A cognitive module in a decision-making architecture for agents in urban simulations

Quentin Reynaud, Etienne de Sevin, Jean-Yves Donnart, Vincent Corruble

A cognitive module in a decision-making architecture for agents in urban simulations

Quentin Reynaud[1, 2], Etienne de Sevin[2], Jean-Yves Donnart[1], and Vincent Corruble[2]

[1] Thales Training & Simulation, 1 Rue du Général de Gaulle 95520 Osny
{quentin.reynaud, jean-yves.donnart}@thalesgroup.com
[2] LIP 6, Université Pierre et Marie Curie, 4 place Jussieu 75005 Paris
{quentin.reynaud, vincent.corruble, etienne.de-sevin}@lip6.fr

Abstract. This paper addresses the issue of hybridization between reactive and cognitive approaches within a single decision-making architecture for virtual agent in an urban simulation. We use a reactive module in order to manage reactive behaviors and agent autonomy, and a cognitive module for anticipation, learning and complex behaviors management. The purpose of the cognitive module is to increase the agent's behavior credibility. The agent's reactive and proactive behaviors are sent to a decision module which is able to integrate, decompose, combine and select an action.

Keywords: Cognitive module, agent architecture, decision making

1 Introduction

Virtual agents simulation is a very active area of research, and finds more and more applications. In this paper we focus on urban simulation, which has a wide range of applications: video games (particularly serious games), urban planning, transportation systems, security…

This work is being carried out within the context of TerraDynamica[1], a collaborative project aiming at building an artificial intelligence framework for the simulation of human-like agents in virtual urban environments. Because of the wide range of applications targeted in this project, we need a flexible agent architecture, fit for several domains. Depending on their objectives, urban simulation designers have to take into account several issues:

- Scalability: a great number of agents might be required in the simulation [1]
- Real-time: agent's response time could be critical [2]
- Rich environments: large cities are complex systems, because of their dynamicity and wide range of interactions [3]

This motivated our initial orientation towards reactive agents because of their low cost in terms of computational resources. It is also easier to simulate a great number

[1] http://www.capdigital.com/projet-terra-dynamica

of agents, in real-time and in a rich environment. In fact, nowadays in urban simulation, significant effort is put on the topics of 3D representation and path-planning, including pedestrian motion [4–6] and road traffic [7]. This work falls usually within the "human crowd simulation" trend [8, 9].

Nevertheless, reactive agents are not sufficient in order to exhibit credible and complex behaviors [10], which is a central goal of our work. Therefore, we add a cognitive dimension allowing an agent to adopt more complex behaviors, and to anticipate its future using its knowledge. In this paper, our basic assumption is that to be credible a simulation must "suspend the disbelief" of observers [11, 12]. According to Sengers [13], the observer's ability to make sense of the agent's behavior is fundamental to immerse him in the virtual environment. It is therefore important to reproduce believable and consistent behaviors, like rational behaviors [14] (maximizing performance taking stimuli and knowledge into account), including behaviors resulting from complex reasoning.

The main contribution of this paper is to describe the interest of a cognitive module plugged in a generic agent architecture. That is why we will not provide much details about our decision module and our reactive module. Further papers will present these modules more precisely.

Firstly, we present the state of the art in agent decision-making architectures, then a generic reactive architecture. Next, we give more details about the cognitive module we want to plug in this generic architecture and explain its impact on the decisional process, especially on the behaviors' integration phase. Before concluding this paper we rapidly discuss the architecture's complexity and its consequences on the scalability issue.

2 State of the art

Two main approaches coexist concerning decision-making architectures: the reactive approach and the cognitive one [15, 16]. A reactive agent acts in response to external (and sometimes internal) stimuli. An internal representation can be used but the agent handles no reasoning. A cognitive agent functions by reasoning on a symbolic representation of itself and its environment. More recently, the concept of hybrid architecture has emerged. Its principle is to use both of the two latter approaches in an attempt to combine their advantages. In this section we base our classification on a recent paper by Duch [17]. But this classification is not as clear-cut as it could be, and others remain possible. In particular, the gaming field does not fit well in this classification. It is a dynamic domain which proposes many decision-making architectures using techniques such as finite state machines, behavior trees, or hierarchical reactive planning, in order to permit credible behaviors while limiting complexity, design and run-time cost ([18, 19]).

Initially, reactive architectures were developed to model simple behaviors for robots. Later, emerged the "animat" approach, which took inspiration from

ethological models [20, 21]. Reactive architectures are strongly related to the bottom-up trend, which advocates the thesis that intelligence can spring from cooperation between simple modules. A good example of this trend is Brooks' subsumption architecture [22]. In this hierarchical architecture, several modules (each in charge of a specific behavior) judge the suitability of their own activation. To avoid conflicts, the modules are strictly prioritized: a high level module inhibits all lower level modules.

Maes proposed a system where each behavior decides on its own activation using an activation level [23]. These levels change dynamically and receive bonuses or penalties which are favoring multi-goals, opportunistic behaviors, while avoiding conflicts. DAMN [24] is also an important work based on a voting mechanism. A DAMN agent has one module for each possible behavior (follow a road, avoid obstacles, maintain internal variable…). Each module grades each feasible action, relevant to its interest. The top-rated action is the most relevant to all behaviors: it is selected.

Based on Rosenblatt and Payton's work [25], Tyrrell proposed another way to handle action selection [26]. The central idea is to decompose behaviors into sub-behaviors until "elementary actions level" are reached. At each step, all relevant stimuli are taken into account. An agent does not take any decision before the final elementary action level is computed. The decision-making process is delayed in order to allow the agent to make compromises (actions which are useful for more than one behavior/goal). That model is called "free-flow hierarchy".

The other decision-making modelling approach is the cognitive trend. Two key cognitive architectures are SOAR [27, 28] and ACT-R [29]. They are based on production rules and follow a top-down paradigm. Behaviors are the result of planning functions that use different types of memory.

The BDI approach [30, 31] is currently widely used. A BDI agent is made up of three parts. It contains desires that can be conflicting or can be irrelevant in the current situation. It has beliefs about itself and its environment and uses them to choose intentions, which are helping the agent to accomplish its desires.

Hybrid architectures try to combine the strengths of reactive and cognitive approaches. For example, the InteRRaP architecture [32] separates the decisional process into three steps. The first one is a reactive step: an InteRRaP agent has a set of behaviors, which can respond to its current objective. If none of them matches, the decisional process goes to step two: planning. The agent tries to organize several behaviors in time to reach its goals. If it does not work, the last step is reached: cooperation. The agent tries to contact the others and asks them for help.

The ICARUS architecture [33] was inspired by SOAR and uses four modules. "*Argus*" selectively perceives the environment. "*Daedalus*" plans agent's behaviors (means-end analysis from GPS [34]). "*Meander*" deals with reactive behaviors and executes plans from "*Daedalus*". "*Labyrinth*" stores the agent's knowledge.

The PECS architecture [35] uses four modules too, but they are not organized into a hierarchy. A physical module deals with homeostatic variables, an emotional module is in charge of the agent's emotional state, a social module manages the cooperation between agents and a cognitive module takes care of the knowledge. They are in

permanent conflict in order to take control of the agent. The PECS architecture determines which module is the most relevant to deal with the current situation. Afterwards, that module has to drive the agent. PECS is a winner-takes-all architecture: never more than one module can drive the agent at a single time.

However, in relation with our work, we pay special attention to Maes' bonus/penalty system which is a simple and efficient way to take the global situation into account. We are also particularly interested in the DAMN voting mechanism which is a fair manner to take many expert modules into account (although each module is disconnected from all others). Each hybrid architecture presented has its own way to organize reactive and cognitive modules. An InteRRaP agent uses his "cognitive" module only if the reactive one does not find any solution: the cognitive module can therefore be bypassed. We find this harmful to the agent's behavior depth. In fact, an InteRRaP agent acts always in the same way every time it finds a reactive way to fulfill its goals. ICARUS agent's reactive behaviors can be considered as "reflex". We prefer the PECS modules organization, but PECS is a winner-takes-all architecture. No compromise and no cooperation is possible, a single module takes full control of the agent.

3 Generic reactive architecture

In the following section, we present a generic reactive architecture, inspired by some recent work [36–38]. We stay at a pragmatic level; we do not try to obtain a psychologically realistic model. We want to imitate human cognitive abilities, not human cognition itself (which is a far more complex work). In addition, we want an architecture which can be applied to a great range of applications. A generic reactive architecture based on the perception-action loop was chosen. A high-level reactive module is in charge of behavior proposals, and a decision module arbitrates and select actions (see figure 1). The high-level module manipulates the agent's internal state and updates it in response to the external or internal stimuli. This module should insure agent's autonomy (for example, de Sevin [37] used a motivational module which handles homeostatic variables).

The reactive module sends behavior proposals to the decision module. In the following, we consider that they are equivalent to the agent's desires (a behavior proposal corresponds to a desire the agent wants to fulfill). Let $B_A^t = \{B1, B2, \ldots, Bn\}$ be a behavior proposal for the agent A at time t, where each Bi is a behavior. We define the size of a behavior proposal by the number of behaviors available at the same time. A behavior corresponds to an objective that can be fulfilled by several different sequences of elementary actions (actions that cannot be decomposed).

In the rest of the paper, we use this running example to make explanations more comprehensible:

Let Paul be an agent. ***Drinking*** *is a behavior. To drink, Paul can "****go to the kitchen****,* ***take a glass****,* ***fill it****, and* ***drink out of the glass****" or "****go to the machine****,* ***buy a soda at the machine****, and* ***drink out of the can****", etc.*

Paul is in the kitchen, holding a full glass in his hand. The behavior ***drink*** *can be decomposed in the sub-behavior* ***drink out of the glass****, which is an elementary action.*

The behaviors are not equally important. In order to represent this importance and communicate it to the decision module, the proposed behaviors are prioritized. This priority is a numeric value between 0 and 1: 1 is vital and 0 indicates indifference. The reactive module proposes only behaviors judged as useful. That is why we do not take negative priorities into account. Let P_B^t be this priority for a behavior B at time t.

We also define the dissatisfaction level of an agent A at time t: $D_A^t = \sum_{i=1}^{n} \left(\frac{P_{Bi}^t}{\mathrm{n}} \right)$. We assume that an agent without any desire is satisfied.

Paul wakes up at the morning (t = 0). $B_{Paul}^0 = \{B_1, B_2, B_3\}$*, with* B_1*:* ***go to WC*** *(*P_{B1}^0 *= 0.6);* B_2*:* ***eat*** *(*P_{B2}^0 *= 0.35); and* B_3*:* ***stay in bed*** *(*P_{B3}^0 *= 0.3). His current dissatisfaction level is* $D_{Paul}^0 = \frac{P_{B1}^0 + P_{B2}^0 + P_{B3}^0}{3} \cong 0.42$

The decision module takes several behavior proposals as input, and gives only one elementary action as output. The actual sequence of actions selected by an agent is called its effective behavior. In order to do so, the decision module has to decompose input behaviors into sub-behaviors until elementary action levels are reached. This decomposition depends on the agent's knowledge: the more knowledge the agent has, the wider the decomposition can be. Behavior priorities have to be correctly spread during the decomposition. In particular, the decomposition should take the agent's individuality (preferences, personality, etc.) into account: it can prefer Indian or French food, water or soda, etc.

Once all action priorities are computed, the decision module selects the action with the highest priority: it is the preferred action. The agent starts doing that action.

In this paper we remain at a generic level: we do not present in detail the reactive module or the decision module, because the main contribution of this paper is the addition of a cognitive module. The generic reactive architecture where the cognitive module is plugged is not central and any other architecture fulfilling the constraints that we discussed in this section can be used.

4 Cognitive module

We need to add a cognitive dimension to this generic reactive architecture in order to observe more diverse and advanced behaviors. For that purpose, an additional

cognitive module is plugged at the same level as the reactive module. It works the same way externally: it sends behavior proposals to the decision module, which is able to decompose reactive and proactive behaviors and choose the preferred action in the same manner (see figure 1).

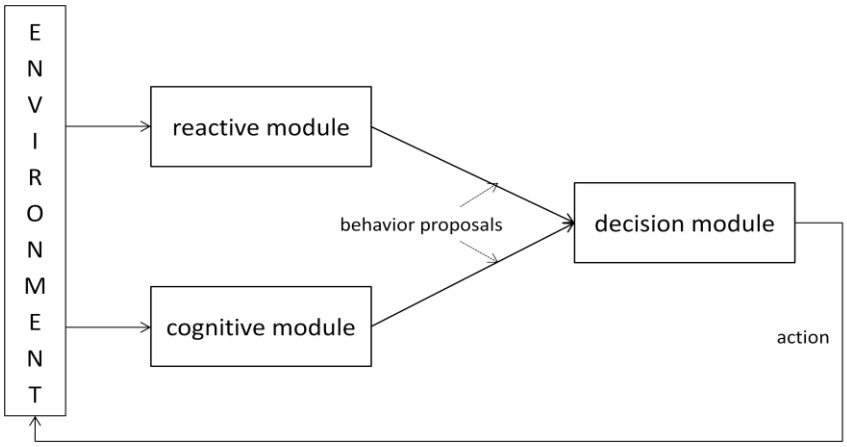

Fig. 1. Schema of the global architecture

In the following sections we present the functionality of our cognitive module. We chose to use this module to give the agent anticipatory [39], learning and planning skills in order to increase the depth and relevance of the agent's behavior [40, 41].

4.1 Anticipatory planning

As mentioned before, our basic architecture is generic: it is composed of a high-level reactive module and of a decision module. From the perspective of the cognitive module, these two other modules are seen as black boxes. The cognitive module knows their inputs and outputs but nothing else. In fact, the cognitive module can access perceptions (same inputs as reactive module), behavior proposals (reactive module's output and decision module's input) and preferred actions (decision module's outputs). Our reasoning module ignores why the agent has desires, or why it prefers doing an action rather than another. However, the cognitive module can use a model of itself, in order to anticipate its inner state. In a similar way, the cognitive module can use a model of the environment, in order to anticipate future events. To go further, the cognitive module would also require a model of agent-environment interactions.

We first consider that all these anticipatory models are provided by the simulation designer, but they can also be built though learning (see section 4.2). Each one of these models can be associated confidence index (CI), expressing the confidence that the agent has in its anticipation skill. The cognitive module uses such models to carry out an "anticipation plan", that is a projection into its future. The cognitive module

tries to anticipate the outputs of the decision module, based on the anticipation of its own future desires.

The "anticipatory planning" process runs asynchronously with the action selection process.

First step: the end of the current action. The cognitive module tries to anticipate the time when the agent ends its current task. To do so, the cognitive module needs knowledge of the action parameters (time before the action finishes) and travel time (time before the next action starts). For the moment, we assume that the cognitive module knows the duration of every action (fixed time) and that it can obtain a good estimation of travel times.

Second step: future desires and dissatisfaction level. The cognitive module tries to estimate the agent's future state (at the end of its current action). It predicts the agent's future internal state and the state of the environment using the models provided as input by the simulation designer. It also needs to simulate the effects of the current action. Using this prediction, the cognitive module tries to infer the agent's future desires (B_A^t*) and dissatisfaction level (D_A^t*). In fact, it predicts the reactive module's behaviors proposal at this future time ($B_A^t* = \{B_1*, B_2*, \ldots, B_n*\}$).

Third step: the search for anticipatory useful behaviors. The cognitive module starts a reasoning process whose purpose is to find (if it exists) a plan giving a lower future dissatisfaction level (i.e. better future satisfaction).

A huge dissatisfaction can be caused by a predictable event: Paul is walking home, when rain starts suddenly to fall, his dissatisfaction increase. If Paul had been aware of that sooner, a better plan for him could have been to take an umbrella before leaving his workplace. In order to have knowledge about the meteorological conditions, Paul can look at the newspapers, or infer them watching at the cloudy sky.

A better plan is in practice the selection of a myopically sub-optimal action because of a possible future reason (taking the umbrella is better than leaving the umbrella because it may rain). We note that if the agent currently knows about the rain, its optimal action is to take the umbrella. It does not need a cognitive module to notice that. These behaviors (called "anticipatory useful behaviors") can in the end be useless, or even harmful as in our example.

Paul goes back and takes his umbrella, but eventually rain does not come.

Fourth step: the sending of anticipatory useful behaviors. If anticipatory useful behaviors are detected, the cognitive module has to determine their priority. It is calculated using the anticipated gain of satisfaction (ΔD) and the confidence index

(CI) of the different anticipatory models used. We notice that CI usually decrease in time: the more distant in time, the less confident the predictions are.

These anticipatory useful behaviors are saved in the cognitive module and sent to the decision module when they are deemed useful. In our current example the behavior "**go back and take the umbrella**" is immediately useful, but it is not always the case, especially if the agent anticipates his distant future. These proactive behaviors are at the same level as the reactive behaviors. The decision module can ignore them if something else is more important (*Paul is in a hurry. He does not really care about the rain*). The decision module keeps complete control over the agent's effective behavior.

Fifth step: the next preferred action. The cognitive module has a prediction of the agent's future desires. This prediction comes from step two (if no anticipatory useful behaviors were found), or from step three (if anticipatory useful behaviors were found). Actually, in order to know if a plan is better than the current one, the cognitive module has to process again steps one and two with the new plan and compare the results. Then, in any case, the cognitive module tries to predict the agent's future decision, i.e. its future preferred action. In order to do so it has two main possible solutions. It can directly use the decision module as a simulator of itself, give it its own behaviors proposal prediction as input, and get its output back. Or it can use a model of the decision module to do the same thing (that option is not detailed in this paper). At the end of this step the cognitive module should have an estimate candidate for the future preferred action (called C_A^t*).

The process can iterate by going to step one again. We define the depth of that process as the number of iterations. We notice that if the effective choices of an agent are not the choices predicted by the cognitive module, this module has to update its predictions. The unpredictability of the future is represented by the decrease of the CI in time. An agent has no advantage in anticipating too many actions in the future, because his anticipatory useful behaviors' priority would get close to zero.

4.2 Anticipatory learning

Using the anticipatory planning process, the agent should exhibit more consistent behaviors. It should be able to think about the future and adapt its behavior accordingly. This mechanism also provides additional tools to the simulation designer. Indeed, he has the opportunity to associate to an agent different types of anticipatory models, more or less correct, in order to exhibit several kinds of behaviors, more or less consistent. In particular, it would be possible to observe inconsistent behaviors because of false anticipatory models (*Paul thought it would rain today, so he decided to take his umbrella; but the weather ended up being totally fine*). For a simulation designer, it is a great opportunity to introduce variety in the agent's behavior (in addition to personality and personal preferences).

An anticipatory learning process is rapidly presented here. It consists in giving to the agent the ability to learn (or complement) their anticipatory models themselves, using techniques inspired from machine learning. If the anticipatory planning gives the agent the ability to take into account future events, the anticipatory learning gives to an agent the ability to take into account the surprise generated by the actual observed events so as to refine its predictions (see [42] for a similar process).

This ability simplifies the simulation designer's work (he does not have to give to each agent any anticipatory model). Two models are concerned: the reactive module model and the environment model. The learning process uses the behavior proposals to improve the reactive module model and external stimuli to improve the environment model. For example, using this learning process, an agent will be able to infer the evolution of its own motivations without any basic knowledge. In the following section, the main example (Paul's hunger) is chosen because it is simple to understand, and not because it would be the most relevant (indeed, it mostly boils down to learn something which has been given as input).

The anticipatory learning process is composed of three generalization levels:

- The first level of anticipatory learning is to collect facts about an agent without interacting with its behavior (*Paul eats at 8a.m, then at 13p.m, then at 20 p.m*).
- The second level is to search for patterns among these facts. These patterns can be detected using the number of occurrences in a period of time (*Paul is hungry three times a day*), an hour of appearance (*Paul eats at 20p.m.*), a location, etc.
- The third level is to infer generic knowledge from the second level. The cognitive module has to abstract the patterns from their surrounding context (*each day, Paul wants to eat at 13p.m.*, or *Paul wants to eat three times per day*). In fact, the anticipatory learning process accesses the agent's desire priorities. It can also infer numeric rules by watching the progression of the desires (*every hour without food intake, Paul's hunger increases by 0.1*).

Given that CI depends on the number of consistent observations (facts according to a prediction), high quality predictions should be automatically stored and low quality predictions should be automatically eliminated. This learning process allows an agent to get its own anticipatory models, which can be more or less correct, and which are used by the anticipatory planning (4.1) and impact the agent's behavior.

4.3 Complex behaviors management

In parallel to its anticipation role, the cognitive module can also be in charge of complex behaviors. Indeed, in order to improve credibility, we want agents to be able to exhibit complex behaviors, e.g. a "mission" in a military tactical sense. A complex behavior is also the specification of many high-level objectives and sub-objectives (i.e. high-level behaviors and sub-behaviors) with failure and success conditions. These conditions can have levels, indicating the force of failure or success. The articulation between (sub-) behaviors can be dynamic and adapted to the effective sequence of events. In order to be integrated in the agent's decisional process, complex behaviors and sub-behaviors have to be prioritized like other behaviors. In

the scope of that paper, we assume that these missions are given by the simulation designer. The capacity to manage complex behaviors relies on the previously seen anticipation capacity (within the capacity to anticipate, an agent improves its management of complex tasks), but in an unpredictable future (the success of the mission is not certain). It is another tool for the simulation designer, in order to describe the simulation scenario.

Paul is not the common citizen he seems to be. In fact he is a dangerous terrorist. During the afternoon, his mission ***plant a bomb*** *is activated. It consists in* ***taking the bomb****, which is hidden under his desk, then* ***choosing a crowded location****,* ***planting the bomb without being seen****,* ***escaping****, and* ***hiding****. At any time, if he has the impression that he is being followed or spotted, his mission is aborted and he has to* ***hide****. Whatever the result of the mission, after a while, he* ***calls his accomplice*** *and* ***meets him****.*

That kind of mission cannot be simply decomposed into independent sub-behaviors. Additional information is required: failure and success conditions, goal updates, etc. In order to achieve these kind of complex behaviors, the agent has to be able to follow many heterogeneous sub-behaviors (in parallel or sequentially), during a long time frame. Instead of making the decision module more complex, in order to make it able to manage complex behaviors; we choose to locate this capability into the cognitive module. That choice has a double advantage. We keep the behavior description language as simple as possible and we are able to manage a hierarchical behavior system such as a hierarchical task network [43], which reduces the complexity.

In the particular case of complex behaviors, because of the long time frame considered and of the impossibility to predict the future (Paul has no way to know if he will be detected or not), it is easier to think about sub-tasks instead of whole mission. The key point is to isolate behaviors that are immediately useful from behaviors that are not. In our example, Paul does not need to think about his meeting with his accomplice until the last moment. Furthermore, he cannot know the time and the location of the meeting in advance. In contrast, while Paul is looking for a good location for the bomb, he could be looking for a hideout.

An important role for the cognitive module is to arbitrate between behaviors that must be sent to the decision module and those that must stay inactive. The complexity of the mission is also hidden to the decision module, and stays in the cognitive module.

Back to our example: During the first part of the mission, the decision module receives two behaviors from the cognitive module: ***find a location for the bomb*** *and* ***find a hideout****. The first one comes directly from the description of the mission, but the second one is not present in the mission. The cognitive module infers its "anticipatory usefulness".*

Priority of cognitive behaviors proposals are here derived from the mission and sub-mission priority.

5 Impact on the decision module

The decision module takes behavior proposals as inputs and gives an elementary action as output. In between, it has four steps to take care of: the integration of heterogeneous behaviors, the decomposition into elementary actions, the combination into new behaviors, and the selection of the preferred action. We rapidly discussed the three last points in section 3, so the interesting point left is the integration.

One could think that the behavior's proposal formalism shared by the two high-level modules is not sufficient to compare directly their behavior's priorities in the decision module. Indeed, the high-level modules could not be sharing a common set of values. The "0" and "1" priorities are known, but the scale between them is not specified. One could also want to ensure that no module is going to take precedence over the other. In order to do so, one could want to normalize the behavior's proposal priorities, by applying to each module a coefficient that equalizes (on average) the behavior proposals' priorities. In the same vein, one could want to verify that the standard deviation is equivalent. If not, a high-level module that always proposes two behaviors, one with very high priority and the other with very low priority, has the same average priority than a module that always proposes middle-prioritized behaviors; but the first module always drives the agent's behavior. That way, we are certain that the high-level modules share a common set of values, and that the priorities they send are comparable.

That is not the choice we made. There are two main reasons for that. Firstly, simulation designers could want, in a specific situation or for a given application, to model agents that are mostly driven by the same high-level module. In the same way that there are people who are provident and others who are inadvertent, there could be agents who listen more to their cognitive or reactive module. This is a way to personalize agents, but can also be a way to adapt them to the application needs. Secondly, we do not think that it makes sense to try to find a common set of values between these modules. The heterogeneous behaviors' priorities do not have the same meaning. Our pragmatic reason is that there is no objective comparison between a plan we made and a reactive desire we suddenly feel. The only possible comparison is subjective, determined by the personality or the current situation.

Consequently, we only give a weight to each high-level module that depends on the personality of the agent. That weight increases or decreases all the priorities sent by the high-level module. The current situation, for its part, is taken into account during the decomposition process (see section 3). The next step is to perform learning about these weights (detect harmful behaviors and adjust the related weight).

In our architecture, we made the choice to combine reactive and cognitive behaviors. That places us in the hybrid approach. As we have seen in the related work section, many hybrid architectures are possible.

- The action selection process can be decomposed into several steps as in the InteRRaP architecture
- The reactive and cognitive modules can be organized in a hierarchical manner, as in the ICARUS architecture
- They can be at the same level, as in the PECS architecture.

We selected this last form of organization because it allows us to introduce a "voting mechanism" as in the DAMN architecture (here there are weighed priorities instead of grades), in order to take all high-level modules into account. In addition, by placing the cognitive module at the same level as the reactive one, its behavior proposals are at the same level too. The decision module is able to find compromise actions between reactive and cognitive behaviors. That way, we are able to overcome limits from other hybrid architectures. Reactive and cognitive behaviors can be created in parallel and no module can be bypassed (as opposed to the InteRRaP architecture); the arbitration between reactive and cognitive behaviors is outside the reactive module, not any module is a priori preferred (as opposed to ICARUS); and our architecture is not working in a winner-takes-all manner (as opposed to PECS).

6 Discussion and conclusion

Despite its benefits, the previously presented cognitive module increases the complexity of our architecture. It is necessary to find ways for our architecture to therefore deal with the urban simulation's scalability objective. Our central idea here is to create several dynamical types of agents, according to their importance (defined by the simulation designer) or their location (visibility from an observer). The cognitive module can be inactivate (or simplified) for some types of agent. This method, related to a level of detail artificial intelligence approach [44], allows us to allocate in priority the computing resources to the most important agents. The main objective is to save computing resources without losing agent's behavior credibility.

In this paper, we propose a decisional architecture for virtual agents in urban simulation. We have huge constraints: we want to be able to simulate a great number of agents (scaling up) in a real-time fashion, into a large and complex environment. In addition, our architecture must fit various application domains (video games, urbanism, transports, security, etc); we also try to model credible agents. In order to do so, we proposed an architecture based on the collaboration between a reactive high-level module (in charge of reactive behaviors and of the autonomy of the agent) and a cognitive module (which allows the agent to have anticipation and learning skills and manages complex behaviors). These modules propose behaviors to a decision module in charge of the arbitration between them and the action selection. That brings us to a new kind of hybrid approach, without hierarchy between reactive and cognitive behaviors and allowing compromise behaviors.

The model we described is currently being integrated into the TerraDynamica platform. The next step for us is to test and validate this model. We have many perspectives in mind. In the short term, we want to improve our decisional process in

order to take agendas into account. These agendas impact also the anticipatory planning. In the longer term, we want to generalize the interface between high-level module and decision module. This gives us the possibility to design an agent with various high-level modules. Of course, they have to respect our formalism, but theoretically, any type of module (and any combination between them) could be accepted. In particular, affective and cooperative modules would be interesting, as the agent's effective emergent behavior would gain in depth. Finally, we plan to add in the cognitive module a mechanism able to memorize plans that the agent constructs. The interest of the plan has to be verified in order to avoid the storing of uninteresting one. A mechanism for comparing current and stored goals, and for adapting stored plans to the current situation is also needed.

7 References

1. Ceranowicz, A., Torpey, M.: Adapting to urban warfare. The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology. 2, 3–15 (2005).
2. Jepson, W., Liggett, R., Friedman, S.: An environment for real-time urban simulation. Proceedings of the 1995 symposium on Interactive 3D graphics - SI3D '95. p. 165-ff. ACM Press, New York, New York, USA (1995).
3. Waddell, P., Ulfarsson, G.: Introduction to urban simulation: design and development of operational models. Handbook in Transport. 5, 203 - 236 (2004).
4. Werner, T., Helbing, D.: The social Force Pedestrian Model Applied to Real Life Scenarios. 17 - 26 (2003).
5. Shao, W., Terzopoulos, D.: Autonomous pedestrians. Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '05. p. 19. ACM Press, New York, New York, USA (2005).
6. Ronald, N., Sterling, L., Kirley, M.: An agent-based approach to modelling pedestrian behaviour. International Journal of Simulation. (2007).
7. Bloomberg, L., Dale, J.: Comparison of VISSIM and CORSIM Traffic Simulation Models on a Congested Network. Transportation Research Record Journal. 98104, 52-60 (2000).
8. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Transactions on Graphics. 25, 1160 (2006).
9. Maïm, J., Yersin, B., Thalmann, D.: Real-time crowds. SIGGRAPH Asia 2008. pp. 1-16. ACM Press, New York, New York, USA (2008).
10. Ochs, M., Sabouret, N., Corruble, V.: Simulation of the Dynamics of Nonplayer Characters' Emotions and Social Relations in Games. IEEE Transactions on Computational Intelligence and AI in Games. 1, 281-297 (2009).
11. Coleridge, S.: Biographia Literaria: Or, Biographical Sketches of My Literary Life and Opinions. (1872).
12. Bates, J.: The role of emotion in believable agents. Communications of the ACM. 37, 122-125 (1994).
13. Sengers, P.: Designing comprehensible agents. Proceedings of the 16th international joint conference on Artificial intelligence - Volume 2. pp. 1227-1232. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999).
14. Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach. Prentice hall (2010).

15. Bryson, J.: Hierarchy and sequence vs. full parallelism in action selection. From animals to animats. (2000).
16. Langley, P., Laird, J.E., Rogers, S.: Cognitive architectures: Research issues and challenges. Cognitive Systems Research. 10, 141–160 (2009).
17. Duch, W., Oentaryo, R.J., Pasquier, M.: Cognitive Architectures: Where do we go from here? Proceeding of the 2008 conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference. pp. 122–136. IOS Press (2008).
18. Orkin, J.: Three states and a plan: the AI of FEAR. Game Developers Conference. (2006).
19. Isla, D.: Handling complexity in the Halo 2 AI. Game Developers Conference. (2005).
20. Meyer, J.A.: Artificial Life and the Animat Approach to Artificial Intelligence. Artificial Intelligence, Boden,(Ed). 325–354 (1996).
21. Guillot, A.: The animat contribution to cognitive systems research. Cognitive Systems Research. 2, 157-165 (2001).
22. Brooks, R.A.: A Robust Layered Control System For a Mobile Robot. Massachusetts Institute of Technology, Cambridge, MA, USA (1985).
23. Maes, P.: Situated agents can have goals. Robotics and Autonomous Systems. 6, 49-70 (1990).
24. Rosenblatt, J.: DAMN: A Distributed Architecture For Mobile Navigation - Thesis Summary. Journal of Experimental and Theoretical Artificial Intelligence. pp. 339-360. AAAI Press (1995).
25. Rosenblatt, J.K., Payton, D.W.: A fine-grained alternative to the subsumption architecture for mobile robot control. International Joint Conference on Neural Networks. pp. 317-323 vol.2. IEEE (1989).
26. Tyrrell, T.: The use of hierarchies for action selection. Adaptive Behavior. 1, 387 (1993).
27. Laird, J.E., Newell, A., Rosenbloom, P.S.: SOAR: an architecture for general intelligence. Artif. Intell. 33, 1-64 (1987).
28. Laird, J.E.: Extending the Soar Cognitive Architecture. Proceeding of the 2008 conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference. pp. 224-235. IOS Press, Amsterdam, The Netherlands, The Netherlands (2008).
29. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. psychological review. 111, 1036-1060 (2004).
30. Rao, A.S., Georgeff, M.P.: BDI Agents: From Theory to Practice. in proceedings of the first international conference on multi-agent systems (ICMAS-95). pp. 312-319 (1995).
31. Georgeff, M., Pell, B., Pollack, M., Tambe, M., Wooldridge, M.: The belief-desire-intention model of agency. Intelligent Agents V: Agents Theories, Architectures, and Languages. 1–10 (1999).
32. Müller, J.P., Pischel, M.: The Agent Architecture InteRRaP: Concept and Application. (1993).
33. Langley, P., Choi, D.: A unified cognitive architecture for physical agents. proceedings of the 21st national conference on Artificial intelligence - Volume 2. pp. 1469-1474. AAAI Press (2006).
34. Newell, A., Simon, H.A., rand corp Santa Monica Calif.: GPS, a program that simulates human thought. Defense Technical Information Center (1961).
35. Schmidt, B.: Human factors in complex systems□: The modelling of human behaviour. Simulation in wider Europe, 19th European Conferance on Modelling and Simulation. pp. 5-14 (2005).

36. Robert, G., Guillot, A.: MHiCS, a modular and hierarchical classifier systems architecture for bots. 4th International Conference on Intelligent Games and Simulation (GAME-ON'03). pp. 140–144 (2003).
37. de Sevin, E.: An action selection architecture for autonomous virtual humans in persistent worlds, (2006).
38. Dujardin, T., Routier, J.-C.: Behavior Design of Game Character's Agent. 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. pp. 423-430. IEEE (2010).
39. Robert, R.: Anticipatory Systems. (1985).
40. Meyer, C., Ganascia, J.G., Zucker, J.D.: Learning strategies in games by anticipation. international joint conference on artificial intelligence. pp. 698–707. lawrence erlbaum associates LTD (1997).
41. Butz, M., Sigaud, O., Gérard, P.: Anticipatory Behavior in Adaptive Learning Systems. Springer Berlin Heidelberg, Berlin, Heidelberg (2003).
42. Burke, R.: Great expectations: Prediction in entertainment applications. Life-Like Characters Tools, Affective Functions, and Applications. (2004).
43. Erol, K., Hendler, J., Nau, D.S.: UMCP: A sound and complete procedure for hierarchical task-network planning. Proc. 2nd Intl. Conf. on AI Planning Systems. pp. 249–254 (1994).
44. Navarro, L., Flacher, F., Corruble, V.: Dynamic level of detail for large scale agent-based urban simulations. The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. pp. 701–708. International Foundation for Autonomous Agents and Multiagent Systems (2011).