Control Your Virtual Agent in its Daily-activities for Long Periods

Lysa Gramoli, Jérémy Lacoche, Anthony Foulonneau, Valérie Gouranton, Bruno Arnaldi

Control Your Virtual Agent in its Daily-activities for Long Periods

Lysa Gramoli[1,2(✉)], Jérémy Lacoche[1], Anthony Foulonneau[1], Valérie Gouranton[2], and Bruno Arnaldi[2]

[1] Orange, Rennes, France
{jeremy.lacoche,anthony.foulonneau}@orange.com
[2] Univ Rennes, INSA Rennes, Inria, CNRS, Irisa, Rennes, France
lysa.gramoli@irisa.fr, {valerie.gouranton,bruno.arnaldi}@irisa.fr

Abstract. Simulating human behavior through virtual agents is a key feature to improve the credibility of virtual environments (VE). For many use cases, such as daily activities data generation, having a good ratio between the agent's control and autonomy is required to impose specific activities while letting the agent be autonomous. This is why we propose a model allowing a user to configure the level of the agent's decision-making autonomy according to their requirements. Our model, based on a BDI architecture, combines control constraints given by the user, an internal model simulating human daily needs for autonomy, and a scheduling process to create an activity plan considering these two parts. Using a calendar, the activities that must be performed in the required time can be given by the user. In addition, the user can indicate whether interruptions can happen during the activity calendar to apply an effect induced by the internal model. The plan generated by our model can be executed in the VE by an animated agent in real-time. To show that our model manages well the ratio between control and autonomy, we use a 3D home environment to compare the results with the input parameters.

Keywords: Autonomous agent · Daily activity model · Scheduling · Control over the agent's decision-making autonomy

1 Introduction

Reproducing human behavior through virtual agents is an essential feature to improve the usefulness of virtual environments (VE). To increase the credibility of agent's behaviors, some human processes can be simulated such as human needs, decision-making, or preferences. With such features, a virtual agent can reason and adapt its behavior according to the situation and its internal state. As a result, it becomes more autonomous in its decision-making process as per the definition of autonomy provided by Avradinis et al. [3].

In this paper, we particularly focus on an agent model allowing the generation of daily routines in indoor environments. Such a model could be useful for virtual environment developers and researchers using simulations to generate new databases. Indeed, it could integrate and improve existing simulators allowing the generation of data for "indoor inhabitant understanding" based for example on computer vision as in VirtualHome [17] or on data from connected objects such as OpenSHS [1]. In such use cases, they may want to impose specific activities to execute in the VE while maintaining an agent autonomous process outside these periods. Therefore, a model adjusting the degree of autonomy and being able to execute activities in a VE is required to generate accurate, diversified and credible datasets. To go further, Suggesting a way to interrupt specified activities is also important to make the generated behaviour more credible.

However, being able to modify the agent's level of decision-making autonomy with the same model is quite rare in the literature. Current approaches are based either on agents reacting to the situation but facing the challenge of respecting strong constraints, or agents being able to plan but facing the challenge of adapting to the situation. Yet, it is important for us to ensure that the activities required by the user are performed on time with the correct duration, regardless of what the agent was doing before.

To address this issue, we propose a model adjusting the degree of agent's decision-making autonomy according to the user's will. Our model, based on a BDI architecture [21], combines an internal state model simulating human daily needs, a module giving the user's constraints through a calendar, a scheduler guiding the decision-making according the both previous parts, and an execution process performing activities in a VE. Moreover, an interruption mechanism is also introduced in the scheduler so that the agent can interrupt, delay or shorten activities indicated by the user. To demonstrate our model, we propose to simulate daily routines in a 3D house environment which could be adapted to our identified use cases, shown in Fig. 1. As our model is oriented towards decision making and activity execution, work focusing on the production of verbal agents is not covered in this paper. The paper is organized as follows. In Sect. 2, we propose some related work. In Sect. 3, we explain our model and its main process implied in the management of autonomy. Finally, the Sects. 4 and 5, are used for the results and the conclusion.

Fig. 1. Global view of the 3D simulator and the agent performing activities

2 Related Work

Approaches to simulate human behavior can be divided into many classes such as reactive-based methods, plan-based methods and learning-based methods.

Regarding reactive-based methods, the agent chooses and executes the most appropriate action according to the current context. They are particularly well suited to dynamic environments and multi-agent issues. One well-known action-based method is Beliefs-Desires-Intentions (BDI) [21]. In BDI, a perception system interpreting the state of the world is modeled through beliefs, the choice of possible goals is modelled through desires, and the choice of predefined sequences of actions to satisfy these goals is done by intentions. In this category, we can also find action selection mechanisms such as the work of De Sevin and Thalmann [19] and MAGE [3] where the level of motivations (or needs) is used to select activities. These approaches stay limited on the control over the agent's decision-making since the activities are chosen in reaction to the VE or the agent's internal state including motivation or needs.

In contrast to reactive-based methods, other approaches use plan-based methods. In this case, the agent sets up a plan according to the current context and constraints. Among the best known methods, we can find Hierarchical Tasks Network (HTN) [8], STRIPS [7] or meta-heuristics such as Genetic algorithms [5]. This category is interesting to schedule activities and to control the agent's choices. However, they have some limitations regarding the reactivity to the change of the environment and the internal state. In our case, only using a scheduler is not sufficient because we want to simulate the agent's internal state for its autonomy as well as interruptions when they cannot be satisfied otherwise.

Finally, recent approaches use learning-based systems to simulate daily activities, such as LIDA [13], the work of Jang et al. [9] or rules-based approaches like Soar and ACT-R models [10]. In the work of Jang et al. a double Deep Q-Network (DQN) structure is used to find the most appropriate goals according to the agent's needs, the input real data and the time. For the Rules-Based approaches, they use memory systems and explicit rules to construct the decision-making process. All these approaches are promising, but they are limited in terms of the behavior extensibility and control, due to their nature.

The three category described above provide autonomous agents which are free to make their own decisions during all the simulation. However, these solutions are not really focused on the problematic of offering different level of autonomy. Thus, the control that we can have over them is insufficient for our case. Therefore, it would be more relevant for us to get close to approaches combining several categories such as the work of de Silva [20] where BDI and HTN are mixed even though our problematic is not still addressed in their work. There is also the work of Azvine et al. [4] where an intelligent assistant is proposed to help the user with communication, information and time management. This multi-agent system includes reactive scheduling methods to manage various level of time constraints. Thus, a good ratio between control and autonomy can be reached with this system. However, it is difficult to use it in our context, since the proposed model is a user-oriented approach and the use cases are distinct.

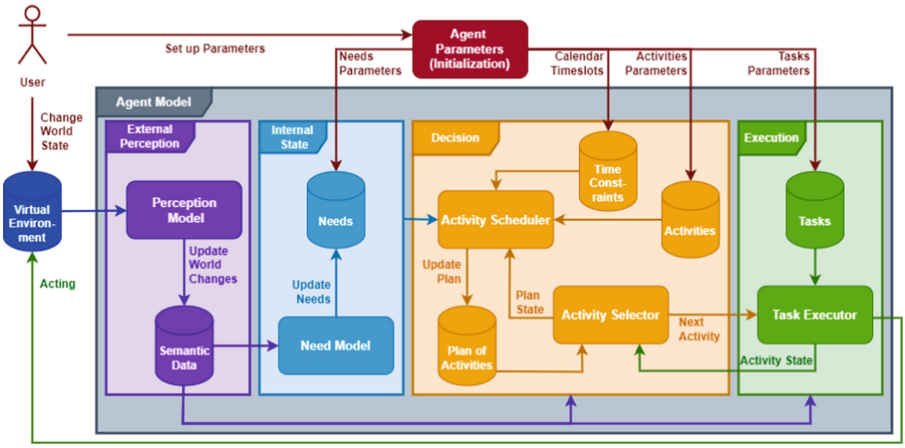

Fig. 2. Our proposed agent model

In addition, this model does not execute actions in a 3D environment. Other approaches also started to address our problematic This is the case of SMACH [2,18] and the work of Ordoñez [15]. For SMACH, the choice of activities is based on probabilities. Consequently, strong constraints have not formal guarantees of being kept. For the work of Ordoñez, the algorithm is run on a user-defined time windows. Unfortunately, this model does not consider the internal state and the effects of activities occurring in another time window. Finally, for both papers, activities are not executed in a 3D Simulator. This is thus limiting for our use cases since the visual aspects are essentials if we want to simulate sensors (cameras or connected objects). This is why, we propose a model allowing the user to adjust the level of decision-making autonomy and allowing to perform activities in a 3D virtual environment.

3 Agent Model Description

3.1 Global Model Structure

Our model structure is inspired from the BDI models [21] where some adjustments were made to consider our requirements regarding the management of the agent's autonomy. BDI was chosen because of its compatibility with our requirements and for its intuitive approach of the human decision model. Our proposed model is configured to adapt the level of decision-making autonomy according to the user's will and to execute activities in the VE. Using a calendar, the user can give activities that must be performed in the required time. Moreover, interruptions can be allowed to interrupt these activities. In this paper, an activity is a concrete formulation of the way to satisfy a goal or a need. For example, if the agent wants to improve its hygiene, then "Showering" can be an activity. Figure 2 shows the global structure made of the following processes:

Agent Parameters: It manages the initial parameters. A calendar can be given in input to provide activities that must be performed. The user is also able to configure activities, needs and tasks.

Internal State Model: It can be related to desires in BDI. However, its name differs because it could include other cognitive factors than desires. For now, it is used to update the urgency of needs. Needs are inspired by human needs defined in the Maslow's theory of needs [12]. They can be physiological such as hunger or they can be more elaborated like self-esteem. More details are given in Sect. 3.2.

Decision-Making Model: At the heart of the decision-making, it can be related to intentions in the BDI process. However, our model does not retrieve predefined plan according to the situation but it builds the plan of activities during the simulation. Our model adjusts the agent's autonomy to respect the user's constraints while producing autonomous behaviours through the internal state. Our scheduler is designed to be able to reschedule at any time, so it is compatible with dynamic environments. More details are given in Sect. 3.3.

Task Execution Model: This model executes the selected activity in the VE by executing the related sequences of tasks. A task is made of basic actions and animations that can be executed in the VE. For instance, a task can be "Getting dry" for the activity "Showering". This model receives from the decision-making model the activity to perform. In exchange, the activity state is returned. More details are given in Sect. 3.4

External Perception Model: This model can be related to Beliefs in BDI, since all the useful data from the VE are stored in its semantics database. It is used to filter activities according to their available resources and to provide the semantics needed to make the execution possible in the VE.

3.2 Agent Internal State

The internal state model provides essential information about the agent's internal conditions. This information will be then used in the decision-making model for autonomy phases and interruptions. For now, only needs are described here, but other factors such as preferences or emotions could be included.

According to the Maslow's pyramid of needs [12], needs are ordered according to their level of urgency: basic but imperative needs are distinguished from more elaborate needs, which are more complex but less urgent to satisfy. To integrate this approach in our work, a value $\text{Pyra} \in \{1, \ldots, 5\}$ is assigned to each need, where 1 corresponds to basic needs and 5 to the most elaborate needs. For this paper, we explain our function-based approach evolving through the time, as in the work of De Sevin and Thalmann [19]. However, the process to calculate

needs urgency can be made with other approaches as in the work of Jang et al. [9] where a fuzzy-logic approach is used. For now, The computation of the level of urgency called P_{Need} is made of 3 steps:

(1) Need threshold Initialization Th_{Need}: this threshold is used to determine when a need becomes urgent to satisfy. Each threshold has a default val $\text{Th}_d \in (0, 1)$ modifiable by the user (in this paper, the default is set to 0.5). At this step, preferences could be used to deviate the threshold from this default value, and thus modifying the time when the need becomes urgent.

(2) Need intensity $i(t)$: Each need has an intensity value evolving through time as shown in Eq. 1. This evolution is modelled as the second half of a parabola bounded in $[0, 1]$, similar to the Need Manager of the work by De Sevin and Thalmann [19]. The start $i(t) = 0$ occurs for $t = t_{\text{start}}$, and the maximum $i(t) = 1$ occurs for $t = t_{\text{end}}$. These two parameters allow us to control the evolution of needs intensity over long periods. Each need has its own $[t_{\text{start}}, t_{\text{end}}]$ interval that can be set in two ways:

- **Specific hours**: they are set so that intensity starts and peaks at specific times. i.e. we can constrain hunger to start at 12 p.m. and peak at 13 p.m.
- **Periods of time**: they are set so that intensity peaks at varying intervals. For example, to simulate the toilet need, which is not constrained by specific hours, we can set time slots for its intensity at regular intervals of 3 h.

In both cases, real data could be used to configure these time slots, such in the work of Jang et al. [9]. The user can configure them through the agent parameters process. When the intensity reaches its maximum, it stays at this value until the need is satisfied. After this, its intensity decreases by the value given by the activity satisfying it.

$$i(t) = \left(\frac{t - t_{\text{start}}}{t_{\text{end}} - t_{\text{start}}} \right)^2 \tag{1}$$

(3) Need priority P_{Need}: the need priority grows in $[-1, 1]$ proportionally to need intensity, considering $Pyra$ and Th_{Need} as shown in Eq. 2.

$$P_{Need} = \frac{i(t) - \text{Th}_{\text{Need}}}{\text{Pyra} \cdot (1 - \text{Th}_{\text{Need}})} \tag{2}$$

This priority could also depend on the satisfaction of another need. For instance, if the agent drinks, then toilet need could evolve faster.

3.3 Decision-Making Model

The decision-Making model is the main process to control the level of the agent's decision-making autonomy according to the user's constraints. Two main process, called activity scheduler and activity selector, are described here.

The Activity Selector retrieves the activities of the plan and runs the scheduler to produce a new plan when there is no more activity to perform. In the latter case, the activity selector relays the start and end times of the plan to

the scheduler. The last executed activity and the one starting just after the plan are also given. These activities surrounding the plan are either calendar activities or special activities only serving to delimit the plan. Thus, the activities calendar are included with the right times while leaving autonomy between them. The activity selector also transfers the selected activity to the task executor and gets its status back.

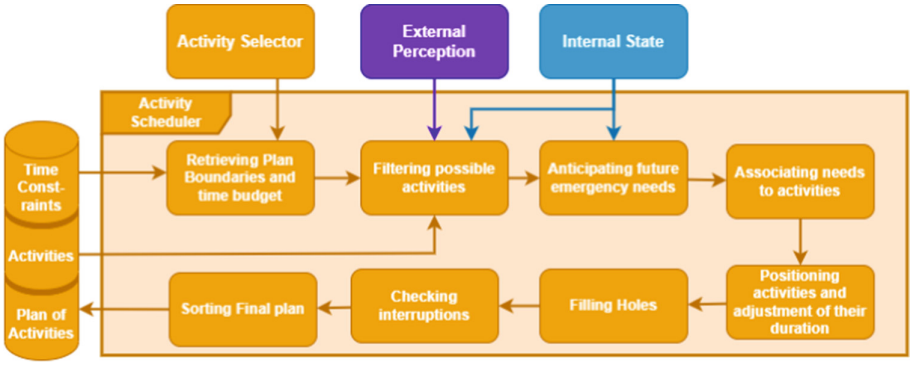

Fig. 3. Main Steps of the activity scheduler

The Activity Scheduler is inspired by the principle of reactive schedulers proposed in particular by the work of Azvine et al. [4]. This process can be relaunched if necessary, making our model compatible with dynamic VE. It has several functions summarized in Fig. 3:

Retrieving Plan Boundaries and Time Budget: This step corresponds to the moment when the agent defines the duration of its free time in relation to the activities already scheduled. The agent will then prepare a plan of activities to be carried out during this free time Concretely, The scheduler retrieves the plan period given by the activity selector. This period is converted into a time budget that must be respected to avoid trimming the future activities of the calendar. The last activity executed and the activity being just after the plan are stored to be considered. They will define what we call the boundaries of the plan. If there is no activity in the calendar after the period, then a special activity called "Activity Boundaries" with a duration of 0 is given by the activity selector to indicate the end of the plan. If the time budget is too short to put any activity, all the next steps until the checking interruption are skipped.

Filtering Possible Activities: During this step, the agent identifies what activity is possible or not during his free time. Activities having a minimum duration exceeding the time budget or reaching its maximum occurrence are excluded. The perception model is also used to exclude activities using unavailable resources. If no activity is found, all the next steps until the Filling Gap are skipped.

Anticipating Future Emergency Needs: In this step, the agent will anticipate its needs for the duration of its free time. To do this, the scheduler identifies the future moments when the needs become urgent. This forecast is limited to the period of the plan. First, the scheduler retrieves the level of urgency of each need at the current time. Then, for each time step, the scheduler launches the process calculating the level of urgency described in the previous Sect. 3.2. The scheduler thus gets the interval of urgency. A time called T_{Need} is retrieved randomly inside this interval. This step is repeated until all possible emergency moments are identified in the duration of the plan. For instance, if the duration of the plan lasts 7 h and that thirst becomes urgent every 3 h, then the algorithm will find 2 moments of urgency which are stored as T_{Need}. After this, the scheduler checks whether the needs are already met by the activity happening just after the plan. In this case, if the time difference between both is below a defined threshold, then the regarded T_{Need} is removed. If no T_{Need} is found during this step, all the next steps until the Filling Gap are skipped.

Associating Needs to Activities: During this phase, the agent tries to reserve activities able to satisfy its needs. After retrieving all the T_{Need} for each need, the scheduler tries to put a possible activity satisfying them. At this step, the start time of the activity is positioned at each occurrence of T_{Need} with its minimum duration. The scheduler starts by placing the activities on the T_{Need} closest in time. When an activity is placed, the minimum duration of this activity is removed from the remaining time budget. This step ends when the time budget is exhausted, in this case all the next T_{Need} are removed, or when all the T_{Need} have been associated to activities. Due to this setting, activities cannot be performed at the same time. For instance, if we have T_{Need} occurring at 1 p.m. for thirst and another at 2 p.m. for hunger, then the scheduler firstly places drinking activity at 1 p.m. and then eating activity at 2 p.m. if the time budget is not exceeded.

Positioning Activities and Adjustment of their Duration: At this step, The agent adapts the duration and the beginning of the activities, in order to be ready at the end of its free time. The scheduler adjusts the start time and duration of each activity so that they are close to their related T_{Need}. It also ensures that the start and end times do not exceed the time window. To do this, the scheduler first goes through the list of T_{Need} in reverse chronological order to temporally shift the activities having their end times arriving either outside of the plan boundaries, or after the start of another activity. Then, the scheduler performs a second run of this list, but in chronological order to shift the activities that have their start time beginning before the start of the plan or before the end of another activity. With these two round trips, activities are sure to start and end within the plan without overlapping with other placed activities. After this, starting with the activity closest in time, the duration of each activity are extended either until its maximum duration or the start of the next activity if the maximum is larger.

Filling Gaps: At this stage, The agent also schedules activities outside of its needs to keep busy. Concretely, time gaps where no activities are scheduled can

appear. Here, gaps are considered as moments when the agent has no constraints coming from the user or its internal state. Therefore, these are periods when the agent can wait if the duration is short or can entertain itself. We thus use activities linked to entertaining or waiting to fill gaps. Of course, these activities can be configured by the user. Concretely, we use activities such as "Watching TV", "Computing", "Waiting" and so on.

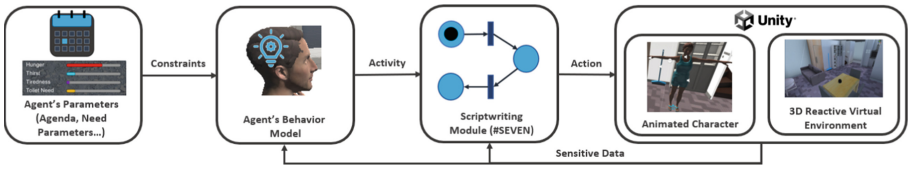

Fig. 4. Global Architecture of the implementation

Checking Interruptions: Sometimes, the agent will also have to interrupt its scheduled activities to satisfy its urgent needs. This is the case when needs could not be satisfied during the plan period. To solve this, the user can indicate whether a need may interrupt a calendar activity when it is urgent. In the same way, it can also indicate what calendar activity can be interruptible. During this step, the scheduler takes the activity situated just after the plan and verifies if this activity is interruptible. In addition, it checks if this activity has a sufficient duration to support a reduction equal to the duration of the interruption. If these conditions are reached, the scheduler checks whether a need that may interrupt and being urgent at the beginning of the plan has a moment of satisfaction in the plan. If it does not, the scheduler creates an interruption. A random moment is retrieved between the start and the end of the interruptible activity. Depending on this moment value, the interruption can starts before, after or during the interruptible activity, without exceeding the initial duration of this activity.

3.4 Task Execution Model

This model executes the current activity in the VE by launching the associated sequence of predefined tasks, containing animations or moves. It communicates with the activity selector to retrieve the activity to perform and returns its status. The perception model is also used to retrieve the needed semantics of the 3D environment so as to correctly perform the activities. For instance, the objects like the door to open or the book to take are given by the perception model. Of course, the user can modify objects used by tasks, as long as the object type and semantics are respected.

For this model, we use existing approaches specialized in the tasks execution. Among them, we can mention petri nets [16], finite state machines [11] or behavior trees [14]. These approaches allow to go from a state to another via

transitions triggering actuators (animations, movements...) while respecting the conditions given by sensors (object state, agent's location...). For our use case, petri-nets were used because some tasks sequences were already provided in this form, but the other approaches can be well integrated instead. The duration of these sequences is automatically synchronized with the activity duration to respect the allocated time. The animation times are also considered in the calculation of the tasks to respect this duration. Figure 1 shows our use case where agents can perform activity in a 3D house environment.

4 Results

In this section, we demonstrate our model by using an agent executing activities in a 3D virtual house. Figure 4 shows the global structure of our implementation that produced the results explained below. To execute tasks in the VE, we use a Petri-Net based scripting module called #SEVEN [6]. *Unity Engine*[1] is used to represent the VE where an animated character as well a 3D reactive environment were imported. This 3D reactive environment mainly contains the 3D virtual house, the semantic data and the algorithm allowing the relationship between the semantic and our behavior model.

More than 20 daily activities were implemented and animated as well as 8 needs to represent the agent's internal states. Some example of activities and needs can be seen in the figures and tables of this part. All the simulations last 8 simulated days. For these simulations, needs and activities were configured

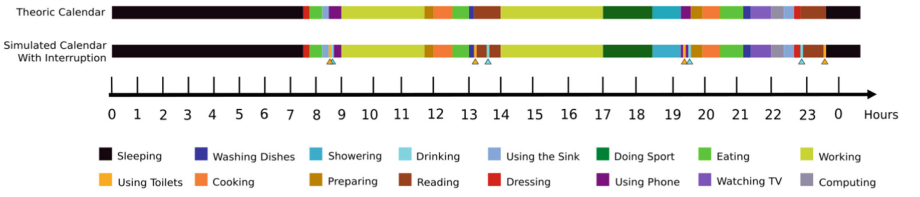

Fig. 5. Comparison between theoretic and simulated timelines for a same day with a strict calendar

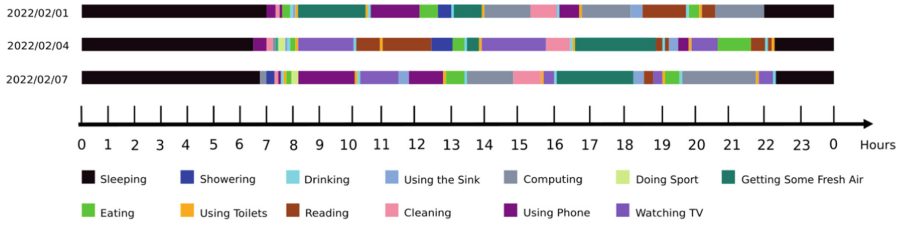

Fig. 6. Timelines of three separated days without theoretic calendar

[1] *Unity Engine*, official website: https://unity.com/.

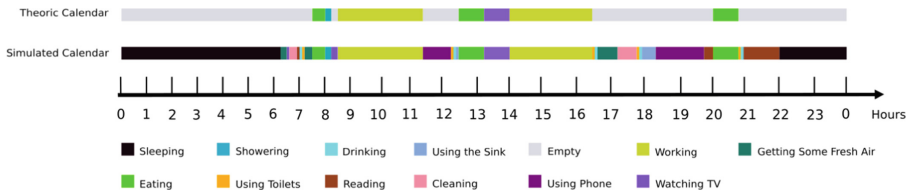

Fig. 7. Comparison between theoretic and simulated timelines for a same day with a moderate calendar

as detailed in Tables 1 and 2 for all these simulations. This section presents results showing the different level of control that the user can have over the agent's decision-making autonomy. Validation is done first on a specific day, then over the long term by analysing several days to obtain quantitative results. The numbers of activities and needs are for illustrative purposes. The user can thus add as many as necessary without impacting the model working. A video of our use case can be viewed at the following link: https://youtu.be/v8GxXCAAV1k

The first result presents the case where the user provides a calendar with no free time. To illustrate it, Fig. 5 compares timelines between a day coming from a theoretic calendar without free time with the results obtained after the simulation of this day. Here, thirst and toilets needs have been authorized to interrupt the calendar. Some calendar activities such as Using Phone, Reading are interruptible. These results show that calendar activities are performed on time and with the right duration. Interruptions are indicated by blue triangles for thirst and yellow triangles for toilets. Thus, the agent is able to interrupt a calendar activity to satisfy its needs allowed to interrupt.

Tables 1 and 2 and the Fig. 6 show the result of a simulation without calendar. They summarize information about satisfaction of needs according to the initial parameters. We can see that needs are satisfied when they are urgent since the average satisfaction over 8 days is globally in the bracket between the beginning of the emergency and its maximum urgency. A better accuracy is also observed when time slots are used rather than periods. Moreover, the frequencies are respected since differences come from the sleep period which approximately takes 8 h. Indeed, in the simulated case, sleep was not allowed to be interrupted, so the nightly satisfaction could not be reached. For the Fig. 6, we can conclude that the periodic satisfaction of needs creates a routine having small variations due to the variety of possible activities and the number of occurrences per day. For instance, eating is made three times per day around the same periods. This variation is interesting to produce more credible behavior.

Figure 7 compares timelines between a day coming from a theoretic calendar containing free times with the results of the same day simulation. Here, the agent must manage its free time in order to be ready for the next required activity. As seen in this timeline, the agent schedules accurately its activities during its free time while taking into account the satisfaction of its needs. The difference between the simulated and the theoretical for the calendar activities is again of

Table 1. Information about the needs configured with timeslots and their associated activities compared to their input configuration

Need name	Related activities	Theoretic time slot	Min/Max start time slot	Mean and standard deviation time slot	Mean Frequency per day (theoretic)	Mean Frequency per day (simulated)	Min/Max Duration (theoretic)	Min/Max Duration (simulated)
Hungry	Eating	[7 a.m., 9.30 a.m.] [12 p.m., 1.30 p.m.] [7 p.m., 9 p.m.]	7.19 a.m./7.48 a.m 12.04 p.m./1.12 p.m 7.04 p.m./8.47 p.m	7.34 a.m. ± 0 h 11 m 12.36 p.m.± 0 h 27 m 7.53 p.m. ± 0 h 37 m	3	3.0	0 h 10 m/1 h	0 h 10 m/1 h
Tiredness	Sleeping	[10 p.m., 12 a.m.]	10.08 p.m./10.40 p.m	10.24 p.m. ± 0 h 12 m	1	1.0	8 h/11 h	8 h/11 h

Table 2. Information about the needs configured with period and their associated activities compared to their input configuration

Need name	Related activities	Theoretic periods	Min/Max gap between 2 satisfactions (except night)	Mean and standard deviation between 2 satisfactions	Mean Frequency per day (theoretic)	Mean Frequency per day (simulated)	Min/Max Duration (theoretic)	Min/Max Duration (simulated)
Thirst	Drinking	Every 3 h (urgency 2 h 05 m)	2 h 13 m/3 h 41 m	2 h 57 m ± 0 h 26 m	8	5.75	0 h 01 m/0 h 05 m	0 h 01 m/0 h 05 m
Hygiene	Showering Using the Sink	Every 6 h (urgency 4 h 02 m)	4 h 25 m/6 h 40 m	5 h 29 m ± 0 h 45 m	4	3.0	0 h 15 m/1 h 0 h 05 m/0 h 20 m	0 h 15 m/0 h 35 m 0 h 05 m/0 h 20 m
Sport	Doing Sport	Every 48 h (urgency 33 h 56 m)	34 h 57 m/37 h 22 m	36 h 10 m ± 1 h 12 m	0.5	0.50	0 h 15 m/2 h	0 h 15 m/0 h 38 m

the same order as the strict calendar, proving that the agent respects the strong constraints. We can also see that there is no redundancy when the calendar activities satisfy needs and are positioned close to the times when these needs are urgent. This is the case with hunger for instance, where the "Eating" calendar activity happens when hunger becomes urgent. The planner also considers the calendar activities to satisfy the agent's needs.

These results show that the agent performs activities in the VE while respecting its input constraints and its needs. We can thus consider that our model is able to adjust the level of autonomy according to the user's configurations and the agent's internal state. The autonomous part satisfies the needs when they

become urgent and respects the schedule given by the user, allowing a good ratio between control and autonomy.

5 Conclusion

We presented a model allowing the user to configure the level of the agent's decision-making autonomy. With our model, combining BDI architecture and scheduling processes, either the user can leave the agent totally autonomous during the simulation, or the activities to perform can be fully or partially controlled by users through a calendar. Interruption can also be used to satisfy the urgent needs of the agent if the calendar is too restrained. In addition, all activities are performed in a 3D virtual environment through animations and movements. Thus, our model is ready to use for data generation: if virtual sensors or simulated cameras are included in the VE, then users could retrieve information about the agent's activities to generate data.

In future work, we will evaluate this model for activity detection with virtual sensors of a smart house. Some experiments will be also made to compare our simulated data with real data. Moreover, the internal state can also be enriched by other cognitive methods such as preferences or emotions. Similarly, perception can be improved to manage resources in addition to activity filtering. We also plan to produce concrete examples showing the management of dynamic VE. After this, our goal would be to extend this model for multi-agent systems.

References

1. Alshammari, N., Alshammari, T., Sedky, M., Champion, J., Bauer, C.: OpenSHS: open smart home simulator. Sensors **17**(5), 1003 (2017)
2. Amouroux, É., Huraux, T., Sempé, F., Sabouret, N., Haradji, Y.: SMACH: agent-based simulation investigation on human activities and household electrical consumption. In: Filipe, J., Fred, A. (eds.) ICAART 2013. CCIS, vol. 449, pp. 194–210. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44440-5_12
3. Avradinis, N., Panayiotopoulos, T., Anastassakis, G.: Behavior believability in virtual worlds: agents acting when they need to. SpringerPlus **2**(1), 1–11 (2013). https://doi.org/10.1186/2193-1801-2-246
4. Azvine, B., Djian, D., Tsui, K.C., Wobcke, W.: The intelligent assistant: an overview. In: Intelligent Systems and Soft Computing, pp. 215–238 (2000)
5. Charypar, D., Nagel, K.: Generating complete all-day activity plans with genetic algorithms. Transportation **32**(4), 369–397 (2005)
6. Claude, G., Gouranton, V., Berthelot, R.B., Arnaldi, B.: Short Paper: #SEVEN, a sensor effector based scenarios model for driving collaborative virtual environment, p. 5 (2014)
7. Fikes, R.E., Nilsson, N.J.: Strips: a new approach to the application of theorem proving to problem solving. Artif. Intell. **2**(3–4), 189–208 (1971)
8. Georgievski, I., Aiello, M.: An overview of hierarchical task network planning. arXiv preprint arXiv:1403.7426 (2014)

9. Jang, H., Hao, S., Chu, P.M., Sharma, P.K., Sung, Y., Cho, K.: Deep Q-network-based multi-criteria decision-making framework for virtual simulation environment. Neural Comput. Appl. **33**, 10657–10671 (2020)
10. Laird, J.E.: An analysis and comparison of act-r and soar. arXiv preprint arXiv:2201.09305 (2022)
11. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines-a survey. Proc. IEEE **84**(8), 1090–1123 (1996)
12. Maslow, A.H.: A theory of human motivation. Psychol. Rev. **50**(4), 370 (1943)
13. McCall, R.J., Franklin, S., Faghihi, U., Snaider, J., Kugele, S.: Artificial motivation for cognitive software agents. J. Artif. Gener. Intell. **11**(1), 38–69 (2020)
14. Miller, D.: Hierarchical task network prototyping in Unity3D, p. 124 (2016)
15. Ordóñez Medina, S.A.: Personalized multi-activity scheduling of flexible activities. Arbeitsberichte Verkehrs-und Raumplanung 1099 (2015)
16. Peterson, J.L.: Petri nets. ACM Comput. Surv. (CSUR) **9**(3), 223–252 (1977)
17. Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., Torralba, A.: Virtualhome: simulating household activities via programs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8494–8502 (2018)
18. Reynaud, Q., Haradji, Y., Sempé, F., Sabouret, N.: Using time use surveys in multi agent based simulations of human activity. In: ICAART, no. 1, pp. 67–77 (2017)
19. de Sevin, E., Thalmann, D.: A motivational model of action selection for virtual humans. In: International 2005 Computer Graphics, pp. 213–220. IEEE, Stony Brook (2005)
20. de Silva, L.: BDI agent reasoning with guidance from HTN recipes. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 759–767. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2017)
21. Silva, L.d., Meneguzzi, F., Logan, B.: BDI agent architectures: a survey. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Japan, pp. 4914–4921 (2020)