameworks, social simulations have become increasingly realistic. However, most agent architectures in these simulations use simple reactive models. Cognitive architectures face two main obstacles: their complexity for the field-expert modeler, and their computational cost. In this paper, we propose a new cognitive agent architecture based on the BDI (Belief-Desire-Intention) paradigm integrated into the GAMA modeling platform. Based on the GAML modeling language, this architecture was designed to be simple-to-use for modelers, flexible enough to manage complex behaviors, and with low computational cost. This architecture is illustrated with a simulation of the evolution of land use in the Mekong Delta.

1 Introduction

Agent-based simulations are widely used to study complex systems. However, the problem of the agent design is still an open issue, especially for models tackling social issues, where some of the agents represent human beings. In fact, designing complex agents able to act in a believable way is a difficult task, in particular when their behaviour is led by many conflicting needs and desires. A classic paradigm to formalise the internal architecture of such complex agents is the BDI (Belief-Desire-Intention) paradigm [4]. This paradigm allows to design expressive and realistic agents, yet it is barely used in social simulations. One explanation is that most agent architectures based on the BDI paradigm are too complex to be understood and used by non-computer-scientists. Moreover, they are often very time-consuming in terms of computation and thus not adapted to simulations with thousands of agents.

In this paper, we propose a new architecture that is integrated into the GAMA platform. GAMA is an open-source modeling and simulation platform for building spatially explicit agent-based simulations ([6, 1]). Its complete modeling language (GAML: GAma Modeling Language) and integrated development environment support the definition of

large scale models (up to millions of agents) and make it usable even with low level programming skills. Our architecture was implemented as a new GAMA plug-in, and allows to directly and simply define BDI agents through the GAML language.

The paper is structured as follows: section 2 proposes a state of the art of BDI architectures and their use in simulation context. Section 3 is dedicated to the presentation of our architecture. In Section 4, we present a simple case study using this architecture to study the land-use change in a village of the Mekong Delta (Vietnam). At last, Section 5 provides this paper with a conclusion and some perspectives.

2 State of the art

The BDI approach has been proposed in Artificial Intelligence [4] to represent the way agents can do complex reasoning. It has first been formalized using Modal Logic [5] in order to disambiguate the various concepts (Belief, Desire and Intention) and the logical relationships between them (concepts are detailed in Section 3.1).

BDI frameworks. In parallel, BDI operational architectures have been developed in order to help the development of Multi-Agent Systems embedding BDI agents. Some of these BDI architectures are included in framework allowing to directly use them in different applications. A classic framework is the Procedural Reasoning System (PRS) [9]. This framework includes three main processes: the perception (in which agent acquires information from the environment), the central interpreter (which helps the agent to deliberate its goals and then to select the available actions) and the execution of intention (which represents agents reactions). This framework has been used as a base for many other frameworks. For instance, the JACK [7] commercial framework inherits many properties from PRS. JACK allows the user to define a multi-agent system with BDI agents using a dedicated language (a super-set of Java). It was used in many commercial applications (e.g. video-game, oil trading,...). Another classic framework for multi-agent system building is JADE [3]. This open-source Java framework integrates several add-ons dedicated to the definition of BDI agents. The most advanced framework is Jadex [12], that is an add-on of the JADE framework. In comparison to JACK, Jadex proposes an explicit representation of goals.

BDI agents in agent-based modelling and simulation platforms. BDI architectures agents have been introduced in several agent-based modelling and simulation platforms. For example, Sakellariou et al.[14] have proposed an extension to Netlogo [21] to deal with BDI agents. The extension allows the model to add to agents a set of beliefs (information it gets by perception of communication) and intentions (what it wants to execute), and ways to manage these two sets. This very simple architecture is inspired by the PRS architecture (in particular using an intention stack) and is education-oriented. Its main aim was to allow modellers to manipulate BDI concepts in a simple language.

Singh and Padgham[15] went one step further in the integration between BDI architecture and agent-based modelling and simulation platforms. They propose a framework able to connect agents-based platforms and an existing BDI framework (such

as JACK [7] or Jadex [12]). An application couples the Matsim platform [2] and the GORITE BDI framework [13]. Their framework aims at being generic and can be extended to couple any kind of simulation platforms and BDI frameworks. This approach is very powerful but remains computer-scientist-oriented, as it requires high programming skills to develop bridges between the framework and the platforms, and to write agents behaviours without a dedicated modelling language.

First attempts already exist to integrate BDI agents into the GAMA platform [6]. Taillandier et al.[16] proposed a BDI architecture where the choice of plans is formalized as a multi-criteria decision-making process: desires are represented by criteria that will be used to make a decision. Each plan is evaluated by each criterion according to the beliefs of the agent. However, this architecture was tightly linked to its application context (farmer decision making) and does not propose any formalism to model the agent beliefs and is rather limited concerning the way the plans are carried out: there is for example no possibility to have complex plans that require sub-objectives. Le et al.[8] proposed another architecture dedicated to simulation with a formalized description of beliefs and plans and their execution. However, the desires and plans have to be written in a very specific and complex way that can be difficult to achieve for some application contexts, in particular for non-computer scientists. In addition, this architecture has a scalability problem: it does not allow to simulate thousands of agents.

3 Presentation of the SimpleBDI Plug-in Architecture

3.1 Overview

Consider a simple Chopper-Fire model: A chopper agent patrols, looking for fires. Whan it finds one, it tries to extinguish it by dropping water, and when it has no more water, it goes to the nearest lake to refill its water tank. With a reactive agent model, defining an agent behavior means to define *What it does* (e.g. patrol, go to the fire, go to take water). This can be achieved both with reflexes or a finite state machine. Using a cognitive model means to define *what it wants* (e.g. to find fire, to extinguish a specific fire, to get water) and *how to do it* (e.g. if I want to find a fire, I patrol (wandering randomly in my environment) and if I see a fire, I want it to be extinguished. If I want to put out a fire, go toward it and put water, and if I have no more water, get some). There are several advantages for using such cognitive approach: complex reasoning (planning), persistence (of the goals), easy to improve (both on what to do and how to do it), easy to use (the modeler can define goals instead of reactions) and easy to analyze (it is possible to know why - for what purpose - agents do what they do).

The architecture and the vocabulary can be summarized with this simple Fire-Chopper example: the Chopper agent has a general *desire* to patrol. As it is the only thing he wants at the beginning, it is its initial *intention* (what it is doing). To patrol, it wanders around (its *plan* to patrol). When it *perceives* a fire, it stores this information (it has a new *belief* about the existence of this fire), and it has a new *desire* (it wants the fire to be extinct). When it sees a fire, the Patrol *intention* is put *on hold* and a new *intention* is selected (to put out the fire). To achieve this *intention*, the *plan* has two steps, i.e. two new *(sub)desires*: go to the fire and put water on the fire. And so on.

3.2 Vocabulary

The vocabulary introduced in the previous example can be summarized as follows.

Knowledge. Beliefs, Desires and Intentions are described using **predicates**. A predicate has a name, and may also have a value (with no constraint on the type) and some parameters (each defined by a name and a value); For example $Fire(true, (Position :: (12, 16)))$ — a Fire is present (value true) at position (12,16) — or $HaveWater(true)$ — the Chopper has some water (value true).

- **Beliefs** (what it thinks). Beliefs is the internal knowledge the agent has about the world. The belief base is updated during the simulation. A belief is described by a predicate and is in general true or false. For example the predicates $Fire(true, (Position :: (12, 16)))$ is added when the agent perceives a fire at position (12,16).
- **Desires** (what it wants). Objectives that the agent would like to accomplish (for example $Fire(false, Position :: (12, 16))$, the agent wants the previous fire to be put out). They are stored as a set of desires. A desire is fulfilled when it is present in the Belief base (or manually removed by the agent). Like the Belief base the Desire base is updated during the simulation. Desires can be related by hierarchical links (**sub/super desires**) when a desire is created as an intermediary objective (for example to extinct a fire can have two sub-desires: go to the fire and put water on the fire). Desires have a **priority** value (that can be dynamic), used to select a new intention among the desires when necessary.
- **Intentions** (what it is doing). What the agent has chosen to do. The **current intention** will determine the selected plan. Intentions can be put **on hold** (for example when they require a subdesire to be achieved). For this reason, there is a stack of intention, the last one is the current intention, and the only one that is not on hold.

Behaviour.

- **Perception.** A perception is a function called at each iteration, where an agent can eventually update its belief or desire bases. It is technically identical to a **reflex** of a reactive architecture (a function called at each step).
- **Plan.** The agent has a set of **plans**, which are behaviors defined to reach specific desires. A plan can be instantaneous and/or persistent ($goToPosition$). Plans may have a **priority** value (that can be dynamic), used to select a plan when several possible plans are available.

3.3 Thinking process

At each step, the agent applies the process described on Fig. 1. Roughly, the agent will perceive the environment, then i) **continue its current plan** if it is not finished, or ii) if the plan is finished and its current intention is not fulfilled, it **selects a plan**, or iii) if its current intention is fulfilled, it **selects a new desire** to add to its intention stack. More precisely:

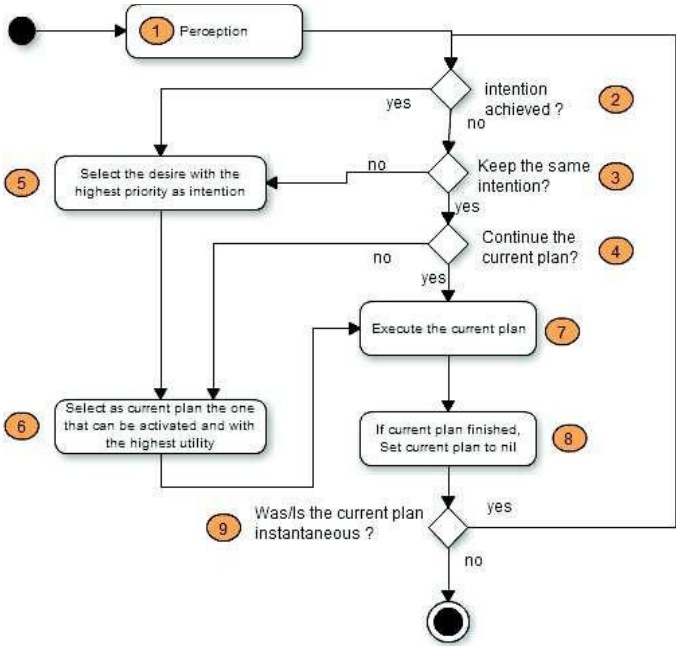

Fig. 1. Activity diagram

1 - Perceive: Reflexes/Perceptions are applied. This may update the Beliefs and add new Desires.

2 - Is one of my intentions achieved?: If one of my intentions is achieved, set current plan to nil and remove the intention and all its subdesires from the desire and intention base (if I or someone else has extinguished $Fire1$, I remove the desire $Extinguish(Fire1)$ from my desires, but also the sub-desires $MyPosition(Fire1)$ and $WaterOn(Fire1)$ if I have them). If the achieved intention super-desire is on hold, it is reactivated (its sub-desire just got completed).

3 - Do I keep the current intention?: To take into account the environment instability, an intention-persistence coefficient ip is applied: with a probability ip, the current intention is removed from the intention stack. More details about this coefficient are given in section 3.4.

4 - Do I have a current plan?: If I have a current plan, just execute it. As for the current intention stability, the goal is both persistence (I stick to the plan I have chosen) and efficiency (I don't choose at each step). For the same reason that the current intention is randomly removed, a plan-persistence coefficient pp is defined: with a probability pp, the current plan is just dropped.

5 - Choose a desire as new current intention: If the current intention is on hold (or the intention base is empty), choose a desire as new current intention. The new intention is selected among the desires with the highest priority (and not already present in the intention base).

6 - Choose a plan as new current plan: The new current plan is selected among the plans compatible with the current intention and with the highest priority.

7 - Execute the plan: The current plan is executed.

8 - Is my plan finished?: To allow persistent plans, a plan may have a termination condition. If it is not reached, the same plan will be kept for the next iteration.

9 - Was my plan instantaneous?: Most multi-agent based simulation frameworks (GAMA included) are synchronous frameworks using steps. One consequence is that it may be useful to apply several plans during one single steps. For example, if a step represents a day or a year, it would be unrealistic for an agent to spend one step to apply a plan like "To fight this fire, lets define two new sub-desires, go to the fire and put water on the fire, and put my objective on hold". This kind of plan (mostly reasoning) can be defined as **instantaneous**: in this case a new thinking loop is applied during the same agent step.

3.4 Properties

Persistence and priority. Persistence coefficients and priority values are key properties of many BDI architectures. Agents with high persistence continue their current actions independently of the environment evolution (they are more stable and spend less time rethinking their plans and objectives). Less persistent agents are more adaptable and reactive but may lead to erratic and computationally costly behaviors. Priority values will determine both the selected desires and plans. The choice can be deterministic (highest priority selected) or probabilistic (highest priority have a higher probability to be selected). One advantage of the GAMA modeling framework for both persistence and priority coefficients is to allow to use dynamic or function-based variables. Plans and Desires priority values and agent persistence coefficients can be changed by the agent itself (for example, a plan could increase the persistence coefficient after evaluating the previous plan success). The modeler can also define functions to update a value. For example, the priority of a plan or desire could be defined as a function of the current step, which would make it more and more probable to be selected when the simulation advances.

Flexibility. One core objective when defining this architecture was to make it as simple-to-use and flexible as possible for the modeler. The modeler can use the architecture in its full potential (for example dynamic coefficients as presented before), but he/she can also use only some parts of the architecture. It is for example possible to define only Desires and Plans, and no Beliefs (the effect would be that the intentions and desires achievement and removal will have to be done manually, i.e. defined by the modeler in the agent plans). Most parameters have default values and can be omitted. For example, the modeler doesn't have to know the existence of instantaneous plans (by default off), plan termination condition (by default true: always terminated at the end of its execution), or the possibility to define sub-desires or put intentions on hold.

GAMA integration. The architecture is defined as a GAMA species architecture. The modeler only requires to define simpleBDI as agent architecture and define at least one

plan to be operational. After that the modeler mostly defines plans to act and (usually one) reflexes to perceive. Many keywords are defined to help the user to update and manage both Belief, Desire and Intentions bases and create/manage predicates. In the next section, we present an application of the architecture to a social simulation context.

4 Case study: land use change in coastal area of the Mekong Delta

4.1 Context of the case study

The Mekong Delta region will be heavily influenced by the effects of global climate change [20]. Indeed, the sea level rise and salt water intrusion will strongly impact the life of people and the situation of agricultural production [18]. Nhan [10] pointed out that the environmental conditions significantly impact the agriculture and fisheries and that ordinary people tend to spontaneous change the land-use, which causes difficulties for land resource management and cultivation of farmers. Another difficulty comes from the behaviors of farmers that tend to adapt their food production to the market [17]. As showed in [11], the difference of planned and real production can be observed at the village level, where the land-use change has not evolved as expected. It is thus important to be able to understand the land-use planning at village level to be able to predict the evolution of land-use change at province level. In this context, we chose to study the evolution of land-use in the village of Binh Thanh. This coastal village of the Ben Tre province of the Mekong Delta is representative of regions with a mix of brackish and fresh water, where the land-use is strongly impacted by the irrigation planning.

4.2 Collected Data

We have collected data concerning the land-use of each parcel of this village in 2005 and in 2010 from the Department of Natural Resources and Environment of the Ben Tre province. In this area, six land-use types were defined: Rice, Rice - Vegetable, Rice - Shrimp, Annual crops, Industrial Perennial tree and Aquaculture. We collected as well the soil map, the saltwater map and the flood map of the regions and defined from them six land-unit types. From each of these land-unit types, we defined with the help of domain-experts a suitability value for each type of land-use (the lower the better). This suitability represents the adequacy between the land unit type and the land use type. For instance, producing industrial perennial tree on a salty soil is very difficult and the yield will be very low. Another data source that was built with domain-experts were the transition values for each type of land-use. This matrix allows to represent the technical difficulty to pass from one land-use type to another. This difficulty was evaluated using three values (1: easy, 2: medium, 3: very difficult). Finally, we collected data concerning the evolution of benefit and cost of each land-use type per hectare from 2005 to 2010.

4.3 Implemented Model

The model was defined in order to simulate the evolution of the land-use of the Binh Thanh village. We make the assumption that each farmer has only one parcel and that

he has to make a decision concerning the land-use of the parcel every year. A simulation step in this model represents then 1 year.

In this simple model, the main species of agents is the parcel species that represents the farmer and his/her parcel (5721 parcels for the study area). We use our SimpleBDI agent architecture for this species of agents.

A parcel agent has the following attributes:

- *shape*: geometry of the parcel (static)
- $profile$: the inclination of the farmer toward a change of production. It is used to set the value of the *intention_persistence* (ip) variable. We defined five possible values: innovator (2.5% - ip: 0.0), early adopters (13.5% - ip: 0.1), early majority (34% - ip: 0.2), late majority (34% - ip: 0.3), laggard (16% - ip: 0.5) (static)
- *landunittype*: type of soil for the parcel (static)
- *neighbors*: list of parcels at a distance of 1 kilometre (static)
- $land - use$: type of production (dynamic)

In addition to the parcel agents, we define a world agent that contains all the global variables:

- $profit\_matrix$: for each year (from 2005 to 2010), for each land-use type, the benefit that can be expected from 1 ha of production.
- *cost_matrix*: for each year (from 2005 to 2010), for each land-use type, the cost of 1 ha of production.
- *suitability_by_landuse*: for each land unit type, the suitability to produce a given land-use type.
- *transition_matrix*: difficulty to pass from a land-use to another one.

At each simulation step (i.e. every year), each parcel agent is activated in a random order.

In our model, each parcel agent has the following belief and desire base:

- $Beliefs$: pieces of knowledge concerning the expected profit for each land use for each land-unit type. Each belief corresponds to a land use type, a land-unit type, and a profit associated to it.
- $Desires$: for each type of production, the agent will have a desire to do it. In addition, the agent could have desires to give information (a belief) to the other farmers in its neighborhood concerning the expected price for a land-use type and a land-use unit (see below)

The priority of its "do a production" desires will be computed according to a multi-criteria analysis. This type of decision making process is often used for land-use change models (see for example [16]). We defined 3 criteria for the decision: the profit, the cost and the transition difficulty. Indeed, it is generally accepted that farmers tend to choose a production that maximizes their profits, that minimizes the cost - avoid risky productions - and that are easy to implement. More precisely, the criterion values are computed as follows for a given transition from *old_lu* to *lu* and a given *soil* type (i.e. land-unit type) and *year*:

$$Profit(lu, soil, year) = \frac{matrix\_profit(lu, year)}{(max\_profit(year) * matrix\_suitability(soil, lu))} \quad (1)$$

With:

$$max\_profit(year) = max(matrix\_profit(lu, year)) \quad (2)$$

$$Transition(old\_lu, lu) = \frac{(3 - transition\_matrix(old\_lu, lu))}{2} \quad (3)$$

To fulfil its desires, the agent can activate a dedicated plan: *do_production*. The GAML code of this plan is presented in Figure 2. This plan is activated when the current intention is to produce something. First, the agent gets the current intention, and changes its land-use according to it. After that, it creates a new belief concerning the real profit that it got from this land-use and updates its old belief (concerning the profit of this land-use). Then, it diffuses its new belief to its neighborhood (call the *diffuse_information* action, see below) and puts its current intention on-hold (wait to finish to diffuse the information before producing again).

```
plan do_production when: is_current_intention(predicate_do)  {
    predicate do_current <- get_current_intention();
    landuse <- string(do_current.value);
    string id_belief <- "profit,"+ landuse + "," + land_unit;
    predicate new_profit_belief <- new_predicate(id_belief, compute_profit(landuse));
    do replace_belief(get_belief_with_name(id_belief),new_profit_belief);
    do_predicates[landuse] <- do_predicates[landuse] with_priority updatePriority(landuse);
    do diffuse_information(new_profit_belief);
    do current_intention_on_hold();
}
```

Fig. 2. Production plan

Figure 3 presents the GAML code of the *diffuse_information* action. When this action is called, the agent does a loop on all the people in its neighborhood. For each of these people, the agent adds a new sub-intention to its current intention (produce a given land-use) to diffuse its new belief to this people.

```
action diffuse_information(predicate information) {
    loop people over: neighbours {
        predicate inform_people <- new_predicate("inform", people::information);
        do add_subintention(get_current_intention(),inform_people, true);
    }
}
```

Fig. 3. Diffuse information action

In order to fulfill its information diffusion intention, the agent can activate a dedicated plan: *inform_people*. The GAML code of this plan is presented in Figure 4. This plan is instantaneous, as we consider that the time taken to inform its neighborhood is insignificant in comparison to the simulation step (one year). It is activated when the current intention is to inform someone. First the agent gets the people to inform and the information to diffuse from the current intention. After that, it asks the people to inform to receive the new information (call the *diffuse_information* action) and remove the current intention (and desire) from its intention base (desire base).

```
plan inform_people when: is_current_intention(predicate_inform) instantaneous: true{
    predicate inform_current <- get_current_intention();
    land_parcel_bdi people_to_inform <- land_parcel_bdi(pair(inform_current.value).key);
    predicate information <- predicate(pair(inform_current.value).value);
    ask people_to_inform {
        do get_information(information);
    }
    do remove_intention(get_current_intention(),true);
}
```

Fig. 4. Information diffusion plan

The complete source code of the model is available on the GAMA SVN [1].

4.4 Experiments

The different parameter values of the models were defined by using a genetic algorithm to find the parameter set that fits the best the real data, i.e. minimization of the fuzzy kappa coefficient[19] computed by comparing the observed data in 2010 and the simulation result for the same date. The fuzzy kappa coefficient allows to compare two maps by taking into account the neighborhood of the parcels. This coefficient is between 0 (not similar at all) to 1 (totally similar).

Figure 5 shows simulation results obtained for the model and the observed data. As shown, the observed land-use is close to the real one.

To quantitatively evaluate the simulation results of the model, we used two indicators: the fuzzy kappa coefficient (local indicator) and the percent absolute deviation (global indicator). This second indicator that is often used to evaluate land-use change models is computed by the following formulae:

$$PAD(\%) = 100\frac{\sum_{i=1}^{n} |\widehat{X}_i - X_i|}{\sum_{i=1}^{n} \widehat{X}_i} \tag{4}$$

with: $\widehat{X}_i$ the observed quantity of parcels with the land-use i and X_i the simulated quantity of parcels with the land-use i.

As our model is stochastic, we ran 100 times each model and computed the average fuzzy kappa coefficient (kappa) and percent absolute deviation (pad). We obtained for the pad a value of 35.98% (the lower the better) and for the fuzzy kappa a value of 0.545

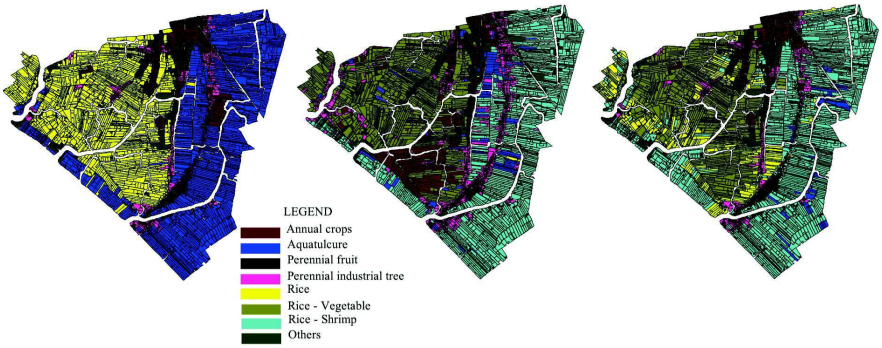

Fig. 5. Land-use for 2005 (left); land-use obtained for 2010 with the simulation (middle); observed land-use for 2010 (right)

(the higher the better). These results are rather good and show that the model is able to reproduce in relevant way the real dynamic.

Concerning the computation time (on a Macbook pro computer from 2011), the mean duration of a simulation step was less than 0.6 seconds. This result is quite promising considering that we have more than 5700 BDI agents that can have many desires and that can activate many plans during the same simulation step with the information diffusion process.

5 Conclusion

In this paper, we have presented a new BDI architecture dedicated to simulation context. This architecture is integrated into the GAMA modeling and simulation platform and directly usable through the GAML language, making it easily usable even by non-computer scientists. We have presented a first simple application of this architecture concerning the land-use change in the Mekong Delta (Vietnam). This first application showed that our plug-in allows to built relevant models and to simultaneously simulate several thousand of agents.

If our architecture is already usable, some improvements are planned. First, we want to improve the inference capabilities of our architecture: when a new belief is added to the belief base, desire and intention bases should be updated in a efficient way as well. Second, we want to make it even more modular by adding more possibility concerning plans and desire choices and not just the plan/desire with the highest priority: let the possibility to make user-defined or with a multi-criteria decision process, etc. At last, we want to add the possibility to use high performance computing (distribute the computation on a grid or cluster) to decrease the computation time.

6 Acknowledgement

This work is part of the ACTEUR ("Spatial Cognitive Agents for Urban Dynamics and Risk Studies") research project funded by the French National Research Agency.

References

1. GAMA website: http://gama-platform.org, 2015.
2. M. Balmer, M. Rieser, K. Meister, D. Charypar, N. Lefebvre, K. Nagel, and K. Axhausen. Matsim-t: Architecture and simulation times. *Multi-Agent Systems for Traffic and Transportation Engineering*, pages 57–78, 2009.
3. F. Bellifemine, A. Poggi, and G. Rimassa. JADE–A FIPA-compliant agent framework. In *Proceedings of PAAM*, volume 99, page 33. London, 1999.
4. M. Bratman. *Intentions, plans, and practical reason*. Harvard Univ. Press, Cambridge, 1987.
5. P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
6. A. Grignard, P. Taillandier, B. Gaudou, D. Vo, N. Huynh, and A. Drogoul. GAMA 1.6: Advancing the art of complex agent-based modeling and simulation. In *PRIMA 2013: Princ. and Practice of Multi-Agent Systems*, volume 8291 of *LNCS*, pages 117–131. Springer, 2013.
7. N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas. JACK intelligent agents-summary of an agent infrastructure. In *5th International conference on autonomous agents*, 2001.
8. V. M. Le, B. Gaudou, P. Taillandier, and D. A. Vo. A new BDI architecture to formalize cognitive agent behaviors into simulations. In *KES-AMSTA*, volume 252 of *Frontiers in Artificial Intelligence and Applications*, pages 395–403. IOS Press, 2013.
9. K. L. Myers. User guide for the procedural reasoning system. *SRI International AI Center Technical Report. SRI International, Menlo Park, CA*, 1997.
10. D. K. Nhan, N. H. Trung, and N. V. Sanh. The Impact of Weather Variability on Rice and Aquaculture Production in the Mekong Delta. In M. A. Stewart and P. A. Coclanis, editors, *Environmental Change and Agricultural Sustainability in the Mekong Delta*, number 45 in Advances in Global Change Research, pages 437–451. Springer Netherlands, 2011.
11. M. of Natural Resources and Environment. Detailing the establishment, regulation and evaluation planning, land-use planning, 2009.
12. A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A BDI reasoning engine. In *Multi-agent programming*, pages 149–174. Springer, 2005.
13. R. Rönnquist. The goal oriented teams (gorite) framework. In *Programming Multi-Agent Systems*, pages 27–41. Springer, 2008.
14. I. Sakellariou, P. Kefalas, and I. Stamatopoulou. Enhancing NetLogo to simulate BDI communicating agents. In *Artificial Intelligence: Theories, Models and Applications*, pages 263–275. Springer, 2008.
15. D. Singh and L. Padgham. OpenSim: A framework for integrating agent-based models and simulation components. In *Frontiers in Artificial Intelligence and Applications-Volume 263: ECAI 2014*, pages 837–842. IOS Press, 2014.
16. P. Taillandier, O. Therond, and B. Gaudou. A new BDI agent architecture based on the belief theory. application to the modelling of cropping plan decision-making. In *iEMSs*, 2012.
17. L. Q. Tri, V. T. Guong, P. T. Vu, N. T. S. Binh, N. H. Kiet, and V. V. Chien. Evaluating the changes of soil properties and landuse at three coastal districts in Soc Trang province. *Journal of Science of Cantho University*, 9:59–68, 2008.
18. V. P. D. Tri, N. H. Trung, and V. Q. Thanh. Vulnerability to flood in the Vietnamese Mekong Delta: Mapping and uncertainty assessment. *Journal of Environmental Science and Engineering B*, 2:229–237, 2013.
19. H. Visser and T. de Nijs. The Map Comparison Kit. *EMS*, 21(3):346–358, 2006.
20. R. Wassmann, N. X. Hien, C. T. Hoanh, and T. P. Tuong. Sea Level Rise Affecting the Vietnamese Mekong Delta: Water Elevation in the Flood Season and Implications for Rice Production. *Climatic Change*, 66(1-2):89–107, 2004.
21. U. Wilensky and I. Evanston. Netlogo. center for connected learning and computer based modeling. Technical report, Northwestern University, 1999.