Simulating Autonomous Agent in Connected Virtual Environments

Lysa Gramoli

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

INSA RENNES

THÈSE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Informatique*

Par

Lysa GRAMOLI

Simulating Autonomous Agents in Connected Virtual Environments

Thèse présentée et soutenue à l'IRISA Rennes, le 28 Septembre 2023
Unité de recherche : IRISA, Institut de Recherche en Informatique et Systèmes Aléatoires
Thèse N° : 23ISAR 26 / D23 - 26

Rapporteurs avant soutenance :

Julie DUGDALE — Professeure à l'Université Grenoble-Alpes
Nicolas SABOURET — Professeur à l'Université Paris-Saclay

Composition du Jury :

Président :	Jean-Pierre JESSEL	Professeur à l'Université de Toulouse 3 (UT3) - Paul Sabatier
Examinateurs :	Ronan QUERREC	Professeur à l'ENIB
	Jean-Pierre JESSEL	Professeur à l'Université de Toulouse 3 (UT3) - Paul Sabatier
Dir. de thèse :	Valérie GOURANTON	HdR, Maître de conférences à l'INSA Rennes
Co-dir. de thèse :	Jérémy LACOCHE	Chercheur, Orange Rennes
Co-encadrant :	Anthony FOULONNEAU	Chercheur, Orange Rennes

Invité(s) :

Bruno ARNALDI — Professeur émérite à l'INSA Rennes

Acknowledgement

Je tiens tout d'abord à remercier mes parents et mes soeurs Cloé et Lola, pour leur amour, leur soutien inconditionnel ainsi que les très bons moments passés ensemble durant ces trois années de thèse. Merci infiniment pour tous vos encouragements, vos conseils et votre bonne humeur qui m'ont été très précieux durant toutes les étapes de la thèse, et plus largement toutes les étapes de la vie.

Je tiens aussi à remercier particulièrement Rodri, mon amour. Merci infiniment pour tout le bonheur que tu me donnes ainsi que ton soutien sans faille, ta gentillesse et tes conseils qui m'ont beaucoup aidée tout au long de la thèse.

J'adresse également toute ma reconnaissance à mes encadrants de thèse, Valérie, Jérémy, Anthony et Bruno pour m'avoir guidée tout au long de ces trois ans. Vos conseils et votre perspicacité m'ont permis d'arriver avec succès jusqu'au bout de cette thèse. Je remercie aussi Pr. Sabouret et Pr. Dugdale pour avoir accepté d'être mes rapporteurs ainsi que Pr. Jessel et Pr. Querrec pour avoir été mes examinateurs.

Je souhaite aussi remercier toute ma famille, en particulier mes grands-parents, ma tante, mes oncles et mes cousins pour leur soutien très précieux. J'ai une pensée particulière pour mon Grand-Père qui m'a toujours soutenue jusqu'au bout.

Je veux également remercier mon amie d'enfance Carla, mes amis de Télécom Saint-Etienne, mes amis d'Orange, ainsi que mes amis d'Hybrid pour tous les très bons moments passés ensemble et pour vos encouragements si précieux.

Je tiens enfin à remercier tous mes collègues d'Orange et d'Hybrid pour m'avoir si bien accueillie, que ce soit au sein de l'entreprise ou du laboratoire, et pour tous les bons moments passés ensemble.

Table of Contents

List of plots

Tables

INTRODUCTION

Understanding human behaviors is essential to develop autonomous and adaptive systems aiming to assist people in their daily life [90]. An example of such system is a smart home where an adequate understanding of occupants can provide appropriate services such as medical care [96] or energy consumption optimization [99], [156]. To acquire this knowledge about daily human behaviors, the use of real databases coming from experimentation or surveys can be a solution. This knowledge can then be exploited by machine learning algorithms trained on these real data. However, due to the high sensitivity of these algorithms related to environment settings (architecture, the number of furniture, sensors, etc.) and occupant behaviors (preferences, hobbies, etc.), collecting such representative data in diverse environments (houses, buildings, cities, etc.) is a major difficulty: there are too many different environment settings and occupant behaviors to collect a global database [92]. In addition, the creation of such datasets in real environments is extremely costly in terms of time, money, and human resources. It can also induce privacy issues since data are collected from real people.

A promising way to deal with this problem consists in simulating these environments and their occupants to generate diverse and less expensive data [88], [90], [92]. These synthetic data, containing diverse information to understand human behaviors (daily routines, labeled daily performed activities, sensors triggered after an action, and so on) could then be used to complete real databases, create new ones, or replicate existing ones. In addition, since we know which activity is performed in real-time, we can automatically label the generated data. This automation is really useful for developers since the labeling step is really time-consuming, as explained by Chen et al. [45]. These labeled data can then be used to train human context-understanding models instead of real ones. However, if we want to substitute real data with simulated ones, we must ensure that such synthetic data are sufficiently credible.

The credibility of synthetic data directly depends on the credibility of the simulated environment and the behavior of virtual humans performing daily activities. One of the main issues induced by this dependency is **the simulation of virtual humans able to produce credible behaviors while being compatible with data generation.** This

Figure 1 – Global view of our 3D simulator and the agent performing activities.

compatibility induces the capacity to interact with Virtual Environments (VE) that can generate data by monitoring the performed actions. In addition, to check the credibility of these behaviors, setting up robust methods of validation is also required. In this thesis entitled *Simulation of autonomous agents in connected virtual environments*, we present our research works to answer this problem. More precisely, we explain how the design of an agent model, able to provide both autonomous and controllable behaviors while allowing the execution of activities in a 3D-connected environment, is a promising approach for solving this problem. We also study the methods to validate the credibility of the synthetic data and those of the agent behaviors as well as the limitations of this kind of solution. Our model is addressed to people (researchers, developers, etc.) wanting to exploit virtual humans to generate synthetic data, populate environments, or improve the performance of their existing simulators. These people will be called *Users* throughout this thesis.

To test the effectiveness of our agent model and its compatibility with data generation, we use 3D simulations representing connected environments equipped with sensors generating data by monitoring the agent actions. These sensors are non-intrusive since they are not microphones or cameras and rather retrieve ambient information such as water flow, movements in a room, door opening, etc. Figure 1 shows an example of our 3D simulation where an autonomous agent, simulating an occupant, performs daily activities such as *Working* or *Eating*. In this simulator, the agent is autonomous in its

decision-making and activity execution, but can also be configured in input to control the generated behavior when desired. An interface also indicates the state of agent needs (hunger, thirst, etc.). The simulated date time which can be modified before the launch is also displayed.

Context

Industrial Context

This thesis was funded by *Orange*[1], a french major telecommunication company that also provides digital services. More specifically, Orange proposes new services in connected environments such as smart homes, buildings, and cities. These environments enriched with sensors and effectors are used to offer services adapted to the occupant's needs such as call management, intelligent assistant, smart TV, security services, network management, and so on. To study and simulate these services, several research platforms were developed. Some of them are transversal to our thesis since they have similar requirements. This is the case for the *Home'In Platform*[2] dedicated to connected indoor environments, where simulating credible virtual humans is essential to properly study home automation services. The same need also exists for the *Plug'In Platform*[3] dedicated to connectivity, since human's actions can impact the network performance. Finally, using virtual humans is also useful for the *Thing'In Platform*[4] developed to study digital twins (i.e. the virtual replication of a real object whose the state is synchronized in real-time). Virtual humans are used in this platform to simulate the impacts of human actions on such objects. Our thesis is a part of the internal project including the latter platform, aiming to study the potential of digital twins and simulators for future Orange services.

To improve these services, retrieving human behavior knowledge is needed. Researchers had therefore developed machine learning algorithms specialized in understanding human context. However, since these algorithms require a large amount of labeled data, and real data are scarce, expensive, and could induce privacy issues, the company wanted to explore synthetic data generation. For this purpose, 3D environments able to

1. *Orange Website*: `https://www.orange.fr/portail`
2. *Home'In Platform*: `https://hellofuture.orange.com/en/sensitive-home-built-rebuilt-homein-platform/`
3. *Plug'In Platform*: `https://hellofuture.orange.com/en/lets-design-5g-network-together-plugin-platform/`
4. *Thing'In Platform*: `https://hellofuture.orange.com/en/thingin-the-things-graph-platform/`

capture data and virtual humans were designed by the Orange researchers to generate new data. 3D environments are effectively well suited for this use case since they provide a finer granularity of performed activities, and enable the simulation of sensors that are environment-dependent, such as presence sensors. Figure 2 shows an example of the 3D environments created by Orange researchers to generate synthetic data. However, the provided virtual humans had predefined behaviors inducing very limited behaviors compared to what a human could do, thus degrading the diversity of synthetic data. Therefore, this thesis was conceived with the aim of improving these behaviors by making the agents autonomous in their decisions and their activity execution, but also configurable for users (to allow several agent profiles, to impose activities at a specific time, etc.).

Figure 2 – Examples of 3D Indoor Virtual Environments made by Orange researchers to generate synthetic data. The two images at the top left are the same 2-floor virtual smart home replicated from a real one. The two images situated at the top right and the bottom left are fictional virtual smart homes. At the bottom right, an image of a smart building replicated from a real one.

Other Application Contexts

Beyond our industrial context, designing credible virtual humans able to interact with VE is also required for other fields of research and use cases.

As mentioned previously, virtual agents can be used in research fields aiming to generate data about human behaviors. They can also be used to improve existing simulators to create new data from existing simulations. This is notably the case where researchers and developers would seek to generate data about the *understanding of inhabitants at home* through the use of simulators based on computer vision (VirtualHome) [144] or on data coming from connected objects (OPENSHS) [4]. Researchers working in the building and energy sectors can be also concerned to know the energy consumption of a household or the impact of building insulation and weather conditions on behaviors [99], [156].

Exploiting credible virtual humans is also essential to populate virtual environments in a credible way. This is the case for the fields of video games, entertainment, cultural heritage, and training courses. Regarding video games [60], Non-Player Characters (or NPC) are used to have a better immersion of the player by showing NPC having habits of life impacted by the time of the day or having different behaviors according to the player's choices. In the field of cultural heritage, virtual agents can be used in Archaeology to reproduce an ancient civilization that had other ways of life [173] or to simulate tour guides in museums [76]. Virtual agents can also be really useful for training courses where they can assist or challenge the user faced with specific situations [113], [168]. In the *Inria/IRISA Laboratory*[5] where the thesis takes place, the use of credible virtual humans is useful to populate Virtual Reality (VR) environments sometimes used for training courses [48] but also to simulate a crowd having individual behaviors [94].

Finally, virtual agents can be useful for fields related to climate change and sociology where researchers can measure the impact of individual behaviors at the societal level faced with specific situations (climate changes, disaster impacts, cultural influences, and so on.)[29], [113]

Challenges

To simulate credible human behaviors that can be used for data generation and thus answer the issue defined previously, several challenges must be addressed such as **automatically producing credible human behaviors, giving a compromise between control and autonomy on behaviors, and validating the credibility of these behaviors**.

5. *Inria*: `https://www.inria.fr/en`; *IRISA*: `https://www.irisa.fr/`

Challenge 1: Automatically producing credible human behaviors

In order to correctly simulate a human, it is important to generate behaviors that are close to what a human could do. The simulated behaviors must therefore be credible.

In our case *Credibility* must be distinguished from *Realism*, which seeks to reproduce as exactly as possible the physical world. However, simulating an ultra-realistic behavior is very complex and in our use case, reaching such a degree is unnecessary since the granularity of data produced by the virtual sensors located in the Virtual Environment is not sufficient. **We rather seek to produce a sufficiently credible behavior so that the sensors of the connected environment react correctly and the data produced is close to those collected with a real human.**

Credibility in human behaviors

Believability in the behavior of an intelligent virtual agent consists in demonstrating ***coherence*** *in the agent's reactions and its motivational states and* ***consistency*** *among similar kinds of situations.*
Definition taken from the work of Avradinis et al. [11]

To avoid scripted behaviors (i.e. behaviors written manually), which would make the realization too expensive and would produce too limited and robotized behaviors, it is essential to generate diversified human behaviors in an autonomous way. This requirement is also confirmed in some existing works, such as the works of Avradinis et al. [11] and Handel et al.[86]. According to their analysis, if we want to obtain credible behaviors (whose definition, taken from the work of Avradinis et al. [11], is stated above), **autonomy is essential since the agent could thus satisfy its own desires and objectives as a human would.**

Judging from the definition of Credibility, the design of credible behaviors concretely implies:

1. **Coherence** in the reactions of the agent and its motivational states. This means being coherent in interaction with the environment and in reaction to events. It also needed to be coherent with the agent internal state (such as physiological needs, preferences, etc.), but also coherent with the initial constraints that could be imposed on the agent (remaining resources, activities possible at certain times, etc.).

2. **Consistency** of behaviors in similar kinds of situations.

Another parameter that is essential to improve the credibility of behaviors is **the diversity of the generated behaviors**. Effectively, since humans do not produce identical routines or behaviors to perform the same activity, this criterion is considered in addition to the coherence and consistency of the behaviors.

To summarize, to simulate human behaviors that can be used for data generation, we must respect this first challenge involving generating credible human behaviors in an autonomous way. To reach this credibility, the simulated behaviors must be coherent, diversified, and consistent when faced with the same type of situation.

Challenge 2: Offering a Compromise between Control and Autonomy over Behaviors

As detailed before, our goal is to propose a credible virtual human compatible with data generation. To do this, we need to produce an agent model providing a balance between control and autonomy over behaviors to allow the user to configure their simulated agents as desired. The respect of imposed constraints will thus compete with the management of the needs (thirst, hunger, etc.) and unforeseen events. Consequently, our agent model must be able to manage and anticipate them to avoid conflict.

The strong constraints given by users will be mainly related to the protocols used to supervise synthetic data creation. These protocols are often designed to impose specific situations which can then be analyzed or exploited by algorithms, such as those used for future activity prediction [52]. The purpose of using synthetic data is to counterbalance the lack of real ones due to their cost in terms of time, human resources, and money. Simulated data can thus be used for several purposes: to complete a real database, create a new one, or replicate a real one. In these three cases, experimental protocols can be imposed by users to lead the creation of their data, such as a calendar of activities to perform at a specific time or with a specific list of actions, the number of days, the protocol to label data, and so on. When users want to create a new database, they often would like to simulate a specific situation they cannot test in reality. Imposing specific constraints on the simulation and on the agent thus becomes necessary. In the same way, to complete a real database or to replicate one, the same situation and the same experimental protocol used during the creation of the real data must be respected. Otherwise, too many biases will be introduced and strong heterogeneous data will be

obtained. Examples of such experimental protocols can be found in the survey of Htun et al. [92] where authors explain that some protocols can impose specific activities to perform (called *Low spontaneous acting*) or can let the participant choose (called *high spontaneous acting*). Another example of protocols can be found in the real Orange4Home real database [51] where we can find experimental protocols containing a calendar of activities that must be performed at specific times. Respecting these protocols also enables a comparison of effectiveness between simulated data and real ones for validation purposes.

Therefore, our agent must be able to respect this kind of protocol to create correct synthetic data. It is also necessary to let the agent be autonomous when it is not submitted to these constraints. To do this, anticipation processes must be set up so that the agent can be ready at the required moment. To achieve this compromise, the human behavior model must be designed to allow a good balance between control and autonomy. These controllability aspects led us to favor agent-based approaches rather than data-based approaches such as Generative Adversarial Networks (GAN) [78] since these aspects impact the generated data and GAN-style approaches are not optimal to manage them, contrary to agents.

To summarize, in order to have virtual humans that can be used for data generation, we must respect this second challenge of having an agent able to manage strong user constraints while remaining able of producing credible behaviors.

Challenge 3: Validating the credibility of the generated Behaviors and Data

To ensure that the synthetic data generated by the behaviors are sufficiently credible to be used as real data, validation methods must be set up. These methods must both verify the credibility of the behaviors by evaluating the criteria stated in Challenge 1, and also verify the credibility of the simulated data produced by the virtual environment (VE) to know if the agent is sufficiently accurate in its interactions so that the synthetic data can provide results close to real ones.

Diverse validation methods exist to either validate behaviors or synthetic data. Unfortunately, few existing approaches propose multiple validation methods to validate their model: they are often limited to a single type of assessment and a single context. **Proposing several validation methods** is yet essential to ensure that the model generally produces credible behavior. Effectively, some situations or some virtual environments could

negatively or positively impact the agent behaviors. For example, if a situation involves an activity interruption, we need to make sure that the model supports this situation. The same requirement, imposing the implementation of multiple validations, has to be also respected for synthetic data to avoid the case where data is credible in one situation but not in another. In parallel, approaches validating both their model functioning and its use in practical applications are still scarce.

Contributions and thesis organization

Unfortunately, few existing approaches are designed to deal with all these challenges at once. Some of them propose totally controllable agents that lose credibility whereas others propose totally autonomous agents that lack control. Moreover, multiple validation methods are rarely proposed to assess the credibility of the generated behaviors and data in an accurate way. Meeting these challenges is even rarer in multi-agent approaches where the control over the agents is often limited and validations are difficult to apply.

Through the different chapters of this thesis, we propose both an agent model able to give a compromise between control and autonomy (respecting Challenges 1 and 2), and validation methods (respecting Challenge 3) to answer our main issue regarding the simulation of human behaviors that are credible and usable in data generation. To illustrate our use case, we place ourselves in the context of a 3D smart home that was replicated from a real one where a real database, called Orange4Home, was created [51]. This database contains the data of sensors having recorded 4 weeks of daily activities performed by a real human. The thesis is structured as follows:

Chapter 1 deals with existing works about the simulation of virtual humans for data generation with their validation methods, the ways to interact with a 3D Virtual Environment, and the existing methods to simulate an autonomous agent with their validation methods.

In Chapter 2, we propose our agent model inspired by BDI architecture [162] and enriched with a reactive scheduler [164] to provide a compromise between control and autonomy over behaviors while keeping credible behaviors (Challenges 1 and 2). Our model is composed of three components: the Internal State Model managing human needs to provide autonomy, the Decision-Making Model containing the reactive scheduler to adapt the level of autonomy according to constraints, and the Execution Model to execute the chosen activities in a 3D smart home. An Input-Output method is also

proposed to make a first validation regarding the credibility of behaviors (Challenge 3).

Chapter 3 develops the aspects of the agent aiming to improve credibility and adaptation to dynamic environments (Challenges 1 and 2), necessary to introduce multi-agent systems. We thus explain how world resources and unexpected events coming from the environment are managed. A user-experiment protocol is also proposed to validate the credibility of synthetic data in the context of Dynamic Environments.

Chapter 4 explains our approach to validating the credibility of simulated data coming from the sensors triggered by the agent's actions (Challenge 3). For this, we use the real Orange4Home database [51] as ground truth and the virtual replication of the real smart home to generate similar synthetic data. This approach aims to show that simulated data can be used as a substitute for real ones. To show this, we train with these data on two machine learning algorithms specialized in human-context understanding: current activity detection and future activity prediction. We then compare their performance rate to know if synthetic data can produce similar results compared to real ones.

In Chapter 5, we conclude the thesis by summarizing our contributions and by explaining the advantages and limitations related to our solution. We also detail our future works from short-term and long-term perspectives.

Chapter 1

Related Work on Human Simulation and Data Generation

In this chapter, we explore existing works related to the simulation of humans in virtual environments as well as simulations that are used to generate data about daily activities. In the introduction, three challenges were put forward to provide a simulated human compatible with data generation: automatically producing credible human behaviors (Challenge 1), providing a compromise between control and autonomy over behaviors (Challenge 2), and validating the credibility of the generated behaviors (Challenge 3). We assess state-of-the-art works related to the simulation of human behaviors to know whether they can answer these challenges at the same time and if they could be usable in data generation. We also introduce definitions related to autonomous agents.

More concretely, Section 1.1 is dedicated to existing simulations especially configured to generate synthetic data about daily-activities. We also explore the ways to validate the credibility of synthetic data produced by such simulations. After this, Section 1.2 details the existing agent models and provides definitions related to agents, environments, and activities. We also explore some existing methods to validate the credibility of behaviors without using synthetic data.

1.1 Simulations to generate Data related to Human Behaviors

In this section, we explore several approaches creating simulations and virtual humans to generate synthetic data. Some of them simulate humans in very large outdoor environments such as a city [24], [86], [93], [94], whereas others simulate humans in smaller indoor spaces such as buildings or houses [4], [88], [104], [117], [159], [181]. Regarding Humans, they can be simulated in various forms depending on the environment or the

type of data that must be produced. For example, human behaviors can be generated through autonomous agents [117], real users controlling avatars [4], or by extrapolating data coming from real databases [2]. To know if we could use existing simulations for our use case (e.g. simulate a human in a smart home to generate data about daily activities and human routines), we analyze if the simulated environments, virtual humans, and interaction between them are sufficiently developed to produce synthetic data accurate enough to be used as real ones. Effectively, as said in Introduction, since real data are costly and rare, we would like to complete them with synthetic ones. In this section, we study some existing works using simulations to generate synthetic data about daily activities and human routines. We detail in section 1.1.1 what is the most accurate environment to produce data when sensors and effectors are simulated to generate them. Finally, we also study in section 1.1.2 what is the most accurate way to simulate humans for addressing our three challenges.

1.1.1 Types of Environment used to generate Data

The accuracy of the simulated environment is essential to create credible synthetic data. Effectively, when we want to replicate real datasets, complete real ones, or even create new ones, the simulated data must be of the same data type with the same level of accuracy as real ones. Otherwise, the set of real and simulated data will be heterogeneous and difficult to use. Judging from our use case explained in Introduction, synthetic data are created from sensors and effectors simulated in a smart home to capture information from movements, sounds, and so on. We thus explore in this section what kinds of environments are used for data generation and which one could be the most accurate when sensors and effectors are used to capture data. We distinguish 3 types of environments: Abstract Environments, 2D environments, and 3D environments. An example of each environment can be shown in Figure 1.1.

Abstract Environments: Some approaches use abstract environments to generate data about human behaviors. In this type of environment, the world and the objects are represented in a schematic and a semantic form (lists of objects, hierarchies, Interface, etc). Unlike 2D or 3D environments, human movements in this environment are not simulated, and the visual representation of the world and humans is often absent. Data that can be retrieved from abstract environments can be activity calendars or statistical results. For instance, In the work of Meister et al. [127], they propose an agent model

simulating journeys within an abstract environment representing a city to produce activity calendars depending on individual and social features. In the TASHA model proposed by Roorda et al. [129], [152] and ADAPTS proposed by Auld et al. [10], travel data are generated by extracting probabilities on real ones. No environment is thus required to create synthetic data. Similarly, Airale et al. [2] use a generative-based model called SocialInteractionGAN to create synthetic data about the sequence of actions by using and analyzing real data and thus avoiding the simulation of environments. Concerning Elbayoudi et al. [68], an older adult's behavior is simulated in an abstract house by the use of probabilistic approaches that generate statistics about the distribution of activities throughout the day. Regarding the SMACH Model developed by Amouroux et al. [6], a schematic interface of the environment is given to the participants during an experiment to control and modify the scenario of the autonomous agent simulating their daily routines. In output, the generated data are statistics about household routines, activities, and energy consumption. For their part, Kashif et al. [99], use an abstract Java interface called Brahms to simulate the behaviors of agents, their communications, the objects of the world, the geographical position, and the activity model. As the output, they can produce an activity calendar and statistical results about the energy consumption of devices. With these kinds of data created from abstract environments, we have a global idea of what objects are used and what is the routine of the agents. However, no details are given about how the activity unfolded and how the agent moves and interacts with the objects. These details may be missing if we want to understand how humans behave in a specific activity for instance. In addition, simulating correctly a smart home with this kind of environment is difficult since some sensor behaviors, such as presence sensors, have their performance directly impacted by their location in a room. It will thus be hard to insert their data in real databases coming from smart home environments such as the Orange4Home dataset proposed by Cumin et al.[51], or the ARAS dataset used by Alemdar et al. [3]. Abstract environments are also unsuitable to create datasets capturing synthetic data through cameras as proposed in the Virtual Home approach created by Puig et al. [144]. Yet, these camera data are useful to train activity recognition algorithms as explained in the survey of Chaquet et al. [42]. On our side, since we want to generate data from sensors and effectors that are sensitive to their environment, these abstract environments are then not suitable.

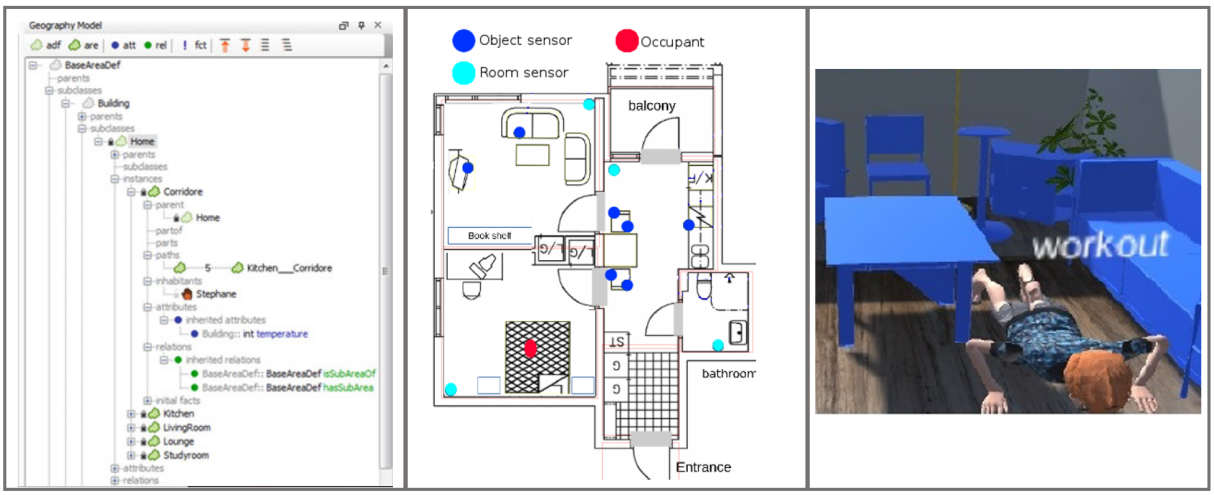

Figure 1.1 – Examples of existing environments. On the left, an abstract environment proposed by Kashif et al. [99]; on the middle, a 2D environment proposed by Renoux et al. [148]; on the right, a 3D environment proposed by Zhao et al. [181]

2D Environments: Some other approaches use 2D environments to capture their data. In this kind of environment, the virtual human can move in a 2D world to interact with different objects. To do this, most approaches use 2D architecture plans, enriched with points that locate the interesting objects and the agent. These environments can thus generate more diversified data than in abstract environments (such as data on movements and certain sensors). 2D environment are used for instance in the work of Renoux et al. [148] through a 2D smart house equipped with sensors capturing the activities of a virtual human. A similar approach is found in the work of Klein et al. [104] where several occupants can interact with electrical appliances (light, desktop, thermostat) sending data about their energy consumption. We also have the approach of Lundström et al. [121] where avatars are used in a 2D smart home to store daily activities data. In the MASSHA model created by Kamara et al. [97], agents are used to simulate daily activities in 2D smart homes capturing data through its sensors. Finally, regarding the work of Bourgais et al. [29] where a multi-agent system called BEN is implemented, they propose a 2D nightclub created with the *GAMA platform*[1] where agents are simulated by triangles. This simulation gives in output statistics about the evacuation scenario reproducing a real one that turned into a tragedy (personality impacts, number of deaths, etc.). These 2D environments provide additional information about the movements of agents and the

1. *GAMA platform*: `https://gama-platform.org/`

location of specific devices needed for activities. However, this is still not enough for some databases created from smart homes that use, for example, presence sensors whose height can impact performances. In addition, this kind of environment is not adapted if we would like to produce synthetic data through a camera for activity recognition purposes [42], [144]. The interaction between the agent and environment is only represented by navigation. Consequently, a fine granularity of activities is therefore not recoverable, and some sensors requiring a finer level of granularity cannot be be simulated (e.g. closet opening sensors). All this imprecision limits the credibility and diversity of ways to perform an activity. This is also not sufficient for our use case since we have sensors (like presence sensors) that need to consider the height.

3D Environments: Some existing approaches use 3D environments to generate data. In these environments, virtual humans can move in 3D world to interact with 3D objects. These environments can be created with 3D modeling tools (e.g. *Blender*[2]), scans of real environments, or files coming from architecture such as BIM (Building Information Modeling). This is the case for the works of Roitberg et al. [151] where 3D houses were created with *The Sims 4*[3] game to replicate real houses. In the work of Alshammari et al. [4] they developed OPENSHS, an open-source 3D smart home simulator platform built to generate daily activities datasets in smart homes. In OPENSHS, the simulated environment and sensors are easily configurable to adapt to the user's needs. Concerning the work of Ho et al. [90], a smart home was created with *Unity Engine*[4]. Regarding Srivastava et al. [166] they propose the BEHAVIOR Benchmark to create new activities with Virtual Reality (VR) tools. The authors use the 3D virtual houses provided by iGibson [118] to allow VR and capture data with virtual cameras. In the BIM SIM 3D approach proposed by Zhao et al. [181], agents are simulated in a 3D smart home designed in Unity Engine with the help of BIM used in architecture. A 3D smart home with an agent following a predefined scenario to create data for activity recognition can also be found in the work of Helal et al. [88]. This approach, called Persim 3D also compares its simulated data with a real dataset called Persim 1.5. In the same trend, we have also SIMACT proposed by Bouchard et al. [28] where a 3D smart home is implemented in Java to be used for activity recognition. Experimenters can also add new scenarios coming from real humans. Finally, in Kashif et al. [100], a real user interacts with a 3D house

2. *Blender*: `https://www.blender.org/`
3. *The Sims 4*: `https://www.ea.com/fr/games/the-sims/the-sims-4`
4. *Unity Engine*: `https://unity.com`

through an avatar. Data are retrieved by the use of 3D sensors capturing the performed activities. Data produced by these 3D simulations are sufficiently complete for the most of real databases and for our use case. However, there is sometimes little information about how sensors are simulated, and whether they are inspired by real ones.

The analysis of existing environments used for data generation allows us to explain why we privileged 3D environments rather than other environments. Effectively, in contrast to 2D and Abstract environments, 3D environments provides a finest granularity in activities and can thus offer more detailed data. However, they are also more cumbersome to simulate. Since we want to make our agent compatible with data generation with any kinds of environment, it must be compatible with 3D ones which are the most complex. 3D environments also essential in the case where data must be generated with sensors and effectors that are sensitive to their environment (as in our use case). The choice to use 3D environments also impacts our agent model design since stronger precision about human interaction is required. In the following section, we study the different ways to simulate humans to know what is the most accurate to reach our challenges explained in Introduction.

1.1.2 Ways to simulate Humans Behaviors for Data Generation

To simulate virtual humans that are compatible with data generation, several approaches exist such as agents, avatars, or data-based models. Figure 1.2 shows examples of diverse approaches to simulate virtual humans.

Avatars-based approaches: Avatars are virtual humans controlled by real users. Some approaches use them to generate daily activity data such as in OpenSHS [4] where users perform activities through the use of Virtual Reality. These activities are then blended with other sets of activities to create routine datasets. In the BEHAVIOR model [166], VR is also used to register a sequence of actions performed by a real user to generate new datasets. In the work of Roitberg et al. [151], data are retrieved with cameras to capture activities. Players were asked to move their avatars around *The Sims 4*[3] game environment to record videos of the performed activities. In parallel, virtual videos were compared with those coming from the Toyota real Dataset [54]. To do this, they virtually reproduced the real houses and occupants registered in the real dataset. Some examples of avatars can be found in the work of Kashif et al. [100] where users can interact with the

devices of a 3D environment to keep their avatar in its thermal comfort zone. Finally, in the work of Lundström et al. [121], 2D avatars managed by players are used to generate synthetic data about daily activities. Generally, Avatars-based approaches are interesting to retrieve credible data since real persons are used to generate synthetic data. However, they remain limited to use since datasets cannot generate long-time periods. They are also costly to produce in human resources and time. In addition, people might not interact in a 3D environment as they would in a real setting, limiting sometimes the realism of the performed activities even though they come from real humans. Moreover, requiring recording sessions is costly and limits the possibility to obtain activities performed in diverse ways.

Data-based approaches: Some other approaches use data-based approaches to simulate humans through generative models. To do this, they directly generate new data by analyzing their input data. Many diverse input data can be provided, regardless of the environment or context where these data come from (activity calendar, sensors data, paths, etc.) and these models will generate the same type of data that the input ones. The main difference between data-based models and agent-based models using learning approaches is the necessity to interact with the environment when data are generated. In data-based models, data are directly generated without any simulation of interactions between the virtual human and its environment. On the contrary, agent-based models can generate data only after an action or a decision to interact with the VE. Some data-based approaches can use for instance GAN-based models [89], [154], such as the work of Airale et al. [2] proposing the SocialInteractionGAN model to generate multi-person interaction sequences. The authors propose to use a Generative Adversarial Net (GAN) neural network to generate simulated human activity data. These data can be used to complete real databases for instance, since little heterogeneity will appear between real and simulated data. The motivation for this approach is that an adversarial strategy can act as a way to enforce the credibility of generated data. In adversarial strategy, an algorithm called *Discriminator* will discriminate the results produced by the other algorithm called *Generator*. The discriminator is thus trained to distinguish between generated and real data, forcing the generator to improve the realism of its generated data. To evaluate the quality of generated interaction sequences, they propose different metrics based on score and distance to assess the quality of the synthetic image generated by GANs [89], [154]. These approaches can thus generate new data that are close to their input. They

only require a trained generative model and thus can be trained with many diverse data, regardless of the environment or context where these data come from. However, controllability over behaviors can be limited due to their nature: If new real data are added or if the input parameters must be changed, re-training is often required. Consequently, some real databases that require protocols could not be easily completed or replicated with Data-based approaches. Furthermore, since data used to train algorithms are not dependent on the actions triggered in the environment, it can generate data that appears to be consistent but actually impossible to produce in the environment. For instance, if we want to generate data about a human's navigation between rooms in a house, the algorithm could generate path navigation between two rooms that seems logical (such as between the kitchen and the living room) but actually impossible to achieve in the house where data must be generated (for example if the kitchen and dining room are not connected).

Agent-based Approaches: Humans can also be simulated through virtual Agents to produce data. These agents are individual entities that can perceive, make decisions, and act in their environment. In contrast to avatars, they are not controlled by a real user but by an algorithm, and unlike Data-based approaches, an environment is required to perform actions. Agents are well-suited to automatically produce data during long-time periods in an inexpensive way. With the diversity and flexibility of agent architectures, the authors can simulate diverse human cognitive processes such as decision-making, memory, motivations, perception, and so on. For all these reasons, agent-based models are one of the most used approaches to simulate humans. Among agent-based models used to produce data, we can find the work of Ho et al. [90], and those of Zhao and al. [181] proposing BIM Sim 3D, where an agent performs activities in a 3D smart home where sensors are placed to capture data. Regarding the work of Renoux et al. [148], 2D agents are simulated in a 2D building where sensors are also placed to capture data. In the SMACH model [6], the work of kashif et al. [99], [101] and those of Klein and al. [104], they use agents to interact with electronic devices and generate data about energy consumption. In SIMACT proposed by Bouchard et al. [28] as well as in Persim 3D proposed by Helal et al. [88], agents follow the predefined scenarios of activities coming from real datasets and user experiments. Finally, regarding the work proposed by Puig et al. [144] and those of Egami et al. [67], agents can perform activities to generate video data by the use of virtual cameras recording the agent actions. Generally, due to their flexibility

and diversity of design, agent-based models can keep a compromise between control and autonomy which is essential to address Challenge 2. In addition, they can quickly create synthetic data for long-time periods in comparison with avatar-based approaches. Regarding data-based approaches, agent models are more demanding for designing since the rule of the environment has to be respected by the agent, necessitating a simulation of this environment. In return, agents will not generate data that are incompatible with the simulated environments, which is required to address our Challenge 1. For all these reasons, agents-based approaches are in our case the best approach to respect our challenges while being compatible with data generation.

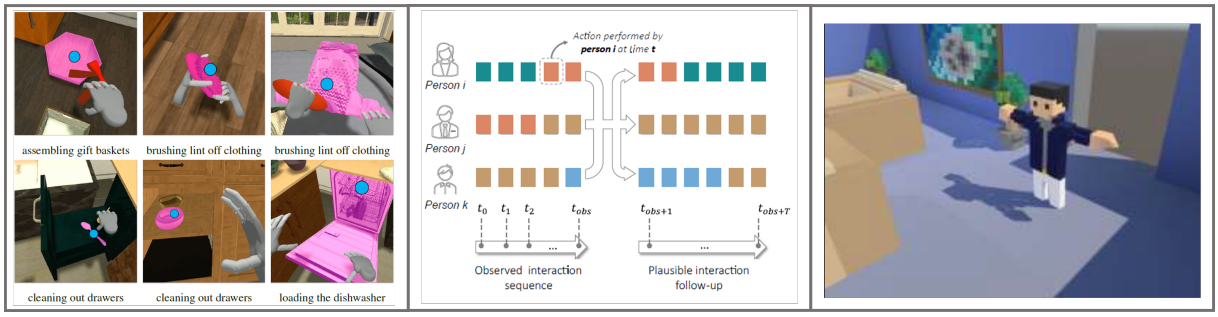

Figure 1.2 – Examples of virtual human simulations. On the left, avatars controlled by players in VR in the BEHAVIOR framework [166]; On the middle, a data-based approach to generate new data from real ones proposed by Airale et al.[2]; On the right, an agent simulated in a 3D smart home proposed by Ho et al. [90]

The analysis of these different ways to simulate virtual humans allows us to explain why we privileged agents rather than avatars or data-based approaches. Effectively, agents will not generate data that are incompatible with the simulated environments in contrast to data-based approaches. They are also less costly to produce and enable simulations over long-time periods in contrast to avatars.

1.1.3 Methods to validate the Credibility of the Generated Data

In this section, we present existing methods to validate the credibility of synthetic data. We find two main paradigms for asserting credibility. Some contributions generate credible data with *a priori* arguments: the models used to generate data were constructed based on real data, which should lead to credible data by design. Other contributions assess credibility with *a posteriori* arguments: generative models are still based on real data,

but credibility is evaluated through the use of specific metrics on these data compared to real data.

1.1.3.1 *A priori* Approaches

Methods using *a priori* arguments involve the use of real data to design the model and produce credible data. Among them, Jang et al. [93] proposes a deep Q-network trained on large datasets coming from social networks, which is an *a priori* way to ensure the credibility of generated goals. However, only global surveys completed by a large population, and storing the start times and duration of activities are used to construct and verify the credibility of generated data. This is not enough to know whether each agent can behave in credible ways in the choice of their activities, the diversity of their choices, and the execution of these activities. Regarding the case of SIMACT proposed by Bouchard et al. [28] and Persim 3D proposed by Helal et al. [88], *a priori* way is also used since agents follow predefined scenarios coming from real datasets: All their acting and decision are bounded by these real input dataset. The model can therefore provide realistic data as these are scenarios reproduced from real humans. However, the agent behavior is entirely scripted, and new simulated data cannot be generated without registering a human with these approaches, preventing experimenters from completing existing databases for example. On the side of OpenSHS [4], *a priori* arguments are also used to construct and validate the credibility of generated data. Since action sequences were set directly by real humans, we could expect that the generated data would be more credible. However, as said in section 1.1.2 people might not interact in a 3D environment as they would in a real setting, and the possibilities of control and variability are limited due to the cost in comparison with agent-based models.

1.1.3.2 *A posteriori* Approaches

Methods using *a posteriori* arguments involve using metrics to check whether the data produced is credible or not. Among them, we can find the work of Renoux et al. [148] where an agent can perform daily activities in a 2D simulation. The generated data are validated by real experimenters that have to guess if the presented activity calendars are made of real or simulated activities. However, this kind of assessment may involve bias or uncertainty due to the subjectivity of experimenters. In addition, people may have difficulty imagining what a real schedule would look like. This is why, even though collecting the subjective opinions of users is relevant, an additional objective method is

necessary to validate the credibility of data. Regarding TASHA, proposed by Roorda et al. [129], [152], a comparison between simulated travel scheduling and the real GTA (Great Toronto Area) dataset is used to validate the agent model. For their part, Ordoñez et al. [135] compared real daily-activity schedules, coming from socio-demographic data and geographical information, with their simulated ones generated by their genetic algorithm (GA). In the MASSHA model proposed by Kamara-Esteban et al. [97], an *a posteriori* approach is also proposed where durations and frequencies of simulated activities and the behavior of virtual sensors are compared to real ones. Finally, in the work of Handel et al. [86], [87], they statistically compare real data made of video and interviews with the result of their agent model.

In the recent works of SMACH proposed by Reynaud et al. [149], [156], the generated routines data produced in an abstract house are analyzed by *a priori* approaches (time-use surveys and statistics methods to design the model) and *a posteriori* approaches (microscale level assessment with 8 real occupants and macroscale level assessment with population surveys). The authors also suggest other ways to evaluate *a posteriori* the credibility of generated data by the use of clustering simulated samples: when clusters are found with both real and simulated samples, then real and simulated data are hard to separate and thus simulated data can be considered as credible. Even though these proposed validations can address our second challenge, SMACH is difficult to use in a database such as Orange4Home, where mandatory activities are required at specific times, since the choice of activities is based on probabilities.

In the recent works of Kashif et al. [99], *a posteriori* approaches are also used to validate the simulation and the agent behavior model. Concretely, they use the real Irise dataset [55] as well as experimental studies to make a comparison between the consumption distribution of devices (such as fridges) obtained by the simulation with the one stored in the Irise dataset. If the gap between the both is too large, the input parameters of the simulation (environmental parameters, time parameters, agent parameters, etc.) are tuned to fit better with the real results. This approach can effectively improve the credibility of the data generated by the model through the use of this tuning. However, the model could become too specialized for the Irise database and could be no longer valid in other environments or other databases without repeating this tuning process in the new environments.

In the SocialInteractionGAN approach [2], which is a data-based approach explained in section 1.1.2, statistical comparisons based on metrics (Inception score and Inception

Distance) provides interesting insight but do not guarantee that data is credible for the task it will be used for. For example, a simulated sample might be very similar to a real sample according to these metrics but lacks subtle information essential for distinguishing human activities since their produced data do not directly depends on any environment.

1.1.4 Conclusion

Through this section, we studied some existing approaches that simulate humans to generate synthetic data about daily activities. We first examine the simulated environment proposed by existing approaches and we showed that 3D environments are the most accurate to illustrate a smart home since some sensors can be sensitive to environments (such as cameras, presence sensors, and so on). We then study the different approaches to simulate a Virtual Human and we sorted them into three main categories: Avatars-based approaches, Data-based approaches, and Agent-based approaches. We explained that agents are more accurate to address our challenges while being compatible with data generation. Finally, we studied different approaches to validate the credibility of synthetic data. We observed that *a posteriori* approaches are the most suitable to make this but must not be limited to the statistical comparison that can induce imprecision in the results. In the next section, we precisely study the existing agent-based model to know which one could address our three challenges. In contrast to this section, the next one will be extended to approaches that are not always used for data generation.

1.2 Agent-based Models to simulate Humans

In the previous section 1.1, we saw that using agent-based models with 3D environments is the most accurate and flexible approach to have a human simulation compatible with any data generation, including those having strict protocol. However, many approaches exist among agent-based models, providing more or less sophisticated behaviors. As said in Introduction, we try to propose an agent compatible with data generation and able to address three major challenges: automatically producing credible human behaviors (Challenge 1), providing a compromise between Control and Autonomy (Challenge 2), and validating the credibility of the generated behaviors (Challenge 3). In this section, we study the existing agent-based approaches to analyze whether they can address our three challenges at the same time. Before this, we introduce some definitions related to

the agent concepts in section 1.2.1.

1.2.1 Definitions related to Autonomous Agents

In this section, we can find some definitions related to autonomous agents. In section 1.2.1.1, we give a definition of Virtual Environments that are essential to simulate agents. In section 1.2.1.2, we define what is an agent exactly. We then explain in section 1.2.1.3 the means of *Autonomy* in our context. Finally, we define in section 1.2.1.4 what is a need, activity, task, and resource in the agent context.

1.2.1.1 Definitions related to *Virtual Environment*

In this section, we give definitions related to *Virtual Environment*. More particularly, we describe the environment features that influence agents, in contrast with the previous section, where virtual environments were analyzed in a broader sense. *Virtual Environment* is a global notion to define the simulated world where a virtual human can interact. Judging from the definition of Dorri et al. [62]: The environment *refers to the place where the agent is located. [...] An agent uses the information sensed from the environment for decision-making.* According to the same authors, environments can have several features affecting the complexity of an agent-based system (All the below definitions come from their works):

Accessibility: This is related to the capacity of the agent to perceive environment data. When the environment is accessible, these data can be easily retrieved and updated by the agent. Otherwise, the agent can only perceive noisy or incomplete data (non-omniscience).

Determinism: This is related to the predictability of the results coming from the agent action. When an environment is deterministic, the result is predictable and the next state of each action is precisely known by the agent. An environment became undetermined when factors unknown by the agent can influence the next state.

Dynamism: This is related to the changes occurring in the environment that are independent of the agent actions. Environments only impacted by the agent are considered static. Otherwise, they are dynamics.

Continuity: This is related to the discreteness of the environment. A continuous environment uses a continuous function to affect the agent state (ex: agent moving in a 3D environment). On the contrary, a discrete environment forces the agent to be in a set of predetermined states. (ex: A timestamped position to move a 3D agent).

For our case where we seek to generate credible behaviors, our agent model must manage environments being at least Dynamic (since humans are influenced by their environment) and Continuous (since the real world is continuous). Regarding Accessibility, having a non-omniscient agent can be desirable for multi-agent simulations and dynamic environments and will be progressively set up in our thesis. Concerning determinism, having a deterministic environment is more suitable in our case since the generated behavior will be more controllable (Challenge 2) and the decision-making will have less complexity.

Some existing works also introduce *Intelligent Virtual Environments* (IVE) [13], [46], [138], [174], sometimes called *Informed Virtual Environments* [32]. These environments are enriched with useful information to describe objects and their possible interactions through the use of semantic databases [31], [32] or ontology-based methods [46], [71], [107]. Introducing IVE is essential to simulate interactions between agent and environment but also to identify which object is useful to perform an activity [174]. For example, if it wants to read a book, it needs to identify which objects in the world are books. Because of their properties, IVE is required to address our Challenge 1 since IVE allows us to produce more credible behavior through the simulation of interactions and world knowledge.

1.2.1.2 Definitions regarding the Notion of *Agent*

Agent is a wide term associated with several definitions. In one of the most global ones stated by the *Collins Dictionary*[5], an *agent is a person or thing that acts or has the power to act*. This is thus more related to the capacity of an entity to perform actions than its nature. In the field of Artificial Intelligence (AI), *Agent* also encompasses any virtual entity able to act and make decisions. According to Handel et al. [86] and Cardoso et al. [35], agent-based models became popular in the 1990s since they can efficiently simulate complex systems and help in problem-solving tasks. We thus find agents such as robots, routers, virtual machines, task schedulers, intelligent objects, virtual living beings, etc. The creation of models allowing the interaction between several agents, called Multi-Agent Systems (MAS), allows scientists to solve some NP-complete problems while limiting the

5. *Collins Dictionary website indicating the agent definition*: `https://www.collinsdictionary.com/dictionary/english/agent`

explosion of complexity. The definition that we officially use is the one proposed by Dorri et al. [62] in their survey:

Agent definition

Agent: An entity that is placed in an environment and senses different parameters that are used to make a decision based on the goal of the entity. The entity performs the necessary action on the environment based on this decision. *Definition taken from the work of Dorri et al.* [62]

In our use case, the entity is a virtual character used to simulate a human, and the environment is a 3D environment equipped with sensors and effectors used to collect data.

We can also add that *Agent*, *Virtual Agent* [24], *Intelligent Agent* [117], *Intelligent Virtual Agent* (IVA) [7] are often synonymous in the literature. In the field of video games, we also have *Virtual Characters* and *Non-Player Characters* (NPC) [167] that are commonly used in the gaming vocabulary to talk about virtual humans that are not controlled by players. They are thus synonyms of agents. Regarding the notion of *Virtual Human*, it includes all the way to simulate a human, as explained in section 1.1.2, making it less accurate than the agent notion.

1.2.1.3 Definition regarding the Notion of *Autonomy*

Among existing approaches, we can find more specific notions defining the *Agent* such as *Cognitive Agent* [125], *Autonomous Agent* [117], *Motivated Agent* [12] or even *Conversational Agent* [76]. These diverse nuances allow the authors to clarify which human features are mainly studied in their work. For instance, conversational agents are more related to human natural language whereas motivated agents are more related to what motivated the agent to make a decision.

In this thesis, we are particularly focused on the autonomy process regarding decision-making and the execution of chosen activities. We thus use the term of *autonomous agent* in our case. According to Cardoso et al. [35] and Avridinis et al. [11], *Autonomy* is an essential feature to have credible agents. Nevertheless, Avridinis et al. point out that *Autonomy* is often confused with *Automation*. According to them, an agent is **automated** if the user gives an explicit list of all goals to perform and all the steps to reach them.

Judging from them, the agent must be able to generate and manage its proper goals to obtain a truly autonomous agent. This assumes that it has intrinsic motivating mechanisms (needs, emotions, etc.) pushing it to act. A more precise definition of *Autonomy* can be found in the work of Brustoloni et al. [34], Castelfranchi et al. [37], [38], Dorri et al. [62], and Franklin et al. [73] where an agent is considered as autonomous when it has the *Power* to interpret the VE, create its motivations, and find out how to drive and achieve its own goals. In this thesis, we officially take the definition of Castelfranchi et al. [38] that summarizes this idea:

Autonomy definition

Autonomy: Autonomy actually is a matter of power (the capability of maintaining and of satisfying adaptive functions and goals through appropriate behaviors); this requires specific *powers* that can be distinguished in *external* and *internal*. *Definition coming from Castelfranchi et al. [38]*

Judging from the authors, *External Powers* are all the action conditions and resources allowing the agent to successfully act in the environment. In parallel, *Internal Powers* are the cognitive capabilities allowing the agent to collect information, create goals, and schedule plans to achieve its proper goals.

For our case, having an autonomous agent rather than an automated one is required since *Autonomy* allows a greater diversity in decisions and actions while keeping coherent and consistent behaviors (see Introduction).

1.2.1.4 Definitions related to *Need*, *Resource*,*Activity*, *Task*, and *Action*

In this section, we define *Need*, *Resource*, *Activity*, *Tasks* in relation to the context of daily life.

Needs: They are related to the human daily needs defined in Maslow's theory of needs [124]. In this theory, recognized by the psychological field, needs are ordered according to their urgency level by following the hierarchical structure called Maslow's pyramid of needs. Concretely, physiological needs (hunger, etc.), which are simple but imperative to satisfy, are thus distinguished from more elaborate needs, which are more complex but

less urgent to satisfy (self-esteem, etc.). Needs are often used as a source of motivation that will push the agent to act in the environment to satisfy them. They therefore enable autonomy since the agent is able to choose its proper goals according to its needs urgency. Among the physiological needs, we can also distinguish needs that are initiated by **internal factors** induced by the organism (food, water, energy, etc.) and those that are initiated by **external factors** induced by the environment (temperature, humidity, etc.).

Activity: An activity is a concrete formulation of the way to satisfy a goal or a need. For example, if the agent wants to satisfy its hungry, then *Eating* can be an activity. However, it is not atomic enough to be applied directly in a virtual environment. For example, if the selected need is *Need to be Entertained*, then possible activities may include *Watching TV*, or *Reading*. No indication is given of where and how they are performed. Our definition of *Activity* is first based on the work of Cumin et al. [51]: *An activity is a set of tasks (or operations), a task being a set of actions.* We also base our definition on the theory of the *Action Cycle* developed by Norman [133]. In this theory, Norman explains what drives humans to perform actions in specific contexts. According to him, a system of perception is used to compare the world state with the internal needs. If needs cannot be satisfied in this world state, a series of actions are set up to satisfy them. Such a series of actions serve as a basis for performing the activities in our work.

Task: To be executable in a virtual environment, activities are broken down into tasks. According to Cumin et al. [51], Tasks are concrete formulations of basic action sequences (such as animation) that can be directly executed by the agent. They can indicate the needed resources, the location, the name of used objects, etc. For instance, in the activity *Watching TV*, tasks can include *turning on the TV*, or *sitting on the sofa.*

Resource: They are related to the natural resources offered by the environment. Our definition comes from the work of Laborie et al. [110]: *We define a resource as any substance or set of objects whose cost or availability induces constraints on the actions that use them.* This definition was chosen since it is well suited to our context where our agent has to manage resources in its daily life such as food, phone, car, etc.

1.2.2 Relationships between Agents and Environments through Intelligent Virtual Environments (IVE)

As explained in section 1.2.1.1 the use of an Intelligent Virtual Environment (IVE) is essential to allow relationships between the agent and its environment. Effectively, if the agent must interact with a simulated environment to produce data, each object must be well recognized. This is why, it is necessary to enrich the environment with information describing and explaining how interactions can be executed. For example, the chair must be distinguishable from an apple so that the chair can be used for sitting and not used as food. This semantic enrichment can be made through two kinds of knowledge representation models: **Semantic databases** [84] and **Ontology-based models** [71]. In this section, we explain the advantages and drawbacks of each existing method and if they can be used in our context to allow the agent to perform daily activities in an accurate way.

Semantic databases: Also called knowledge bases, they are concepts explicitly describing different content features at different levels of abstraction [71]. These concepts can be set up with several kinds of structures such as metadata, database, languages, rules, etc. Among them, we have the object-relation-based model called #FIVE, proposed by Bouville et al. [31], which is implemented in the XR toolkit *Xareus*[6]. In #FIVE, the description is oriented towards possible interactions between the objects and other objects. #FIVE thus describes the world in how objects and humans can interact. For instance, we can group doors and drawers in the same category, since they share the same interaction: humans can open or close them. A hierarchy between them can also be set up to create more specific interactions. Another approach can be found in the work of Gutierrez et al. [84] where the rule-based standardized framework called MPEG-7 is introduced to reuse the virtual objects in other simulations while keeping intact their geometrical information. In the work of Gröger et al. [82], the CityGML standardized semantics, based on Geography Markup Language is also proposed to manage the semantic aspects of 3D city models such as their structures, taxonomies, etc. This allows users to employ virtual 3D city models for advanced analysis and visualization tasks. Other approaches also use rule-based approaches through constraint programming where the Prolog language [13] can be used to describe the features of objects. Sometimes, Unified Modeling Language

6. *Xareus Software*: `https://team.inria.fr/hybrid/xareus/`

(UML) is used as semantics to describe virtual objects [171]. There is also the use of BIM processes [181] to describe 3D objects. With semantics-based approaches, there is an explicit description of the relationships and features of virtual objects. They can be shareable with other simulations using the same kind of objects and can be ordered in a hierarchy. Their structure is generally less complex than ontologies, allowing an easier integration into a new environment, easier addition of new objects, and easier modification of existing objects. However, they cannot manage implicit knowledge, and they sometimes lack formalism [71].

Ontology-based models: Judging from the definition of Flotynski and al. [71], an ontology is defined as *specifications of conceptualization, in which objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them are included.* As with semantics approaches, ontologies use concepts to describe objects and relationships at different levels of abstraction. The main difference is that object description can be implicit. For instance, if we group metallic objects together to give them a special characteristic such as conductivity, then we can implicitly define other objects as non-metallic and therefore non-conductive objects. With this, new useful interactions can be managed, such as in our example the search for isolating objects without being forced to explicitly define the category since metallic ones exist. Among existing ontologies, we can find the surveys of Flotynski et al. [71] describing several categories of ontologies such as Web Semantics [59] and the Resource Description Framework (RDF) which are the first ontologies used for web contents. Other semantics web standards also exist such as the *Web Ontology Language (OWL)*[7]. Some approaches use ontologies to describe the contents of video games such as Kessing et al. [102] while some other approaches use ontology in a knowledge-based scenario framework to support agent planning processes [41]. Other approaches use ontologies for modeling VR scenarios such as in the work of Dragoni et al. [63] or for Augmented Reality (AR) content such as in the work of Walczac et al. [175]. In the work of Chevailler et al. [46], they present MASCARET, a UML-based ontology, developed for VR applications to describe 3D objects in terms of geometry, interaction with users, and physical constraints. Finally, some works also use ontologies to describe the decision-making content in a structured way [83]. The main assets of ontologies are the possibility to describe contents in an explicit and implicit way. However, their level of complexity due to many layers of abstraction

7. *https://www.w3.org/OWL/*

can be harder to understand and integrate into a specific simulation than other semantics-based approaches.

To conclude this section, we could observe that most of the semantic databases and ontology-based models are sufficiently suitable to be used for our case. Effectively, we need to have semantics that can categorize objects according to their possible interactions and that can be reusable in several environments. This is why, semantics models such as OWL, #FIVE, MASCARET, or BIM can all be used in our case. In the next session, we then focus more on the way to simulate agent behaviors.

1.2.3 Models with Predefined Behaviors: The Automatons

To start our exploration of the existing approaches that propose agent models, we study the ones simulating predefined behaviors. For this, they use Automatons or Automated agents to automatically launch behaviors in a fully controlled way: all the decision-making steps of the agents are explicitly described through graphs or scripts that display the different available possibilities that can be chosen to manage the current situation. Figure 1.3 shows some examples of such automatons. Among the existing approaches using them, we can find the Persim 3D [88] and SIMACT [28] using predefined scenarios to control their agents. we can also find approaches using Finite State Machines (FSM) models [116] where a finite number of states is represented showing the possible agent states. The FSM can change the agent state when a new situation happens or a new condition is reached. This change from one state to another is called a transition.

We also have Petri-Nets-based approaches [140] where a token is used to pass from one place to another one. The places are linked by transitions letting the token pass when the condition is reached. Transition can also trigger effects, such as animations, when the token passes through it. A multi-level improvement of these Petri-Nets can be found in the work of Claude et al. and Lecuyer et al. [49], [114], [115] where a scenario model called #SEVEN, which is implemented in the XR Toolkit *Xareus*[8], is proposed to create scenarios for Virtual Reality (VR) applications.

We can also find Behavior Tree (BT) approaches [27] where agent states can be ordered as a tree. Concretely, BT is a directed acyclic graph composed of Nodes and edges. Each parent node can take four types: Sequence, Selector, Parallel, and Decorator whereas

8. *Xareus Software*: `https://team.inria.fr/hybrid/xareus/`

child nodes can be Actions or Conditions. Parents continuously check the state of their children to know if they succeed or fail. Depending on the type of parent, the success condition will not be the same. For instance, the parent being a Sequence type returns a success when all its children return a success, whereas a Selector type returns a success when at least one of its children returns a success. Actions are used to execute methods changing the system or the environment state.

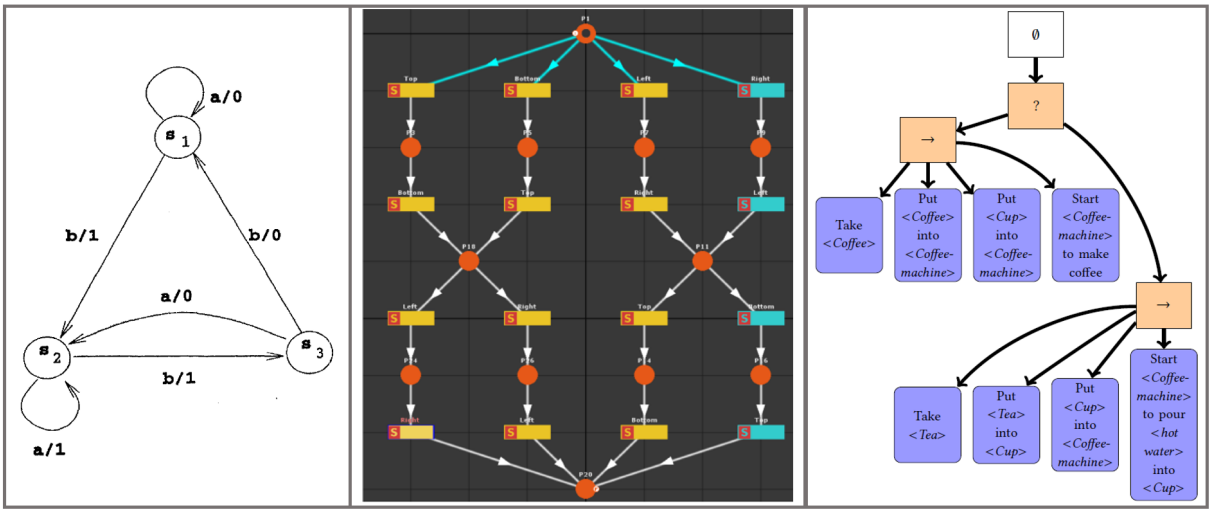

Figure 1.3 – Examples of Automaton-based approaches. On the left, an example of a transition model made with Finite State Machine [116]; On the middle, an example of a Petri-nets model coming from #SEVEN [114]; On the right, an example of a Behavior Tree used to implement the task *Preparing Coffee*

Such fully controlled techniques have the advantage of quickly producing behaviors. However, agents are not able to adapt to a situation that is not considered in the structure. They also cannot anticipate their future constraints (activity to make at specific times, unavailable resources, etc.). In addition, scalability is limited since the complexity quickly increases and the addition of new content can be a tedious task. For all these reasons, automated agents cannot offer sufficiently credible behaviors and address our first challenge. Nevertheless, they can be interesting to use as a complementary approach to manage the execution of activities for instance, in order to let the researcher control how the agent can perform an activity and thus better match with its expectations.

1.2.4 Models with Goal-Oriented Behaviors

In this section, we explore approaches using a goal-oriented agent model to simulate agents in a better autonomous way than automatons. According to the definition of autonomy given in section 1.2.1.3, an agent can be considered as autonomous if it can produce its proper goals and choose the way to satisfy them. In the next sections, we study existing works that use goal-oriented methods to obtain an autonomous agent. We then analyze these approaches to know whether they can address all 3 challenges mentioned in Introduction at the same time.

1.2.4.1 Reactive-Based Models

In reactive-based methods, the agent chooses and executes the most appropriate action according to the current context. With these kinds of approaches, the agent can quickly react to changes caused by the environment or by its internal constraints (needs, emotions, preferences, etc.).

In this category, we can find approaches using **action selection mechanisms** either based on **motivation-based methods** or **needs-based methods**. Figure 1.4 shows some examples of such approaches. Generally, a threshold system is used to assess which motivation or need must be urgently satisfied and then which activity must be performed in priority to satisfy them. The difference between motivation and need is relatively subtle. Needs are more based on what physical and mental processes coming from biological aspects can affect the human's actions whereas Motivations can encompass broader sources of action that may not necessarily come from a need (habits, emotions, preferences, etc.). Most approaches using needs are based on Maslow's definition [124] described in section 1.2.1.4. Nevertheless, motivation and needs are sometimes used as synonyms.

Needs-based approaches with internal factors : Among needs-based approaches simulating needs caused by internal factors, we have the work for Avradinis et al. [11], [12] where an agent in a 3D house satisfies its needs inspired by Maslow's Pyramid of Needs [124]. In this approach, they simulate the interdependence between needs with mathematical functions based on medical studies. In addition, Positive and negative thresholds are implemented to indicate when a need should be addressed or avoided. In the work of Lee et al. [117] and those of De Sevin et al.[159], activity selection systems are proposed to choose the best activity according to the context and the agent needs. Regarding De Sevin et al., a viability zone inspired by the work of Meyer et al. [128] is

proposed, where 3 zones estimate the urgency of needs (comfort, tolerance, and danger zones). In the work of Handel et al. [86], they simulate agents in a 3D square where needs are modeled with Dynamic systems describing how they are impacted by actions and other needs. However, in all these approaches, no external factors are simulated and few clues are given to understand the relationship between time and needs. Thus, time drifts may appear after a certain time and needs not being urgent every day cannot be simulated. This limits credibility during long time periods and their use for data generation. Finally, in the work of Bogdanovych et al. [24], where they propose to simulate the daily life of a Mesopotamian population, they compare the need-based approach developed by *The Sims*[9] game with the Goal-Oriented Action Planning (GOAP) [136]. In *The Sims* game, each need has a decay rate represented by a mathematical polynomial function bounded between -100 and 100. When a need value reaches 100, this need is considered as totally satisfied. The value of each need decreases over time and can increase when the agent interacts with objects. In this approach, each agent aims to maximize its mood value which is the sum of needs ones. In contrast, the GOAP approach uses planning-based approaches to trigger a goal when a need reaches a critical value and construct a plan of action to reach it. Both approaches are interesting since they are adapted for long periods and can manage several agents. However, the authors notice that *The Sims* game approach produces scripted behaviors, has scalability issues when a new object is added [24], and provides few hints about needs satisfaction due to its commercial nature. Challenge 1 is thus difficult to address. Regarding GOAP, a lack of anticipation is observed since agents only react to changes and do not anticipate future constraints, limiting the possibility to address our Challenge 2.

Needs-based approaches with external factors: Some approaches simulate needs that are induced by external factors coming from the environment such as temperature or humidity. This is the case in the model proposed by Kashif et al. [99], [101] where air quality, temperature, and humidity are considered in addition to hunger. In the work of Kim et al. [103], they simulate occupants trying to reduce their total energy consumption while improving Indoor Environmental Quality (IEQ) score impacted by external factors (thermal comfort, acoustic comfort, air quality, etc.). For this, a hybrid optimization function is used to optimize this ratio. Thermal comfort preferences are also simulated in the work of Klein et al. [104] and SMACH [156]. However, judging from several

9. *The Sims*, official website: *https://www.ea.com/fr-fr/games/the-sims*

reviews related to the impact of occupant behavior on energy consumption in indoor environments, the credibility of the simulated agent considering external factors is not sufficient to retrieve relevant synthetic data [58], [66], [119], [158], [180]. More particularly, according to Li et al. [119] and Ebuy et al. [66], simulations are not yet sufficiently efficient to produce synthetic data usable to predict the energy consumption caused by occupants since external and internal factors are rarely simulated simultaneously.

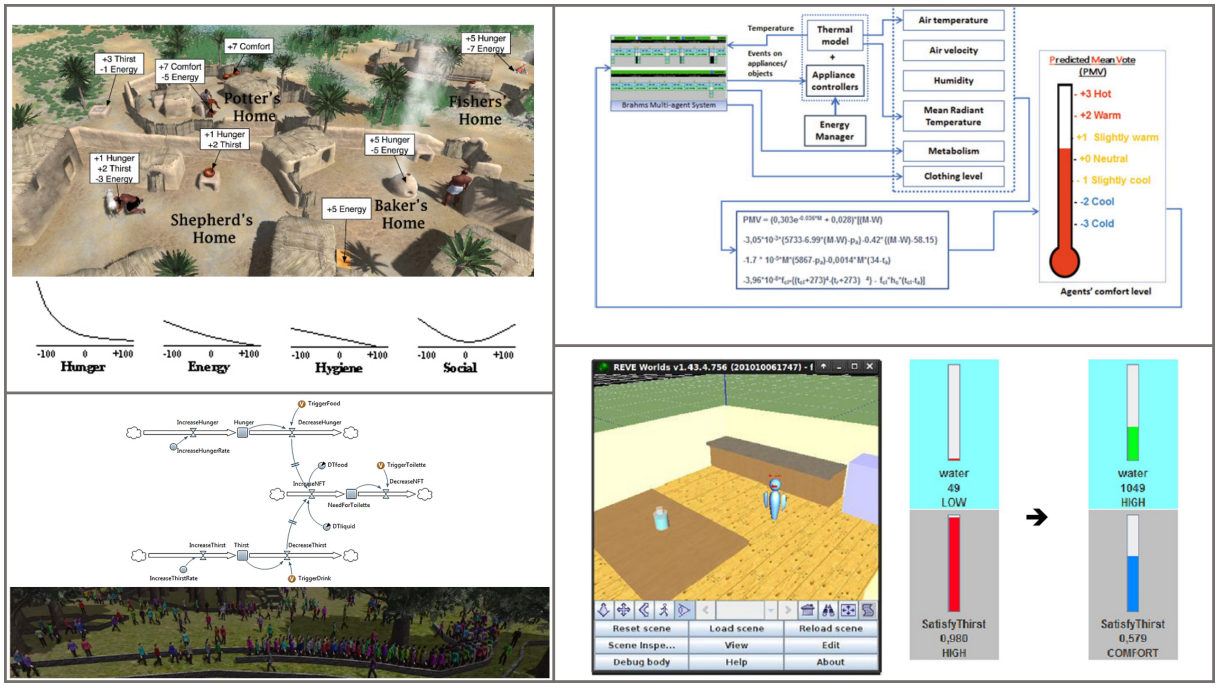

Figure 1.4 – Examples of Reactive-Based approaches. At the top left: the needs model of *The Sims* game tested by Bogdanovych et al. [24]; At the top right: the agent thermal comfort proposed by Kashif et al. [99]; At the bottom left: Dynamic systems used by Handel et al. [86] to simulate needs; At the bottom right: the MAGE model proposed by Avridinis et al. [11] to simulate needs.

Motivation-based models: Other approaches not only use needs as the main motivation for the agent but also use other cognitive factors such as preferences, emotions, personality, coping, appraisal, or social factors. For instance, Bourgais et al. propose in their works a review of the existing emotional models used in agent models [30] as well as the BEN social agent model [29] where emotions are included and based on the OCC cognitive appraisal theory proposed by Ortony, Clore and Collins [137]. According to this theory, emotions are caused by the evaluation of 3 types of stimuli: The desirability of an event in accordance with its goals, the merit of an action respecting standards,

and the attractiveness towards an object or a person. In addition, the OCEAN model (or the big five-factor model) [77] is also used to give personalities to their agents. In this model, personalities are a combination of five parameters (Openness, Consciousness, Extroversion, Agreeableness, and Neuroticism) bounded between -1 and 1. OCC and OCEAN are cognitive models that have the advantage of broadly making a consensus in the scientific community as a way to simply represent human emotions and personalities. They are used in other simulations of emotions such as the ALMA model [74], ERiSA model [47], and FAtiMA toolkit [123]. In the work of Shirvani et al. [160], emotions based on the OCC theory are used in a story-telling system as the main factor impacting the decision-making to create a story. In the HUMANS model [113], agents are simulated in critical situations through the REPLICANT module including OCEAN, OCC, and relationships between agents to manage new situations caused by the user trying to solve the situation. Although OCC and OCEAN can simplify the representation of emotions and personalities, they are sometimes difficult to configure when we want to simulate a particular emotion or personality, since they are the result of several variables. Some approaches simulate other cognitive processes such as *appraisal* and *coping* that come into play when a person is faced with a critical or unexpected situation that is hard to manage. Coping and appraisal processes are explained and simulated in the work of Marsella et al. [122] in the use case of medicine where a doctor has to make a decision causing serious consequences. As another possible motivation, some approaches use the time of day to select the next activity to perform. This is the case for SMACH [5], [149], [156] where a time-of-day-dependent probability law based on real data is used to select the next one.

BDI architectures: In addition to action selection mechanisms approaches that can use either needs-based methods or motivation-based methods, we also have other existing reactive-based approaches such as Beliefs-Desires-Intentions (BDI) architectures. This well-known reactive-based architecture, which can be shown in Figure 1.5, was created by the philosopher Bratman [33]. In BDI, a perception system interpreting the state of the world is modeled through beliefs, the choice of possible goals is modeled through desires, and the choice of predefined sequences of actions to satisfy these goals is done by intentions. When Intentions are chosen, they are executed and the Desires are updated to consider the goal satisfaction. One of the first implementations is proposed by Rao et al. [147] and surveys about BDI are given by De Silva et al. [162] and by Adam et al. [1]. BDI architecture has inspired many agent-programming languages such as AgentSpeak [147],

CAN [177], or CAN-PLAN [155] and a lot of implementations exists, giving an interesting adaptability. Among classic BDI frameworks, we have the Procedural Reasoning System (PRS) proposed by Myers [130], JACK proposed by Howden et al. [91], or Jason by Bordini et al. [26] which was also integrated into the JaCaMo platform proposed by Boissier et al.[25]. This last platform mixes three existing ones: Jason for managing the autonomous agents, Moise for agent organizations, and CArtAgO for integrating shared environments. We also have JADE, an open-source Java framework, developed by Bellifemine et al. [18] and improved by Nunes et al. [134] through the BDI4JADE platform and by Pokahr et al. [141] through Jadex. Among BDI-based implementations, some improvements were made for BDI such as preferences [36], needs [99], [153], planning [57], [179] or emotions [29]. We also have the work of Vosinakis et al. [173] where BDI agents having physiological needs simulate an ancient Greek population. We have also the work of Ramos et al. [146] using BDI agents for serious games, the MASSHA model [97] simulating BDI agents in a smart house as well as the MOVICLOUD platform proposed by Barriuso et al. [16] where disabled BDI agents are modeled in a 3D building.

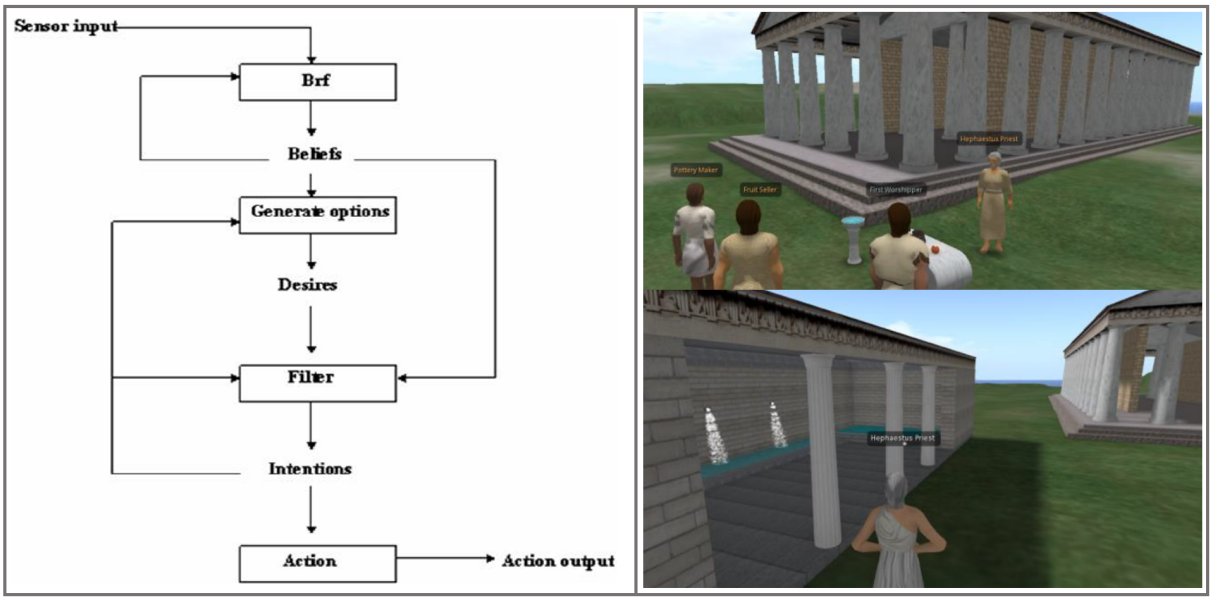

Figure 1.5 – On the left, a diagram of the global BDI Architecture proposed by Saadi et al. [153] where Brf means Belief Revision Function, used to update the agent beliefs. On the right, an example of BDI agents proposed in Virtual Agora [173]

The massive implementation of BDI architecture in the agent domain can be explained by its advantages: BDI allows the simulation in real-time of agents adapted to dynamic

environments. BDI is also a flexible architecture where extensions can be easily added. In addition, it is well-suited for MAS while allowing the agent to act in an individual way. Since the agent uses the perceived information to choose its own goals and the accurate action plan, BDI allows for both consistency and diversity in behaviors, enabling us to address Challenge 1. However, the main implementations of BDI architecture, without mixing with planning-based approaches, do not allow the creation of plans during the simulation. Actually, plans of action are rather retrieved from predefined plans stored in a library. This is why, BDI used alone is not sufficient to address Challenge 2 related to giving a compromise between control and autonomy since no anticipation process can be made with the existing platforms, except those that combine BDI with planned-based approaches which are detailed in the section 1.2.4.6.

To conclude this section, reactive-based approaches are particularly well suited to dynamic environments and multi-agent issues. Most of them can also produce credible behaviors and thus address our first Challenge. However, imposing strong constraints is more difficult since no anticipation mechanism is set up to prepare the agent for future constraints that can be imposed by experimental protocols of a real dataset for instance: the agent cannot anticipate the depletion of resources and be sure to be ready at a specific time to perform an imposed activity. Consequently, used alone, they are not sufficient to give a compromise between control and autonomy, making it difficult to address our second challenge. These approaches stay limited in the control over the agent's decision-making since the activities are chosen in reaction to a situation and not in anticipation of a future situation.

1.2.4.2 Planning-Based Models

In contrast to reactive-based methods, other approaches use planning-based methods and scheduling-based methods to set up a plan according to the current context and constraints. Unlike reactive-based approaches, it is easier to impose strong constraints on agent behavior. Among the constraints that can be managed by these models, we can find the management of resources [64], [109], goal [70], [75], space [43], [94], [129], preference [142], or time [94], [127], [148]. In the literature, confusion may arise between schedulers and planners. Judging from Smith et al. [163] **Planners** are designed to manage the ordering of actions, tasks, or activities in order to achieve a goal. The structures used can be tree, loop, or parallel actions. The aim is to find a series of actions to reach the

goal. However, they may have difficulties in managing resources and time constraints. In contrast, **Schedulers** are specifically designed to deal with time and resource constraints but may have some difficulties to order actions, tasks, or activities. However, with recent methods, the frontier between both methods tends to be less perceptible. In this section, we explore the planner-based approaches and in the next section, we study the scheduling-based approaches.

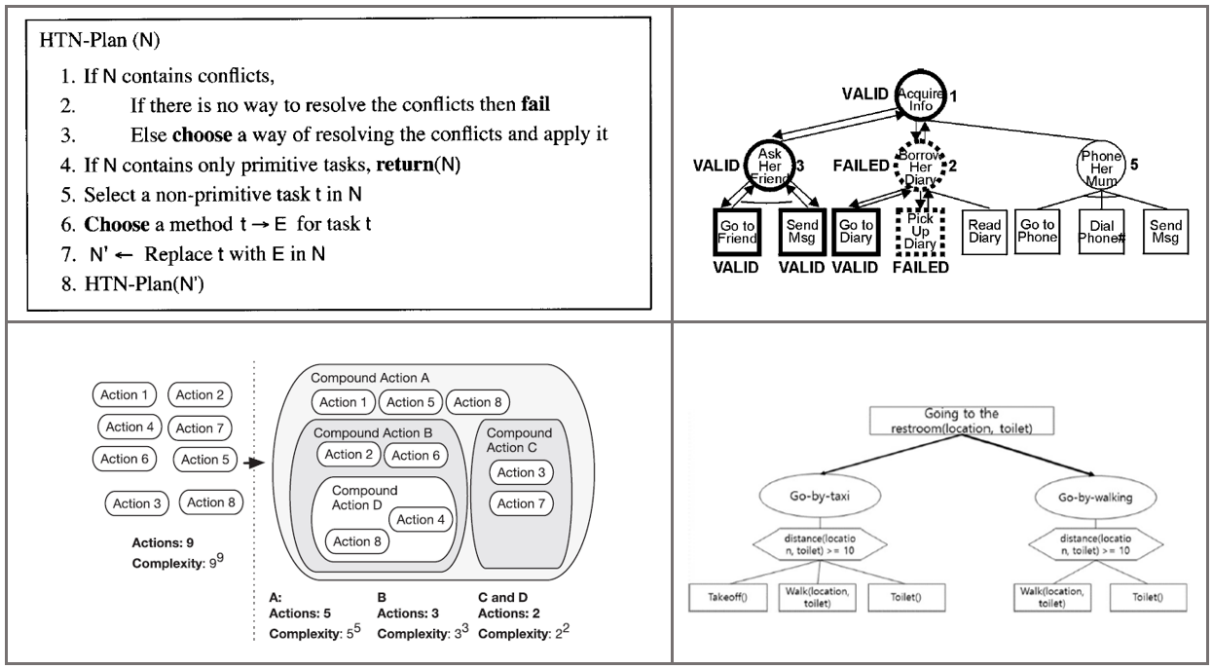

Figure 1.6 – Pseudo-code of HTN algorithms and examples of some implemented HTNs. At the top left: HTN Pseudo-code proposed by Smith et al. [163]. At the top right: Example of HTN used by Cavazza et al. [40]. At the bottom left: HTN processes and complexities described by Trescak et al. [170] and explained in section 1.2.4.5. At the bottom right: Example of HTN used by Jang et al. [93].

Existing planners are extremely broad and domain-dependent. Therefore, our study focuses on the main planner categories that can be used to simulate human decision-making. Generally, planners are used to find a way to achieve a set of goals. They also manipulate 3 inputs written in a formal language (either STRIPS [70] or PDDL [126]) that use logical predicates. The first input is the description of the world state, the second one is the description of the goal to reach, and the last one is the set of possible actions (also called operators). Each action can contain preconditions that must be satisfied in the current state to be applicable and post-conditions that can apply effects and change the world state. As indicated in the works of Smith et al. [163], McDermott [126], Trentin

et al. [169], and Köckemann et al. [105], several categories of planners can be found and are detailed below:

Classical Planning: In classical planning, the aim is to produce a solution executable in the initial state to reach a situation where the targeted goal is true. The solution is often a sequence of actions or activities. In classical planning, the problem is often treated in the same way as in graph exploration: The planner explores a set of states that can be represented by nodes and linked by edges including possible actions and allowing the transition between nodes. The resulting plan is then the set of edges to take that leads to the targeted state. Formally, a classical planning problem is defined by a tuple $< S, A, T, s_0, G >$ where S is a set of states, A is the set of actions, T is the transition function between two states stored in an action, s_0 is the initial state, and G is the set of goal states. Several techniques can be used to explore the set of states and build a plan of action and are described below:

- **Fast-Forward technique**: In this technique, the planner starts from the initial state and chooses an action where preconditions are satisfied to construct a new state by applying its effects. Search continues until the desired state is reached.

- **Backwards technique**: In this technique, the planner starts from the goals and constructs the plan going backward. To do this, an action that can accomplish one of the goals is chosen and replaced by subgoals corresponding to its preconditions. The process is then repeated until the remaining subgoals are included in the initial conditions. Backward techniques are used for instance by STRIPS [70] and its derivations such as GOAP [167], but also by the MASCARET multi-agent model [145]. In contrast to other planners, GOAP (Goal-Oriented Action Planning) has the particularity to be well suited for real-time simulations as shown in the work of Bogdanovych et al. [24], where a Mesopotamia population can manage its needs and resources through this planner. The MASCARET multi-agent model [145] is also optimized for real-time simulations since it simulates agents in 3D simulations to train Firemen.

- **GraphPlan**: In this technique created by Blum et al. [22], the planner either returns a shortest-possible partial-order plan or indicates that no valid plans exist.

A partial-ordered plan, contrary to totally-ordered plans, indicates the actions to perform without imposing a specific action ordering, unless when essential.

If we want to use Classical planners, two assumptions must be met, restricting their use. Firstly, the actions must be deterministic, i.e. we know in advance the effects they will produce. Secondly, the agent must be omniscient, i.e. it must know the state of the world in its entirety. Regarding our case, planners will systematically propose the most optimal solution, which leads to a loss of diversity in solution, which is not suitable to address our Challenge 1.Classical planners are more focused on scheduling actions according to the current situation rather than on anticipating resources and time constraints. Even with GOAP, the agent cannot anticipate when a resource will be critical, and will only act when the resource is yet exhausted. Some of them also have difficulties to manage unexpected changes impacting the planning.

Hierarchical Planning: In contrast to Classical planning, Hierarchical planning starts planning at a higher level and then goes into detail when it is required. According to Erol et al. [69], Georgievski et al. [75] and Smith et al. [163], Complex Actions and objectives, becoming *Compound Task*, can be decomposed into more basic actions called *Primitive Tasks* or in other *Compound Tasks*. Primitive tasks are those that can act directly on the environment and are often formalized with STRIPS. Among compound tasks, we have specific tasks called *Goal Tasks* only made of conditions that must be true to finish the task. To enable this kind of planning, the hierarchical description of the actions must be indicated. All the compound tasks and primitive tasks are linked with a task network called Hierarchical Task Network (HTN) [69], [75]. The possible decomposition of a compound task is stored inside *methods*. These methods allow to link between the parent task and its children's tasks and thus constitute the task network. The methods also contain the conditions for completing the concerned task. Figure 1.6 shows some examples of HTN, as well as the HTN pseudo-code proposed by Smith et al. [163]. We can find HTN in approaches such as SHOP [131], SHOP2 [132], PANDA [19], or in the work of Cavazza et al. [40]. In this last work, the authors use HTN to control autonomous agents and their interactions with the player and environment. HTN has the advantage of being easily combined with other approaches such as schedulers to reduce the search space or automatons to execute concrete actions in the environment. In our case, even though time and resource variables can be added to some HTN extensions, it remains

difficult to integrate an anticipation mechanism to foresee a future resource depletion or a future time constraint. The reactivity of the planner to manage rapid changes in the current situation is also limited.

Temporal Planning: These planners allow the management of the time and resource variables. To improve STRIPS and PDDL languages that are not well suited to manage time and durations, a new language called PDDL2.1 is proposed by Fox et al. [72] to allow the integration of actions with variable durations. Temporal planning thus allows the expression of action durations as well as conditions and effects at the beginning, at the end, and during the action. Some approaches have also proposed improvements to consider the time while pruning the search space to reduce the complexity, such as Vidal et al. [172]. Temporal planners are mainly used to solve Simple Temporal Problems (STP) that use the start and end bounds of a time interval. Temporal constraints are then propagated and bounds are reduced until a solution is found, or an impossibility is returned. As with classic planners, these algorithms will lack diversity in the solutions concerning the plans produced and have difficulties managing quick changes caused by the environment. In addition, using a temporal planner alone does not enable us to consider other future constraints such as future urgent needs or resources.

To conclude this study about planning-based approaches, we found that most of them have limitations regarding the reactivity to changes in the current situation. In addition, no anticipatory mechanisms are integrated in most cases. Moreover, planners are often used to find the best solution to a problem, which is not our purpose since we want diversity in the produced solutions. Effectively, humans do not perform activities all the time in the same way and do not choose the same plan of activities for the same situation. We thus just need to have a plan of activities that can work with the current situation without obligatory being the best one. The credibility of behaviors is thus limited and not enough to satisfy our Challenge 1.

1.2.4.3 Scheduling-Based Models

After exploring planning-based approaches, we now focus on scheduling-based approaches to know if they could address Challenges 1 and 2. Schedulers are algorithms developed to solve optimization problems implying time and resource constraints. Globally, they allow the assignment of limited resources and variable durations to tasks in order

to optimize several objectives. According to the work of Smith et al. [163], their definition is based on three principles: The core of schedulers is to reason about time and resources, they almost always solve optimization problems, and they involve choices regarding the implied resources, the duration of actions and the order of actions. For instance, a given task may have several alternatives with different costs or durations. Sometimes planners can work with schedulers to speed up their search for solutions. Several scheduling techniques have been used to manage different constraints (time, resources, space, etc.) of daily life in their simulation. We study below some essential categories of schedulers:

Constraint-based scheduler: The most common approach to solving a scheduling problem is to represent the problem as a set of constraints to satisfy. These constraints can be described either in the form of a Constraint Satisfaction Problem (CSP) or a Resource Constraint Satisfaction Problem (RCSP) and can be solved by a specified language such as Prolog. In the work of Kökemann et al. [106], they propose a constraint-based scheduler to create plans managing time during the run time that will be adapted in the work of Renoux et al. [148] to schedule the activities of agents in a 2D smart home. With this, they can force activities to start in a time interval as well as before and after some other activities. The duration is also bounded between a min and a max value and can change randomly. We also have the work of Anastassakis et al. [7], where they present a tool based on Prolog to schedule the agent behaviors in a virtual environment. Unfortunately, for all of these kinds of schedulers, agents often have difficulties to manage unexpected events that disrupt the plan during the simulation. In addition, it could be really difficult to design such constraints: The developer has to make sure that there are no conflicting, redundant, or missing constraints. This can become a very hard and tedious task, especially since all the constraints of everyday life must be considered. The addition of a new constraint must be done with great care to avoid the solver getting stuck. Different heuristic methods can be used when solutions require finding the minimum cost value or the maximum utility value. Among them, we can find the *Local Search* heuristic, where the neighbor of the candidate is explored in an iterative way. This heuristic is used in the work of Pougala et al. [143] to schedule daily collaborative activities. To do this, a utility function was set up to calculate the efficiency of the produced activities plan according to their duration, location, number of involved agents, and mode of transport. The goal is to maximize the utility function and a Local Search-based method called Metropolis-Hasting algorithm is used. Another heuristic is the *Constructive Search*, where at each level of search, the

algorithm tries to assign a value to a non-assigned variable and make a backward process when the value is inconsistent [163]. The main risk with these heuristics is to fall into a local extremum (i.e. minimum or maximum), preventing to find a global extremum.

Meta-heuristics: Meta-heuristics can combine several heuristics to find a global optimum. We can find algorithms such as simulated annealing algorithms, ant colonies, Tabu search, graph exploration, or genetic algorithms. Regarding the simulation of virtual agents performing daily activities, we can find the work of Charypar et al. [43] where a genetic algorithm (GA) is used to schedule activities during a day. In this work, a city is simulated and the agent has to schedule activities according to their distance from the location, their duration, and the opening time. The GA uses a fitness function to evaluate the performance of the produced plan. In the context of activity scheduling, a fitness function can be defined as a function gathering the utilities (or the cost) of the activities contained in the plans. The main goal is then to maximize (or minimize) the utility (or cost) of this function. In a synthetic way, GA proceeds as follows: They take a population of daily plans containing activities and calculate the utility of each plan by applying the fitness function. It will then eject the plans that have the lowest score. After this, a new plan is created by mixing the activities of two plans randomly selected. Finally, a mutation process is used in the activities of the plans with a certain probability to change, for example, their duration or their position in the plan. The algorithm then repeats until it finds a plan that exceeds a certain utility threshold or after a finite number of iterations. A Similar GA approach can also be found in the work of Meister et al. [127] to schedule daily activities performed in a house. In the work of Jorgersen et al. [94], a graph-based approach is used to generate a crowd with individual agent behavior in a virtual city. Concretely, a combination of a topological graph and an activity graph is used for decision-making. An A* algorithm is used to explore these graphs and find the best compromise between the distance, the duration, and the load carried by the agents that can change their travel speed. The agents can thus schedule their own activities based on space and time constraints. Similarly, we can find the work of ordoñez et al. [135] where a graph exploration algorithm is used to schedule activities and construct the travel path. Concretely the agent tries to maximize its utility path function. This approach has the advantage to plan activities during a time window controllable by a user. This allows for better control over the agent's activities since the user can indicate where the agent should be at a certain time and thus prepare them to perform a specific activity: we

can therefore have a certain form of anticipation and a certain control over the agent's actions at certain times. Unfortunately, this model does not consider the internal state, and the effects of activities occurring in another time window are not included. All of them limit the consistency between the behaviors and thus limit the credibility and their use for global data generation, preventing us to reach Challenge 1.

Econometrics-based models: These approaches use formulas coming from economics to schedule activities according to their costs or utility. They are often based on the optimization of a budgetary cost (such as limited time budget, limited resource budget, limited travel budget, etc.) to guide the decision during the activity scheduling process. Among econometrics-based model, we can find the work of Habib et al. [85] where a Random Utility Maximization (RUM), coming from economics, is proposed to dynamically plan the daily activities of a weekend. We can also find the work of Bhat et al. [20], [21], [79] where RUM is also used in their simAGENT model to schedule activities of each agent living in a 2D city to manage travel constraints, time constraints and vehicle constraints.

To conclude this section we saw that schedulers have the advantage of being specialized to solve complex problems implying constraints such as time, resource, space, or global costs in a reasonable time. To allow this, they provide a solution working with the situation without being necessarily the best one. Solutions are thus more diversified than planners. Unfortunately, used alone, some of them cannot concretely execute activities in an environment: they must be combined with other approaches to allow this. In addition, most of them have difficulties in quickly reacting when something happens that strongly impacts the plan. Consequently, most existing schedulers cannot really manage the interruptions of activities that can occur when an unexpected event happens (a resource is depleted at the last minute, a need becomes too urgent and requires interrupting the current activity, etc.). Finally, most existing approaches do not propose schedulers that can consider needs or motivation but only constraints related to time, resources, and space. Some limitations can appear regarding the simulation of punctual activities that not happening every day and having no specific constraints. For all of these reasons, they have limitations regarding the credibility of behaviors if they are used alone, and thus not entirely manage Challenge 1.

1.2.4.4 Probabilistic-Based Models

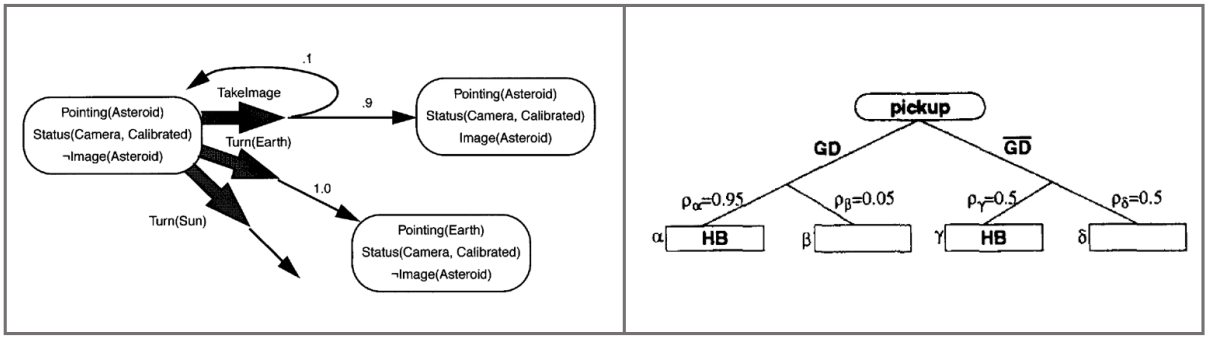

Figure 1.7 – Examples of a Markov Decision Process given by Smith et al. [163] on the left, and Kushmerick et al. [108] on the right. GD means *Gripper Dry*, HB means *Holding Block*

In the literature, some approaches try to deal with decision-making when the results of actions become uncertain. These approaches are used to select activities by following probability laws or to construct plans with activities that cause uncertain effects. In this section, we thus study two kinds of probabilistic-based models: approaches that use probabilistic planners and those using probabilistic schedulers.

Probabilistic Planners: Also called Stochastic Planners, they were initiated by the works of Kushmerick et al. [108]. In this planning, the effects of actions are described with probabilities and the goal is to construct a plan that optimizes the likely rewards produced by some action effects. An optimization process close to schedulers is thus used to maximize this reward. The decision model takes the form of a decision tree (Markov Decision Process (MDP) in its graphical form) or an influence diagram (Bayesian Network). Among approaches that use MDP for daily activities simulations, we can find the work of Banovik et al. [15] where a routine model captures the causal relationships between situations and actions. To do this, they retrieve the results of predictive models to establish the probabilities between an action and a situation. In the work of Au et al. [9], MDP can also be used to optimize the management of several agents simultaneously. Some examples of MDP can be found in Figure 1.7. The main problem with these approaches is that the impact of actions becomes uncertain. Consequently, The user is not sure that the agent will be in a desired situation at a specific moment: The agent cannot actually anticipate its needs or constraints with accuracy since even the consequences of its action

become uncertain. Probabilistic planning is useful for managing situations where the outcome is unknown by agents but at the cost of losing control over their behaviors.

Probabilistic Schedulers: They select activities and construct a plan by respecting probability laws. In contrast to the previous case, activities do not have uncertain effects but they have a defined probability to occur at a specific situation. To know this probability, real databases are often used as input to analyze the occurrence and the duration of activities according to the time of day and other parameters (such as preferences, previous activities, etc.). A probability law can be then established to select activities according to the situation. They are mainly used to obtain a routine reaching a level of credibility close to the used real data. Among the existing approaches, we can find the BIM SIM 3D approach proposed by Zhao et al. [181] where activities are randomly selected with a specific weight during the plan construction. Users can also request to enter specific activities to perform during the day, but not for a specific time. However, the control over behaviors is not sufficient since strong constraints are not sure to be respected. Roorda et al. [152] also propose the TASHA model, where probabilistic functions established from real databases are used to schedule daily activities during 24H. In the same trend, the MASSHA model proposed by Kamara-Esteban et al. [97], where an agent can perform activities in a 2D smart home. In this case, daily activities are selected according to their priority based on the desired (or mandatory) start time, duration, preconditions, current time, and preferences. If the activity is not mandatory, activities are taken randomly by considering their weight through a Roulette Wheel. However, although MASSHA can handle mandatory activities and time constraints, it is not certain whether the agent will be ready to start a mandatory activity in time. Effectively, there is no guarantee to respect the start time and duration of the required activities since they are selected just before being done, without anticipation process. Control over behavior is thus insufficient.

To conclude this section, we explored probability-based approaches, using either probabilistic planners or probability-based action selection mechanisms. They offer interesting aspects, notably on the diversity of behaviors and on the credibility of the generated routines which can be close to real ones when real databases are *a priori* used to retrieve the selection probability laws. However, strong constraints are hard to impose on these approaches, such as an activity to execute at a specific time. These approaches do not offer any certainty that the agent will be ready at the desired time since the choice is based

on probabilities and the agent does not really anticipate its future constraints. There is, therefore, a lack of control over the behavior that prevents us from meeting Challenge 2.

1.2.4.5 Learning-Based Models and Cognitive-Based Models

Some approaches use learning-based models and cognitive-based models to design their autonomous agent models. Among them, we can find approaches using reinforcement learning, case-based planning, and cognitive approaches based on the use of memory and past experiences. Among cognitive approaches, we can find ones managing past experience with rules-based models such as SOAR and ACT-R. Regarding case-based planners, memory is used to store plans that are worked in a specific situation.

Reinforcement learning: Judging from the definition given by McCall et al. [125] and Schwartz et al. [157], reinforcement learning (RL) in the case of agent models is *a form of machine learning in which software agents learn to optimize their behavior through a trial-and-error exploration of their environments.* Agents will therefore attempt to respond to a situation by trying several activities. Typically a reward system is used to indicate to the agent when it chooses the right choice. The reward is then optimized by testing several activities or tasks during a situation. The best sequence allowing it to have the best reward is then memorized. Thus, when a similar situation occurs again, this sequence can be tested in priority. Some examples of reinforcement learning can be found in Figure 1.8. In the work of McCall et al. [125], they propose to use the cognitive LIDA model based on RL to simulate human cognitive processes such as emotions, memory, decision-making, and appraisal (to quickly manage urgent situations). Some Reinforcement learning approaches [61], [93] are used to simulate needs. For instance, the PSI model proposed by Dörner et al. [61] is an RL approach where needs are simulated using liquids levels representing satisfaction levels. When the tank is not full, there is a need to satisfy. PSI structure works as a reward system: the more urgent the need is, the stronger the unpleasant signal is. The systems learn through these signals to avoid situations involving displeasure.

Among reinforcement learning methods, we can also find approaches using Q-Learning that were initially proposed by Watkins et al. [176]. In contrast to R-Learning where short-term rewards and future rewards have the same importance, Q-Learning will rather prioritize the short-term rewards than the future ones. Q-Learning methods can be found in the work of Charypar et al. [44] where they are used to build 24-h daily activities plans. In this approach, reward tables giving the utility per time slot are used to execute

an activity. This reward depends on the activity type, time of day, travel time, and starting time. Q-learning is also used by Jang et al. [93] where a double deep Q-network (DQN) approach is proposed to find the most appropriate goals according to the agent's needs, the input real data, and the time of day. To find the best sequence of actions to reach the selected goal, an HTN manager also dynamically builds in real-time an HTN containing the task sequence to perform. This approach also uses Maslow's Pyramid of needs to guide decision-making. Real data coming from Social Networks were used to train the double DQN and to learn the relationships between needs, time, and goals. The behavior of virtual humans, as well as animals living in the 3D city, are thus simulated with a level close to the ones found in real data. Generally, Reinforcement Learning approaches are efficient to produce credible data according to the unexpected situation or those that are close to those encountered before. However, agents have difficulties to manage strong constraints since they do not have sufficient anticipation mechanisms.

Cognitive-based models: Cognitive-based models use memory processes to improve the speed of decision-making when a similar situation happens. In addition, natural language, adaptation, activity selection, or emotional processes can also be set up in cognitive models. To implement them, some approaches use rule-based approaches (or symbolic-based approaches), such as ACT-R and Soar models [112] where explicit rules and previous experiences stored in memory are used to manage the agent decision-making. Different kinds of memory (Procedural Memory and Declarative Memory) are used to select the appropriate action according to the current situation. However, imposing punctual strong constraints on agent behavior is difficult since the agent reacts to the situation according to internal rules and past experiences. In the FALCON-X model proposed by Kang et al. [98], a fusion between ACT-R and a Reinforcement Learning-based model called Adaptative Resonance Theory (ART) is proposed. This model is used to simulate non-player characters (NPC) in a 3D virtual environment. Soar-based models are also found in the Virtual Human model proposed by Swartout et al. [168] where agents are simulated in critical situations.

Case-based planning: Case-based planning is a category of planners using a memory system to register plans that have already worked in a previous situation. These plans (or some parts of them) will then be reused and adapted when a similar situation occurs. If no plan works, heuristics are used to create a new plan. Among existing approaches

using Case-based planning to simulate agents in VE, we can find the work of Trescak et al. [170] which is an improvement of the previous works made by Bogdanovych et al. [24]. In this approach, the authors propose to use case-based planning to schedule activities by considering the resources of a Mesopotamian population. Another example can be found in the ADAPTS model proposed by Auld et al. [10] where a memory module is added to a planning model to improve the process of decision-making. With the reuse of past plans, These planners are faster and more robust than conventional ones. However, as with reinforcement learning, the control we can have over them stays limited.

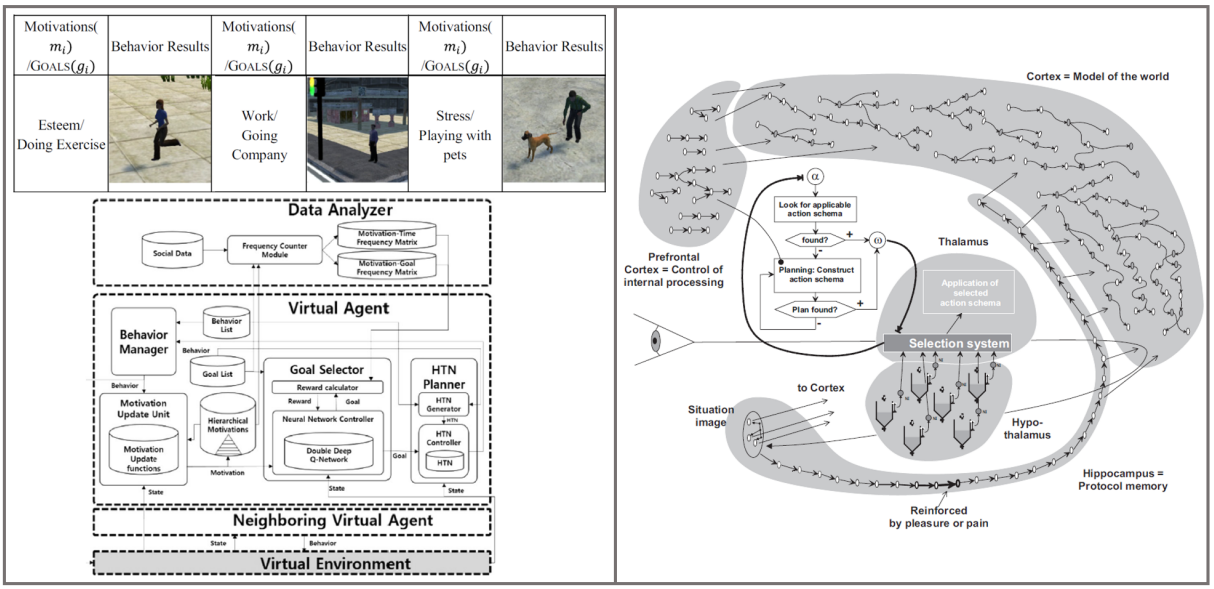

Figure 1.8 – Example of existing reinforcement learning approaches. On the left, the results and global structure of the Double DQN approach proposed by Jang et al. [93]. On the right, global structure of the PSI approach proposed by Dörner et al. [61]

Generative models: The most recent advances in AI are beginning to make such cognitive approaches more controllable, particularly with the use of a Natural Language-based model as in the work of Park et al. [139] where *ChatGPT*[10] allows the user to put a text containing the input parameters and constraints of the agents. These are approaches that became publicly available during the writing of this manuscript. However, these models are only dependent on input data: they may produce behaviors that are not consistent with the simulated environment. The control we can have over the agent is not

10. *ChatGPT*: `https://openai.com/blog/chatgpt`

obvious since it depends on the interpretation of the input text. Challenge 2 is therefore not certain to be achievable with this approach.

In this section, we explored existing learning-based approaches to know whether they could address or not our challenges. These approaches use memory to adapt more quickly to situations already experienced in order to accelerate the agent's reactivity while increasing autonomy. Unfortunately, control of the behavior stays limited, due to the nature of these learning-based approaches since the agent is fully autonomous. Imposing strong constraints can be difficult without re-training or significantly modifying the system. In addition, if we want to add or change input parameters such as new preferences or physical characteristics, it is necessary to retrain the model to adapt plans to the new preferences. For all these reasons, challenge 2 is thus difficult to reach.

1.2.4.6 Approaches mixing several Models

To obtain a more efficient and credible agent model, some approaches have merged several categories mentioned in the previous sections to recover the advantages of each method. We present here some of the existing approaches that could meet our challenges by merging several approaches.

Reactive schedulers: Reactive schedulers are a mix between reactive-based approaches and scheduling-based approaches: they can quickly adapt the plans when unexpected events disturb them. Based on the works of Smith and al. [164], they can thus be used in dynamic environments to construct plans that can be quickly adapted to unexpected situations. To do this, corrective mechanisms (replanning, reorganization, etc.) are set up to check and adapt the rest of the plan after unexpected events. Among existing approaches using this, we can find the work of Wockbe et al. [178] and those of Azvine et al. [14] where an intelligent assistant, based on reactive schedulers, is proposed to help the user with communication, information, and time management. 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. Another approach combining reactive scheduling and multi-agent systems can be found in the work of Archimede et al. [8] to manage cooperation between agents in manufacturing systems. However, no simulation of an agent performing daily activities in a virtual environment is proposed. Actually, few implementations of reactive schedulers have been used to simulate autonomous agents performing daily activities

in a virtual environment. This shows that interesting contributions can be made in this category since they have the potential to offer a compromise between control and autonomy, allowing us to address Challenge 2.

Approaches mixing BDI with planners: In some existing works, we can find combinations between BDI models and planners to create plans during execution rather than retrieving predefined plans. This is the case in the work of De Silva et al. [161] and in Xu et al. [179] where First Principle Planning (FPP) or HTN are used with BDI. However, they are not really focused on the problem of offering different levels of autonomy. Thus, the control that we can have over them is insufficient for our case. In addition, they do not propose a concrete simulation where the agent could interact with a virtual environment. In addition, no validation is proposed to validate the generated behaviors.

In this section, we study some approaches mixing diverse methods that offer better flexibility between control and autonomy. Challenges 1 and 2 could therefore be met by these methods, but few of them offer concrete implementations. In addition, they rarely focus on this ratio. They also do not propose validation methods, necessary to reach Challenge 3. This study nevertheless allows us to know which solutions are promising to achieve our goals. In the next section, we explore what solutions are proposed to validate the generated behaviors and thus address Challenge 3.

1.2.5 Ways to validate the Credibility of Agent Behaviors

In this section, we show the possible validations to assess the credibility of the agent behaviors. Since we studied in section 1.1.3 the use of real data to validate behaviors, this section more focus on other validation methods that do not require real databases.

Output-Input comparison: Among these alternative approaches, we have methods that compare the effects of input parameters on results. This is the case of Trescak et al. [170] where they statistically analyze the behavior of their agents according to the changes they made in input (overpopulation, decreasing resources, etc.). This is also the same approach used in the work of De Sevin et al. [159] where 32000 iterations of the simulation are made to analyze the time-sharing of activities, the impact of motivation over the action selection mechanism as well as the moments where the needs are in the comfort zone, tolerance zone, and danger zone. In the work of Charypar et al. [44], their

Q-Learning method is validated through the use of 3D plots showing the reward value according to the duration and the starting time (opening hours, maximum duration, etc.). Finally, In the work of Meister et al. [127], they analyze the impact of the agent' jobs on the travel and on their activity timeline generated by a genetic algorithm. The output-input comparison method has the advantage of objectively assessing whether the generated behaviors are those expected with the given input parameters. This validation is also less complex to implement than other methods since no user or real data are involved. However, the credibility assessment stays limited and needs to be combined with other validation methods to avoid design bias.

User Questionnaire: A user questionnaire can be used to ask participants whether the agent's behavior seems credible, such as in the work of FALCON-X [98], Park et al. [139] and SMACH [156]. In the work of Renoux et al. [148], participants are asked to say if the given daily activity planning has been generated by a human or an agent. Regarding the work of Darty et al. [53], a validation method combining a user questionnaire and a clustering method is used to find out whether participants can distinguish a human driving a car from an agent performing the same task. User questionnaire methods are interesting to get a subjective view of what real people think about the generated behaviors. However, biases may be present when users are not really able to identify the characteristics of behaviors or schedules that would be credible.

Human-Agent objective comparison: Some approaches compare the behaviors of an agent with those of a human in a specific situation. They then are compared with objective validation methods such as Clustering or Statistical Comparison. Regarding the Statistical Comparison, approaches compare the gap between the behaviors of agents and those of real experimenters for the same task. This validation method can be found in the work of Jorgersen et al. [95] where they study the statistical distribution of tasks sequences localized in a city between those performed by a human and those provided by their agent model. The gap between them is then analyzed in order to know if the agents can generate activities plan close to those of a human. In the work of Park et al.[139], This kinds of test is also used to compare answers given by their agents regarding questions written in Natural Language about their daily-activities schedules in comparison with those given by a real human. Regarding Clustering, they are used by Darty et al. [53], to compare a human driving a car with an agent executing the same task. In concrete terms,

the clustering method will compare logs corresponding to the same type of behavior and will create two clusters for the agent and human logs. A k-means method is then used to calculate the distance between the logs and aggregate the parts of the agent's cluster that are close to the human's ones. These approaches are interesting since they are objective. However, behaviors are often compared with a single human in a specific task, which can produce biases. Generally, these methods need to be combined with other approaches to avoid overestimating the results.

To conclude this section, we explored several ways to validate the credibility of behaviors, which can be used for our proper validations. During this analysis, we noted that few approaches in the field of planners-based approaches propose validation methods. In the majority of the remaining cases, the input-output comparison approach is favored since its implementation is simpler and fundamental to verify the consistency of the generated behaviors. However, as said before, this is not sufficient to validate the credibility of the agent's behaviors. Few approaches go further by adding new validation methods based for example on User Experiences, Human-Agent comparison, or real data. In the case of real data explained in section 1.1, the vast majority of approaches study behavior at a global level (through statistics analysis) but few at an individual level.

By studying existing validation methods, we noted that many possibilities to validate agent models exist, but few approaches combine both subjective and objective validations at both global and individual levels. There are many reasons for this, such as the lack of available real data, the lack of time to construct sophisticated studies, or the lack of human, material, or technical resources. Challenge 3 is therefore partially met by existing approaches and a combination of validation techniques is still to be tested.

1.2.6 Conclusion about Agent-Based Models

To conclude this section focusing on the ways to simulate autonomous agents, we can deduce that the literature is wide in proposals to offer an agent able to make decisions according to the environment, its own motivations, the input constraints, the possible rewards, or its past. It can sometimes execute them in 3D environments. We first studied how we could simulate the interaction between agents and virtual environments through the Intelligent Virtual Environment (IVE) where we concluded that most of the semantics methods can meet our requirements regarding the description of objects and interactions.

We then studied the approaches proposing models of autonomous agents. Table 1.1 summarises the advantages and disadvantages of each approach in relation to our three

challenges. We can see that automatons approaches do not allow us to reach our Challenge 1 since there is a lack of autonomy to offer credible behaviors. Regarding the Reactive-based approaches, they are well suited to address our Challenge 1, but they do not provide sufficient control over behaviors which is a limitation to address our Challenge 2. In contrast, Planned-based approaches have difficulties offering diversity in the solutions and managing unexpected events. Consequently, they do not fully address Challenge 1. On the scheduling side, they provide a good diversity and reactivity, while enabling a compromise between control and autonomy. Unfortunately, they still have limitations to address Challenge 1 since reactivity is not sufficient to allow activities to be interrupted when an unforeseen event occurs. Concerning probabilistic methods, they have the advantage of offering a great diversity of possibilities and are the least deterministic approaches. However, the controllability of behavior becomes more difficult since there is no certainty that the agent will select the desired activity at the desired time, making it difficult to reach our Challenge 2. We also explored learning-based and cognitive-based approaches using memory and learning to simulate agents. They have the advantage to provide great reactivity and adaptation in the behaviors when faced with a situation similar to those encountered in the past. The generated behaviors can thus provide interesting credibility since the agent has "learned" as a human would. The drawbacks are the controllability of behaviors and the difficulty to change initial parameters since re-training is often required to memorize new choices and to ensure that past experiences are coherent with the current choices. Challenge 2 is thus hard to address in this case. Finally, we studied approaches that mix several methods which are the best positioned to meet both challenges 1 and 2. Unfortunately, They do not focus on offering a compromise between control and autonomy, but rather on the best performance rate. They also rarely propose concrete methods to validate the credibility of behaviors, which is limiting to address Challenge 3. Nevertheless, hybrid approaches stay promising to address our challenges.

After this, we gather all the approaches to see which validation methods exist. We found that planners and mixed methods offered the least validation regarding credibility since their validation is more oriented on the performance rate. The most advanced validation methods are given by schedulers and probabilistic approaches. The most common method is input-output verification because it is the simplest and the most fundamental to implement since it enables to check if the algorithm respects the input parameters. However, it remains insufficient to ensure that generated behaviors are credible. In addition to validation using real data described in section 1.1.3, we can find validation methods

Table 1.1 – Contribution of the different existing approaches for our three challenges

Existing Approaches	Challenge 1 (Generate Credible Behaviors)	Challenge 2 (compromise between Control and Autonomy)	Challenge 3 (Existing Validations of Credibility)
Automatons	- - -	+	-
Reactive-based approaches	+ + +	- -	+
Planning-based approaches	- -	+ +	- - -
Scheduling-based approaches	+	+ +	+ +
Probabilistic-based approaches	+ +	- - -	+ + +
Learning-based approaches	+ +	- - -	+
Mixed approaches	+ +	+ + +	- - -

such as user questionnaires to assess the credibility of behaviors in comparison with real humans, and objective comparison between humans and agents for a same task through statistic or clustering methods. Due to their complexity of implementation, the scarcity of real data, or the high demand in terms of human resources or time, these approaches remain unfortunately in a minority in the literature, limiting the number of methods that can handle Challenge 3.

1.3 Conclusion

To conclude our state-of-the-art, we can say that many existing approaches can address some parts of our challenges detailed in the Introduction (Model able to generate credible behaviors, able to offer a compromise between control and autonomy, or able to validate the credibility of generated behaviors). However, no approach can address all of them at the same time. In addition, since we want to have an agent model compatible with data generation, we need to simulate 3D virtual environments to allow interactions and activity execution.

We first noticed that some approaches generating data have limitations related to their simulation in a 3D environment, restricting their use for the generation of smart home data and human activities. Actually, approaches not using 3D environments are not able to accurately simulate some sensors and activities. Some of them have also

limitations related to validating their simulated data. We have found two main processes for assessing the credibility of data in the literature: *A priori* approaches, relying on the use of real datasets to build behavior models and ensure a degree of credibility by design, and *A posteriori* approaches, proposing experimental protocols to compare the credibility of simulated data with real ones through the use of specific metrics. However, these metrics are often statistical arguments that do not guarantee the credibility of behavior when they are taken individually at a specific time. For example, in the case of human activity recognition, simulated samples might be statistically close to real samples but not when samples are taken individually.

Some approaches have also limitations related to their agent behavior regarding our challenges. Many interesting strategies are proposed to control agents or make them fully autonomous, but few allow flexibility between controllable and autonomous behaviors. Interruptions of activities caused by unforeseen events are also seldom managed in most proposed approaches. We believe that these shortcomings are drawbacks for the simulation of credible human activities compatible with data generation: control is important to simulate situations when we seek to generate datasets, whereas autonomy is important to have diversified, credible behaviors and long-term simulations without human intervention. Interruptions of activities are also common in everyday life and should thus be allowed. In addition, many existing works do not propose sufficient methods to validate the credibility of behaviors, even though some of them propose original validation that will inspire us for our works. We saw that the closest approaches able to address our challenges are schedulers and mixed-based approaches combining reactive-based methods with schedulers (such as reactive schedulers). However, schedulers have some difficulties in handling sudden interruptions that disturb their initial plan. Concerning mixed approaches, they do not really focus on managing control and autonomy and few implementations and validation are set up. As a result, no existing approach simultaneously addressing our 3 challenges at the same time.

Since the existing approaches does not simultaneously address our challenges, **we propose an agent model compatible with data generation and able to generate credible behaviors (Challenge 1) while offering a compromise between control and autonomy (Challenge 2). In addition, we propose several validation methods to validate the credibility of behaviors and synthetic data (Challenge 3)**. Concretely, to make our agent compatible with data generation and to address our first challenge, we propose a BDI-based model where an internal state model representing

Desire could manage needs and preferences. This model can also execute activities in a 3D environment. To address our second challenge, we then integrate a reactive scheduler to manage the strong constraints of time, resources, and preferences, but also to manage the possible interruptions caused by the needs or environment. It is also calibrated to anticipate time and resource constraints imposed by a user or other agents. Concretely, our agent model is detailed in Chapter 2 where we explain the global structure as well as the management of time constraints and needs. After this, in Chapter 3, we explain how our model can manage Dynamic Environments through the management of resources and unexpected events coming from the environment. Finally, to address our third challenge, we propose in Chapter 2 to first use an input-output validation to be sure that our agent model respect input constraints. We then propose in Chapter 3 a user-experiment protocol to validate the credibility of behaviors in Dynamic Environments. We finally propose in Chapter 4 to validate the data generated by our agent model by comparing their results with those produced by real data for two human context understanding tasks: Future activity prediction and current activity detection.

Chapter 2

Agent model allowing both autonomy and control on the behaviors

In this chapter, we present a new agent model able to give a compromise between control and autonomy in order to respect Challenges 1 and 2 presented previously. Effectively, we deduced in Chapter 1 that existing approaches do not simultaneously address our challenges at the same time. More concretely, some of them do not produce sufficient credible behaviors (Challenge 1) due to insufficient reactivity to adapt to sudden changes. On the contrary, some others cannot offer a compromise between control and autonomy (Challenge 2) due to a lack of anticipation regarding future constraints to have an agent ready at the desired time. In parallel, we show that few existing approaches provide multiple validation methods to check the credibility of behaviors and data (Challenge 3).

This lack of complete existing solutions led us to create our own agent model designed to respect these requirements. Concretely, we propose to set up a 3D animated agent whose behaviors are generated by a BDI-based model [162] enriched with a reactive scheduler able to adapt the level of autonomy according to users' constraints and the agent's internal motivations, such as needs (hunger, tiredness, etc.). As said before, users can be developers or researchers wanting to use our simulation to generate data, populate environments or improve their own simulators. This is why, they could want to control our agent model to produce specific situations or to follow a specific protocol required for the generation of their data, such as an activity calendar containing activities to perform at a specific time. This is why, allowing control over some parts of the autonomous agent model is essential in our case. In our agent model, interruption mechanisms are also added to relax users' constraints when the agent needs to have additional time to satisfy some urgent needs. These interruptions can thus improve the credibility of simulated behaviors while letting users have the choice to authorize them or not.

In this chapter, we focus on the global structure of our agent model as well the management of time constraints and agent needs (such as thirst, tiredness, etc.). Our system is also designed to support Dynamic Environments with rescheduling methods to be compatible with data generation requiring several agents, limited resources, or unexpected events (this part is more detailed in Chapter 3). We also provide in this chapter a first functional validation based on an Input-Output comparison method. As explained in Chapter 1, this method is used to know whether the outputs of our model are coherent with the input parameter.

More concretely, we explain in this chapter the initial requirements that lead us to create an agent model in Section 2.1. In section 2.2, we explain how our agent can interact with a 3D environment to be compatible with activity data generation. In section 2.3, we explain the global structure of our agent model and we detail its components related to *The Internal State Model* in section 2.4, *The Decision-Making Model* in section 2.5, and *The Task Executor Model* in section 2.6. The functional results based on Input-Output Comparison methods are summarized in Section 2.7. We conclude and discuss about our agent model in section 2.8.

2.1 From Specifications to Agent Model

Our agent model was designed to respect specifications coming from our challenges and our necessity to produce a model compatible with data generation. Although our model could be used for other purposes such as populating environments, The Orange industrial context of this thesis requires the creation of an agent model for data generation. Effectively, these synthetic data are needed to train some algorithms specialized in human-context understanding which requires a large amount of labeled data. In this use case, 3D environments are chosen as simulations for several reasons. Firstly, as developed in Chapter 1, 3D environments are well-suited to generate any daily activity data since they enable precise simulation of environment-dependent sensors and a finer granularity of the performed activities. Secondly, these 3D environments are also used by Orange researchers to simulate test environments that do not exist in reality and to create digital twins of real connected buildings (i.e. virtual replicas of real buildings in which the states are synchronized in real-time with those of real buildings). All these reasons lead to our first requirement: *our agent must accurately interact with 3D environments and thus contain a process enabling it.*

Our thesis was initially created to help Orange researchers to make their agents autonomous since the latter had predefined behaviors that degraded the credibility of their synthetic data. This issue also led to our thesis problem: *Proposing a virtual human able to produce credible behaviors while being compatible with data generation.* As detailed previously, three main challenges were stated to address our thesis issue: **Automatically producing credible human behaviors (Challenge 1), giving a compromise between control and autonomy on behaviors (Challenge 2), and validating the credibility of these behaviors (Challenge 3).** These challenges allow us to build the following specifications for our agent model:

To address Challenge 1, our agent model has to produce credible behaviors. Judging from the definition of credibility detailed in the Introduction, *Autonomy* is an essential part since the agent can choose its proper objectives. These goals can be related to satisfying human needs (hunger, thirst, etc.), preferences, or even emotions. This is why, *we need to integrate a process able to generate the proper agent motivations* such as needs, and *a decision-making process to automatically select activities satisfying these motivations.* In addition, three requirements must also be respected to offer credibility:

- **Coherence** in the reactions of the agent and its motivational states. Concretely, this means keeping **coherence with what motive the agent**: activities must be chosen by respecting the agent profile (preferences, etc.) and the urgent need which must be satisfied in priority. In addition, the agent must be **coherent with the environment state**: interactions with objects must be accurate (opening a door and not crossing it, turning on the TV before watching it, etc.) while considering the world constraints and its available resources. Finally, having **coherence regarding reactions faced to unexpected disturbances**. When an unexpected event happens either coming from the agent internal state (needs being too urgent, etc.) or environment (resource depletion, the intervention of another agent, etc.), *our agent must be able to interrupt activities in progress.* All activities must be interrupted in a coherent way (for instance, if reading must be interrupted to answer the phone, the used book must be first closed and the agent has to wake up before going to the phone). All of this requires having an adaptive agent model to face unexpected situations. *This is why a perception model must be included to update the world state and communicate with the other models to quickly adapt the decision and motivations according to the new situations.* All the parts of our agent model are thus concerned with coherence.

- **Consistence** of behaviors in similar kinds of situations. To do this, *our decision-making model must recognize the set of activities solving the same situation and select one of them when this situation happens* (e.g. if a *Hungry* must be satisfied, the *Eating* activity must be selected and not *Sleeping*; if the phone rings, then *Answer the phone* must be selected; etc.).

- **Diversity** of behaviors. Since humans do not systematically choose the same activity or the same task sequences to reach a goal, our agent must be able to perform activities in different ways. Even with routines, our model has to create variations between the days to avoid the agent repeating the same pattern with the exact activity sequences, start times, and duration, as a robot would. To do this, *our execution model must have the choice between several ways to perform activities.* In addition, *our decision-making model must be able to change activity duration, select randomly equivalent activities, and schedule recurrent activities with some variations in their start time* (ex: For Eating, our agent cannot exactly go to eat at 12:02 p.m. every day).

To address Challenge 2, our agent model must give a compromise between control and autonomy to be compatible with data generation. As said in Introduction, the users of our agent model such as researchers could want to impose strong constraints when they create synthetic data, to simulate for instance specific situations, or respect the same protocols used to create the real database they desire to enrich. To allow this, our decision-making model must be able to respect strong constraints while keeping credible behaviors. These constraints can be related to input parameters where users can indicate the activities to perform at a specific time, agent profile (preference, etc.), or even initial available resources (treated in the next chapter). Anticipation mechanisms must be set up to make the agent aware of these future constraints and thus adapt its schedule according to them. Judging from the conclusion of our state-of-the-art, addressing Challenge 2 thus *leads us to use a reactive scheduler in our Decision-making to create activity plans, anticipate future constraints, and allow plan adaptation when faced with unexpected events.*

To address Challenge 3, the credibility of generated behaviors must be checked. This is why, we use several validation methods to assess credibility. *We must first use an Input-Output comparison method to compare the resulting behaviors with the input parameters.*

We can thus assess whether our agent can produce coherent behaviors in comparison with the input parameter, but also consistent behaviors facing a similar situation (e.g. *Eating* activity is systematically chosen when *Hungry* is urgent) while keeping diverse behaviors to avoid identical routines causing loss of credibility. As said in the stat-of-the-art, the Input-Output comparison method is essential but not sufficient to validate the credibility of behaviors. *It will therefore be necessary to combine it with other validation methods* such as the performance obtained by using synthetic data in comparison with real ones.

In the next sections, we explain more precisely the contents of our agent model, but also how the agent can concretely interact with the VE (section 2.2) and the results obtained with our agent model (section 2.7).

2.2 Interactions between the 3D Environment and the Agent

Before describing our agent model, we explain how the agent is able to interact with a 3D environment in order to be compatible with our use case and more generally for data generation. Effectively, 3D sensors and effectors are used in our use case to generate synthetic data. The animated agent is situated in a 3D smart home created with *Blender*[1] and simulated on *Unity Engine*[2]. This example of a smart home, depicted in Figure 2.1, is the environment used in this chapter, but other 3D smart homes and buildings are available and could also be employed, as shown in Figure 2.2. This virtual home is a replication of the 2-story real apartment used to collect the Orange4Home dataset [51]. To create this connected environment as well as the others described in this figure, a Virtual reality (VR) tool given by Lacoche et al. [111] is used to easily place interactive objects such as furniture but also virtual sensors and effectors.

All virtual objects, whether they are sensors, rooms, or tangible objects such as doors or plates, must be defined and classified to be identified and correctly used by the agent. To do this, we use #FIVE, an object-relation-based model proposed by Bouville et al. [31], and its implementation in the XR toolkit *Xareus*[3]. With this toolkit already integrated into Unity, we can add new object types and relations in a reasonable time (around 15 min). This information allows the agent to interact with the VE since all the interac-

1. *Blender*: `https://www.blender.org/`
2. *Unity Engine*: `https://unity.com`
3. *Xareus Software*: `https://team.inria.fr/hybrid/xareus/`

Figure 2.1 – 2-story virtual apartment used in our validation experiments and replicated from the real one used in the creation of the real Orange4Home database [51]

tions can be described for each 3D object represented. As mentioned in Chapter 1, this description is essential to turn a 3D static VE into an accessible VE where the agent can automatically recognize and interact with the objects required for its activities. It also avoids aberrations such as an agent washing a telephone instead of a plate. In addition, #FIVE allows the classification of all objects of interest, according to the actions and behaviors they can have. With this model, we can group objects by interaction similarity describing how the objects and the agent can interact. We can also construct a hierarchy between them to produce more specific interactions. For example, we can group sofas and chairs of the VE in the same category, since they share the same interaction: the agent can sit on them. In order to make two objects interact with each other (for example an agent with a chair, a glass with a water tap, etc.), we can assign a *relation* between them, which can include a 3D animation representing this relation. For example, for the relation "Washing a plate", an animation is assigned in our VE such that the agent makes circular motions with its hand on the plate that it holds. Our animations are based on two main blocks: Inverse kinematics with *FinalIK*[4] solution and a *bank of animation*[5] [56] storing everyday activities.

The agent displacements made in the VE are based on a navigation grid, called *NavMesh*[6], which is part of Unity Engine. Using a collision detection algorithm, Unity computes a Navmesh showing the grid of places where an agent of a given height and width can fit in the VE. This grid forms the basis of a graph used by the agent to choose what path to select to go from point A to point B.

4. *FinalIK:* `https://assetstore.unity.com/packages/tools/animation/final-ik-14290`
5. *Mocap:* `http://mocap.cs.cmu.edu/`
6. *Unity NavMesh*: `https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html`

Figure 2.2 – Other examples of 3D connected buildings where the agent was simulated

2.3 Global Structure of the Agent Model

We present in this section our agent model used to simulate the human decision-making process involved in the choice and execution of daily activities. The structure of our model is inspired by BDI architectures which can be shown in Figure 2.3. As said in Chapter 1, BDI is composed of three parts: The *Beliefs* simulating a perception system to interpret the state of the world, the *Desires* storing the possible goals updated in accordance with *Beliefs*, and *Intentions* using a filtering system to choose the predefined sequences of actions satisfying the goals given by *Desires*. When the sequences are chosen, they are executed and *Desires* are updated to consider the goal satisfaction.

We choose to base our model on BDI for its intuitive and flexible approach to the human decision model. In addition, it was chosen for its compatibility with our requirements: BDI can theoretically allow us to combine modules managing autonomy (such as needs in Desires) and also modules dedicated to the control of behaviors (such as the insertion of schedulers in Intentions) even though the existing BDI implementations rarely focus on it. The other important reason is its compatibility with several cognitive theories that can be used as a basis for simulating credible human behavior. Among them, we find the Norman theory of action [133] explaining through the *Action Cycle* what drives

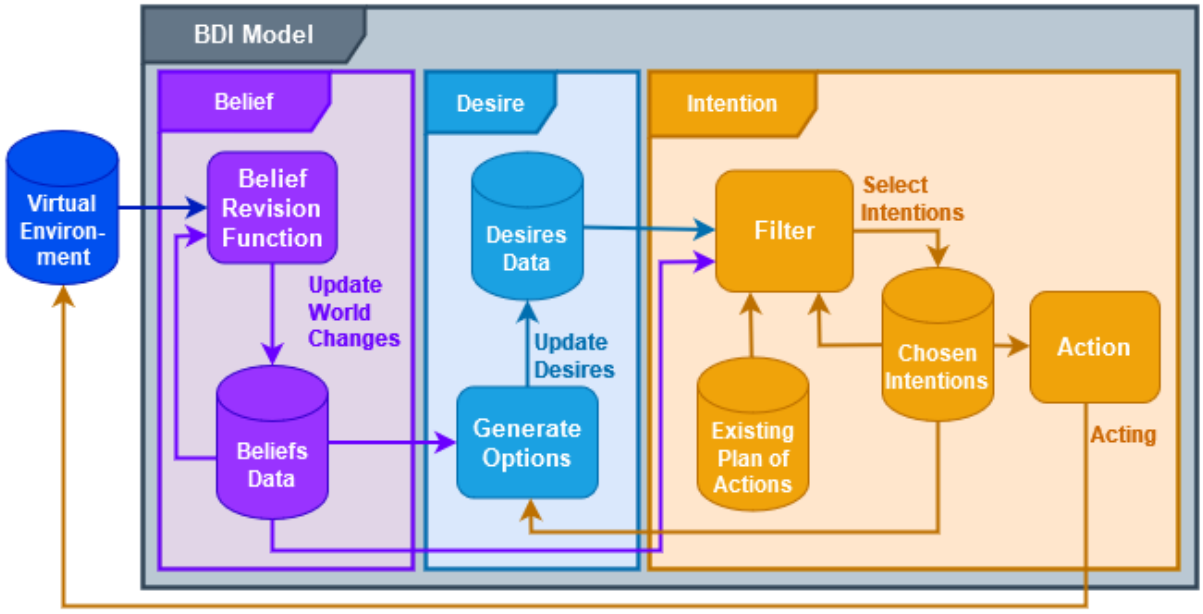

Figure 2.3 – Diagram of the BDI Architecture inspired by the diagram of Saadi et al. [153]

humans to perform actions in specific contexts. According to him, humans use a system of perception to compare the world state with their internal needs. If needs cannot be satisfied in this world state, a series of actions are set up to satisfy them. This concept is really close to what BDI proposes, showing a certain consensus in the literature on what could represent a relatively credible model of human decision-making.

All these reasons explain our choice to base our agent model on BDI since it is a flexible and intuitive architecture, able to implement what Norman theorized. However as shown in Chapter 1, BDI models are reactive-based approaches that more focus on the reaction of humans facing situations than on the anticipation of future constraints. They are thus adapted to manage the autonomous part of our model but not really for anticipation processes, making it difficult to address Challenge 2. This is also not enough to manage our use case where we want to consider users' constraints which can be a calendar of activities to perform at a specific time for instance. As said previously, these users can be researchers or developers wanting to generate synthetic data through the use of our agent model. Among the methods studied in Chapter 1, we concluded that mixing approaches such as using reactive schedulers [164] are promising to manage our Challenges. Unfortunately, few implementations exist and they do not really focus on these issues. This is why a reactive scheduler was added to our BDI-based model in order

to manage simultaneously autonomy and strong constraint anticipation. In addition, our model can take as input a calendar that lists all activities to be performed at a specific time. This allows the user to impose easily specific activities that might be required to replicate a real database such as Orange4Home [51].

Our model is also built to handle dynamic environments through our reactive scheduler able to reschedule if situations have changed. Our approach is also modular, allowing us to change each function indicated in each sub-models, without modifying the structure. Our agent model, described in Figure 2.4, is made of four models, in contrast to BDI where three are used. In fact, we choose to create the *Task Execution model*, in addition to the ones existing in BDI, since we want to have better accuracy in activity execution. We also add the *Agent Parameter* process to centralize the initial agent settings so that the user can configure it more easily. Based on this organization, our agent model is structured as follows:

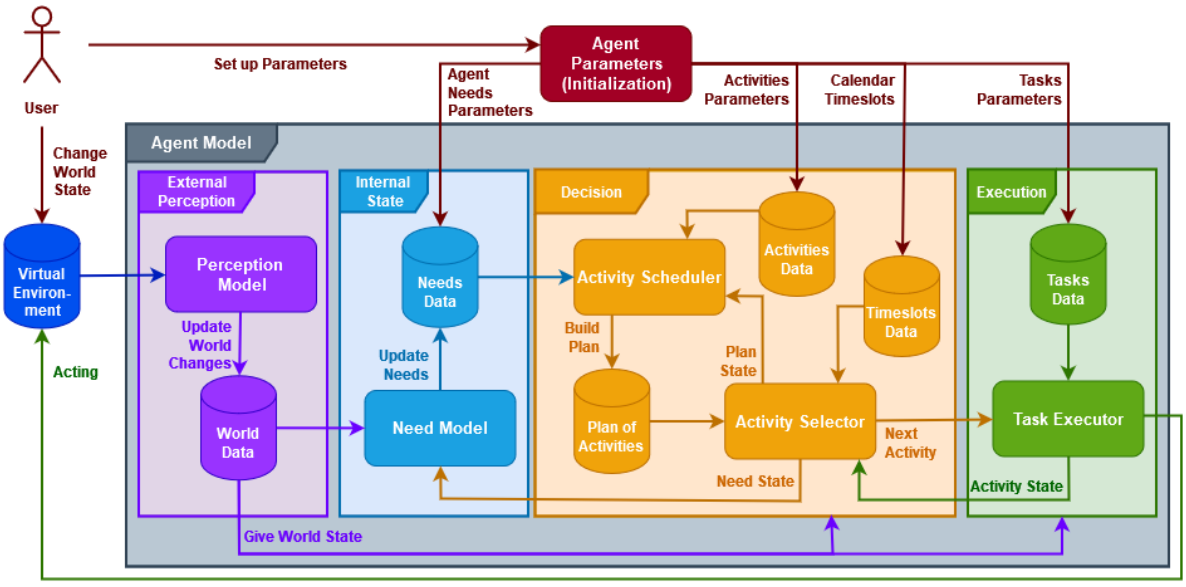

Figure 2.4 – Diagram of our proposed Agent Model

Agent Parameters : This process gathers all initial parameters and user constraints that must be considered during the simulation. The user can give an input activity calendar to provide activities that must be performed at a specific period. **This calendar gathers all the mandatory activities given by the user. These activities are**

mainly used in the Decision-Making Model to be integrated into the plan with the correct times and duration. For instance, a Mandatory Activity can be a Medical Appointment on Thursday, March 3 from 3 p.m. to 4 p.m. The user is also able to configure activity, need, and task features such as duration or occurrence. By modifying the needs parameters, specific agent profiles can be created (the agent eating more often, going to bed later, etc.), thus increasing the diversity of behaviors. All these constraints are used for the control of the agent's behavior.

External Perception model : This model is similar to Beliefs in BDI since all relevant data from the VE are stored in its world database managed by the #FIVE model [31]. Our agent is for now omniscient in contrast with BDI, explaining why we call this part the External Perception model rather than Beliefs. This model is mainly used to filter activities according to their constraints and provide the objects and interactions needed to make the execution possible in the VE.

Internal State Model : Similar to desires in BDI, it models the agent's motivation to perform an activity. However, in contrast to Desires, other cognitive models could be added such as preferences or emotions. For now, it is mainly used to update the urgency of needs and apply the input agent profile on them, such as preferences. Needs are inspired by human fundamental needs defined in Maslow's theory of needs [124]. They can be physiological, such as hunger, or they can be more sophisticated, such as self-esteem. This model is used for the autonomy of the agent since it defines goals that must be reached during free time periods. It is also involved when an interruption must occur. The needs are configured by using a temporal function, as in the work of De Sevin and Thalmann [159]. Concretely, to calculate the urgency of needs, a priority value is evaluated according to a tolerance threshold, used to know when a need becomes urgent, as well as the need intensity evolving through time. The user can also indicate whether a need is able to interrupt activities in case of urgency.

Decision-making Model : This process manages the agent decision-making, and can be related to Intentions in BDI models. However, in contrast to BDI, our model does not retrieve a predefined plan of activities but instead sets up plans during the simulation. This plan is built to respect user constraints while producing autonomous choices to satisfy needs during free-time periods. Our decision-making model contains two main

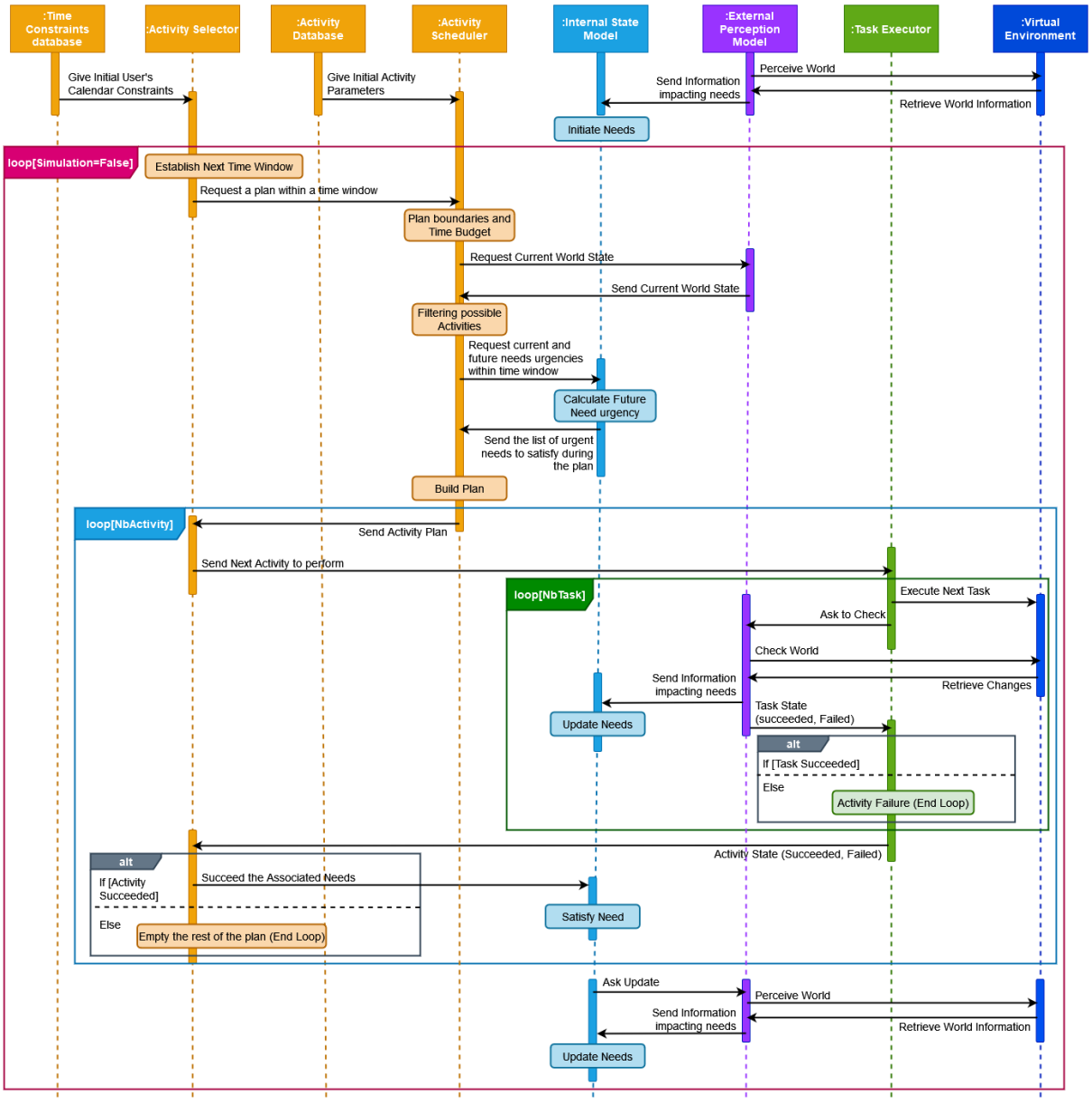

Figure 2.5 – Sequence Diagram of the Decision-Making process showing the interactions with other models

processes shown in Figure 2.4: the Activity Scheduler and the Activity Selector. These processes, as well as their interaction with other ones, are shown in Figure 2.5 through a sequence diagram. On the one hand, the Activity Scheduler builds an activity plan within a time window given by the Activity Selector. This scheduler considers both initial constraints stored in the Agent Parameters and needs whose levels of urgency in the present and the future are computed by the Internal State Model. Our Activity Scheduler is designed to be able to reschedule at any time, making it compatible with dynamic environments. Interruptions are also managed when a need is too urgent and when an activity can be interrupted. During the creation of the plan, activities satisfying needs are placed after assessing the approximate time when their need becomes urgent within the plan period. If the agent still has free time after scheduling activities satisfying its needs, default activities, such as entertainment activities, are added to keep the agent busy. These default activities are chosen randomly with optional weights affected by the agent's preferences. The approximation of needs urgency as well as the random choice of the default activities allows us to introduce diversity in the behaviors while remaining coherent: the same input constraints do not cause the same sequence of activities in output. On the other hand, The Activity Selector gradually retrieves the activity to be performed from the generated plan. If no activity can be performed anymore or if an activity failed (i.e. prematurely stopped for various reasons), the Activity selector sends a message to the scheduler to rebuild a plan until the next mandatory activity or for a predefined time window. The Activity Selector also transmits the selected activity to the Task Execution Model and receives the activity state in return. In BDI, a single filter function is used to select Intentions. Effectively, since the plans already exist, filtering is sufficient to retrieve a plan. However, in our case, since we want to build plans during the simulation, we replaced the filtering function with our Activity Scheduler. Regarding the Activity Selector, we add it to supervise and synchronize the scheduler with the rest of the system: the Scheduler is launched when the Activity Selector requests it to schedule a given time period. The scheduler thus creates a plan step by step and not over a whole day or even a week. This allows us to limit complexity while allowing greater reactivity: if an event occurs, only a small part of the day is impacted. The agent is thus able to adapt more easily than if it had planned for the whole day. This idea of partitioning plans for better reactivity is inspired by the work of Ordoñez et al. [135] where a user-defined time window is used to define the scope of a genetic scheduler algorithm.

Task Execution model : This process executes the selected activity in the Virtual Environment (VE) by executing the related task sequence. These tasks are made of basic actions and animations that can be directly executed in the VE. For instance, the activity *Showering* includes the task *Getting dry*. This process is new in contrast to BDI, and it was chosen to allow better management of the activity execution. This model receives the activity to perform from the Decision-Making model. In exchange, the activity state is returned. In our use case, the sequence of tasks is represented by a Petri-Net-based scripting model called #SEVEN [49] implemented in *Xareus Software*[3], where a token, moving from one place to another place, triggers an interaction defined by the #FIVE [31] semantics model described in section 2.2. The token progression can be managed by checkpoints verifying if the conditions are achieved. #SEVEN have several advantages for activity execution: We can launch them in parallel (allowing multi-tasking), create junctions to offer diverse way to perform a task, design loops, make scenarios without any code [115], easily see the advancement of activities through visual feedback, control the execution of the task with checkpoints and easily modify scenarios.

2.4 Internal State Model

The Internal State Model simulates all the human internal modules that can influence the decision-making process, as shown in Figures 2.4 and 2.5. It can be related to fundamental needs such as hunger, thirst, or tiredness, but also other modules such as preferences or emotions. In this thesis, needs and preferences coming from the agent profile are considered, but other processes could be added such as personality or emotions. The agent internal state is used by the Decision-Making mode for the autonomy phases when the agent has free time and must decide which activities to perform. It is also implied in the interruption process since it indicates to the Activity Scheduler what needs are still urgent to satisfy after the designing of a plan.

All simulated needs are inspired by Maslow's Pyramid of Needs [124] showed in figure 2.6. In this theory, needs are ordered according to their level of urgency. At the bottom of the pyramid, we have physiological needs that are basic but imperative to satisfy. For instance, we can find *Hungry*, *Thirst*, *Tiredness*, or *Toilet* needs. On the contrary, at the top of the pyramid, we have the most elaborate needs that are more complex but less urgent to satisfy. For example, we can have *Hobbies* or *Sport* needs. To integrate Maslow's hierarchy, an integer value called Pyra $\in \{1, \ldots, 5\}$ is introduced where 1 corresponds to

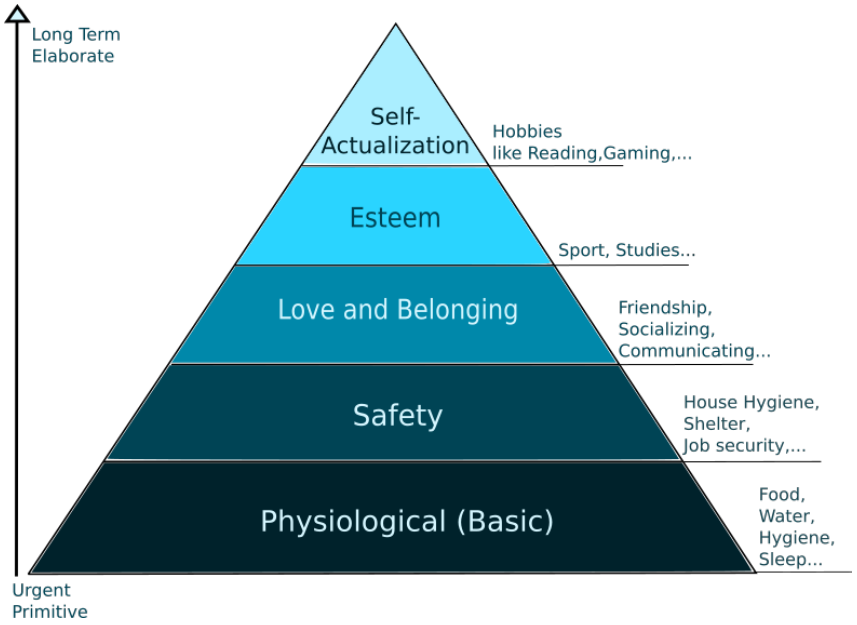

Figure 2.6 – Representation of Maslow's Pyramid of Needs

basic needs and 5 to the most elaborated ones.

Needs are configured by using a function depending on the time that calculates their level of urgency throughout the simulation (see Equation 2.2), as in the work of De Sevin and Thalmann [159]. Contrary to existing need-based approaches, our function can simulate needs that are not urgent every day. Effectively, unlike state-of-the-art approaches, the relationship between time and need is more important, enabling us to calculate the precise evolution of a need over several days. In addition, needs are not reset each day, allowing continuity through time. We can thus simulate occasional activities that do not happen daily, such as *Doing Sports* which can be performed every day, a few times a week, or a few times a month. The use of a temporal function is not the only way to simulate needs. Effectively, other approaches use fuzzy-logic approaches to simulate them such as in the work of Jang et al. [93]. Both approaches give satisfactory results, but the urgency of the need with a fuzzy-logic approach is less predictable. We thus preferred to use a more deterministic approach to ensure the exact number of needs to be satisfied in a period of time. It is effectively a decisive element in the scheduling process since the entire construction of the plan is based on the number of needs to be satisfied in a given period of time. However, The use of fuzzy-logic functions could be used to configure the agent internal state model from real statistics, as shown in the work of Jang et al. [93].

Therefore, our model is an example of motivation-based model that can be modified and improved at any time to make it more credible and effective.

For now, each need has several input parameters that can be configured by the user. The first one is the Pyra parameter described above. We then have the input parameters involved in the three following steps used to calculate the urgency of a need called P_{Need}:

(1) Tolerance threshold Initialization $\mathbf{Th_{Need}}$: This threshold seeks to simulate the tolerance limit for a need, i.e. the moment when a person feels urgent to satisfy this need. For instance, it could represent the point where someone starts to feel thirsty and think about how to satisfy it. People have different tolerance limits depending on their habits or preferences: For example, someone who likes to eat might want to eat more often. This is why our threshold value is variable. In our simulation, The tolerance threshold is used to indicate when a need becomes urgent to consider. Each threshold has a default value $\text{Th}_d \in (0, 1)$ that can be editable by the user. In our work, we put a neutral value of 0.5. Preferences could influence this threshold by deviating the threshold from its neutral value, thus modifying the time when the need is considered urgent. The closer this value is to 0, the closer the urgency thresholds will be in time, and the more frequently the agent will choose an activity satisfying this need. For instance, if we want to simulate an agent which is always hungry, we will set the hunger threshold close to 0. This would allow us to simulate various personalities such as being greedy, energetic, and so on.

(2) Need intensity $i(t)$: This parameter simulates how intensely a person feels a need: the greater the intensity is, the more disturbed the person will be by the concerned need. This intensity can vary due to several factors: too much time has passed since the last need satisfaction, a performed activity has increased the intensity, etc. For example, someone who has not eaten for a while will become increasingly hungry and obsessed with its satisfaction. In our case, all needs have an intensity value evolving through time. Equation 2.1 shows how the intensity is calculated. This value will evolve over a time interval $\text{T} \in (t_{\text{start}}, t_{\text{end}})$. Inspired by the work of De Sevin and Thalmann [159] and Avradinis et al. [11], the intensity curve is a semi-parabola going from $i(t_{\text{start}}) = 0$ to $i(t_{\text{end}}) = 1$. These parameters (intensity value and period of time) allow us to control the evolution of needs intensity over long periods. When the intensity reached its maximum value, it will stay at this value until the satisfaction of the related need. There is two way to configure the time interval $\text{T} \in (t_{\text{start}}, t_{\text{end}})$:

— **Specific hours**: The time interval is configured so that intensity starts and peaks at specific times. For instance, we can constrain *Hunger* to start at 12 p.m. and peak at 13 p.m. every day.
— **Periods of time**: The time interval is set so that intensity peaks at regular time intervals. For example, to simulate the *Toilet* need, which is not constrained by specific hours, we can set a time slot for its intensity at regular intervals of 3 hours.

$$i(t) = \left(\frac{t - t_{\text{start}}}{t_{\text{end}} - t_{\text{start}}}\right)^2 \tag{2.1}$$

(3) Need priority $P_{\textbf{Need}}$: This parameter enables the agent to know which need is urgent and has priority over another. In concrete terms, it illustrates the analysis that someone would make about his needs: *I want to sleep more than to eat, I am more thirsty than hungry*, etc. The priority of each need is then used to orientate the choice of activities accordingly. The need priority $P_{\text{Need}} \in (-1, 1)$ evolves proportionally to the need intensity $i(t)$ while considering the level in Maslow's pyramid of needs Pyra and the Threshold value Th_{Need}, as shown in Equation 2.2. The evolution of P_{Need} is thus a linear function of the intensity. When the needs priority value P_{Need} reaches the threshold value Th_{Need} described above, the corresponding need starts to be a priority. Pyra is used to change the slope coefficient of P_{Need}. This allows primitive needs to have a higher priority over a more elaborate one, even though they have the same Threshold Th_{Need} and the same Intensity $i(t)$. This priority could also depend on something else like another need. For instance, if the agent drinks, then the need to go to the *Toilets* could evolve faster. However, this interdependence is not implemented for now to facilitate the anticipation of urgent needs during the scheduling phase.

$$P_{Need} = \frac{i(t) - \text{Th}_{\text{Need}}}{\text{Pyra} \cdot (1 - \text{Th}_{\text{Need}})} \tag{2.2}$$

For each need, the user can also modify another parameter which is the ability to interrupt an activity when the related need is urgent. This parameter is useful when an activity plan is built by the decision-making module described in the next section. If this latter is not able to place in the plan an activity satisfying the urgent need, the Internal State can indicate to the Decision-making Model that this need is ready to interrupt. If the activity is interruptible, an interruption can be triggered to integrate the need in the

slot of this activity.

2.5 Decision-Making Model

The decision-Making model is the main process to control the level of autonomy according to strong constraints. The goal is to produce an activity plan including the user's constraints and the Internal State ones. It also selects the activity to send to the execution. Two main processes are involved to do this: the **Activity Scheduler** and **Activity Selector**. They are described below.

The Activity Selector : It is used to select the next activity to send to the Task Executor. In return, the activity state (started, finished, etc.) is retrieved. When the Activity Selector sends the activity to the executor, it also gives the duration of this activity. The Activity Selector also communicates with the Activity Scheduler to gradually retrieve the activities of the generated plan. When all the activities of the plan have been performed, the Activity Selector asks the scheduler for a new plan. To do this, as shown in Figure 2.5, The Activity Scheduler first establishes the time window to schedule. This window will start at the current time and ends at the next mandatory activity indicated by the user's calendar. This mandatory activity is also sent to the Activity scheduler to be inserted into the future plan from the start. With this, the activities given in the user's calendar are sure to be considered at the right times. If the input calendar is empty or if no more mandatory activity must be performed, then a default duration, configurable by the user, is employed.

The Activity Scheduler : The Activity Scheduler is inspired by the principle of reactive schedulers where some implementations can be found in the work of Smith et al. [164] and Azvine et al. [14]. Since our scheduler can be relaunched at any time, our model is also compatible with dynamic environments. The use cases showing these features are more developed in chapter 3, where additional models are introduced to manage resources and unexpected events. The scheduler uses several functions to build a plan described below and summarized in figure 2.7. At the same time, Figures 2.8 and 2.9 give some examples of a concrete plan construction.

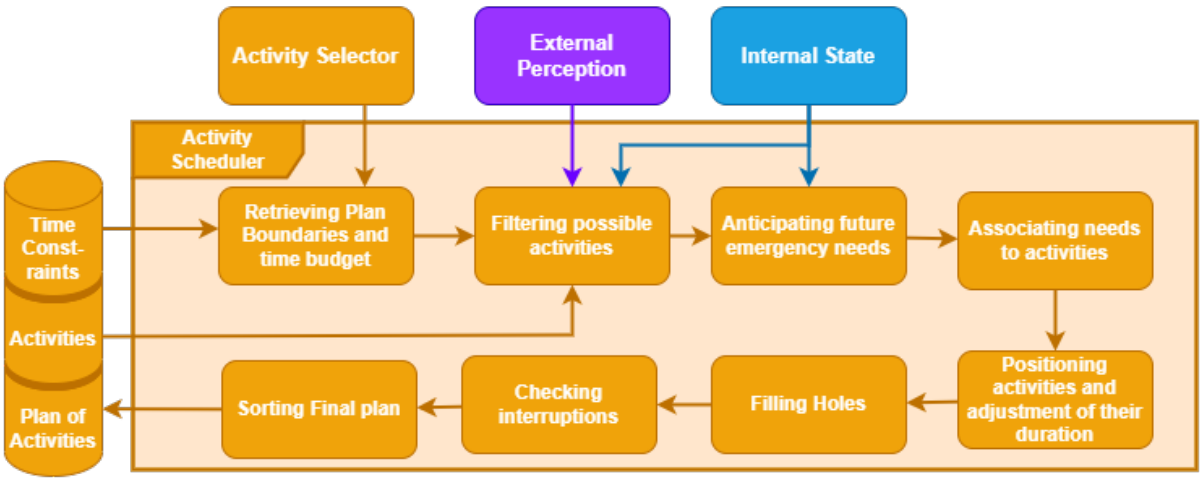

Figure 2.7 – Main Steps of the activity scheduler

Step 1. Retrieving Plan Boundaries and Time Budget: During this step, the agent evaluates the duration of its free time before the next mandatory activity, while considering the activity already performed. Concretely, the scheduler receives a request from the Activity Selector to start a new plan. The duration of the plan given by the Activity selector is converted into a time budget that must be respected to avoid impacts on the next mandatory activity. The activity already performed and the ones starting just after the plan are also stored to consider their effects. If the time budget is too short to put any activity, all the next steps until the checking interruption step is skipped. In the example given by figure 2.8, we have a time budget of 3 hours between the *Working* activity just performed and the next mandatory activity *Watching TV*. In the other example shown in Figure 2.9, the time budget is 20 min between both mandatory activities.

Step 2. Filtering possible activities: At this step, the agent identifies what activities are possible during its free time. Activities are excluded if they meet one of the following conditions: their minimum duration is greater than the remaining time budget, their daily maximum occurrence is exceeded, or they are not authorized to be performed during this time window. If no activity is found, all the next steps until the Filling Gap step are skipped, and default activities configurable by the user will be selected to fill the plan.

Step 3. Anticipating future urgent needs: In this step, the agent tries to anticipate

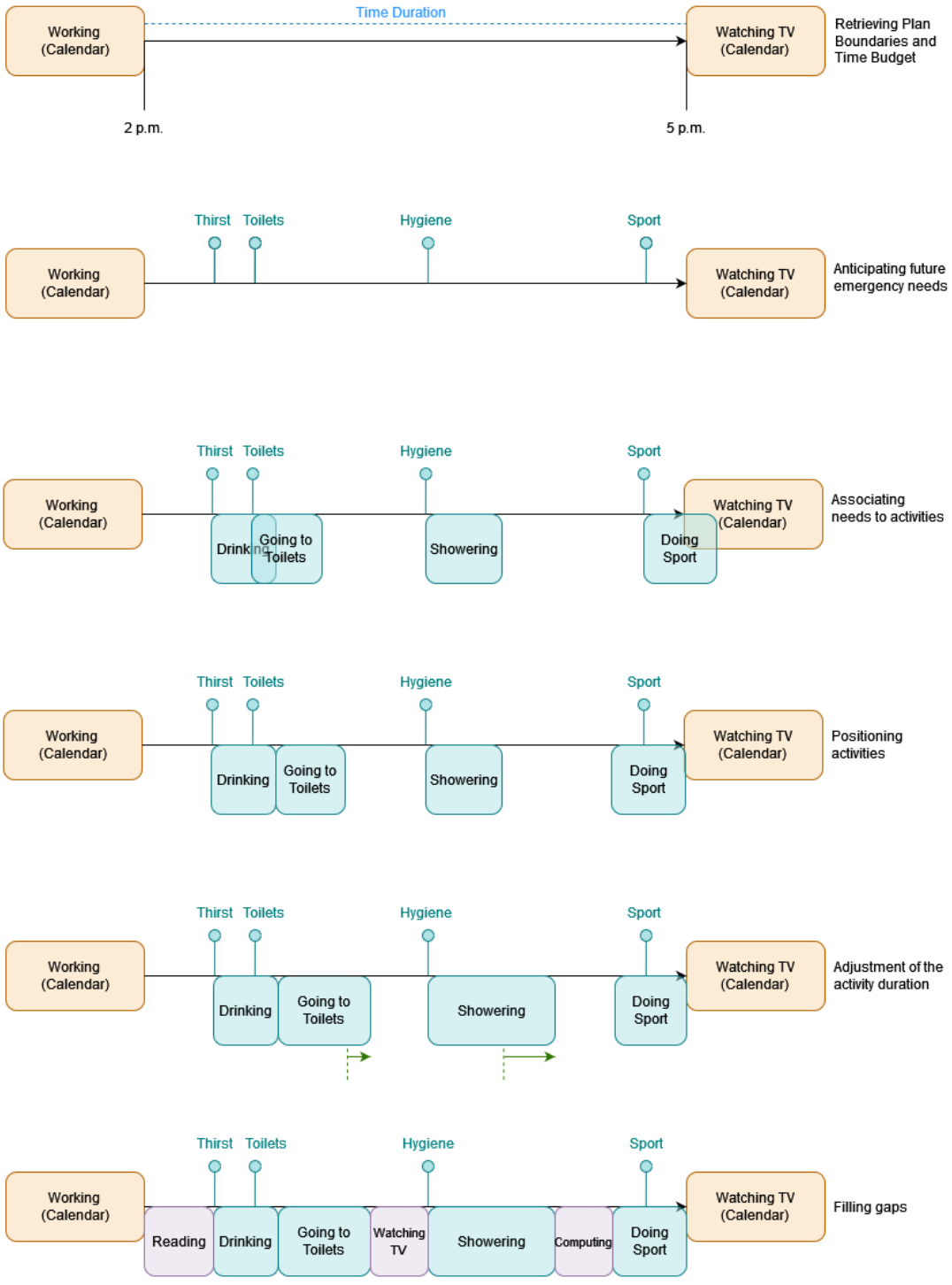

Figure 2.8 – Example of the different steps made by the Activity Scheduler

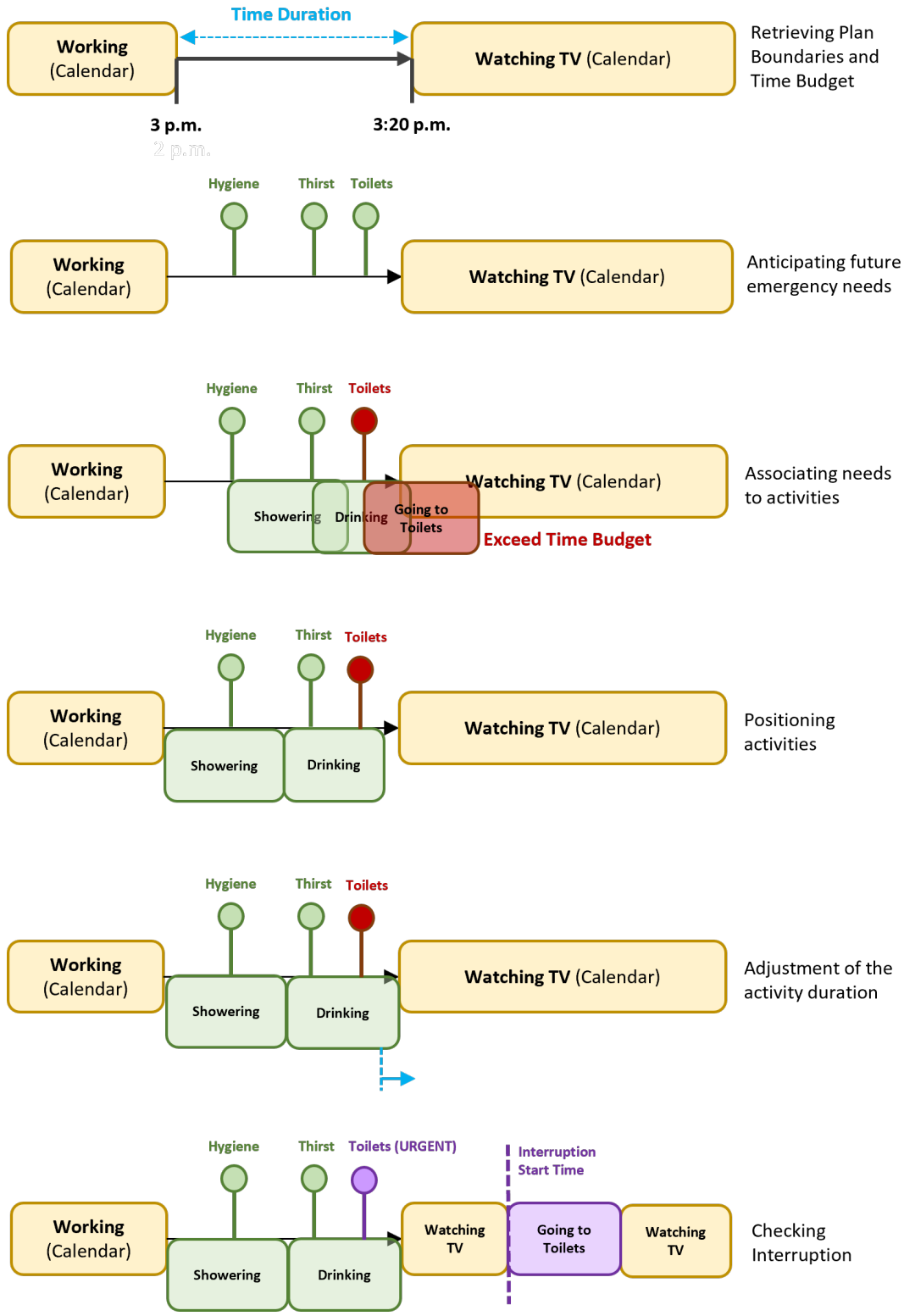

Figure 2.9 – Second example of the different steps made by the Activity Scheduler

its needs that will be urgent during its free time. As shown in Figure 2.5, the Scheduler requests the Internal State model to retrieve the list of needs that will be urgent during the plan time window. To do this, the Internal State calculates the urgency level of needs, evaluated at each time step by launching the process described in section 2.4. This time step has a 5 minutes default value but can be configured to obtain more or less accuracy. Through this process, The Internal State Model calculates an interval of urgency delimited by the time when the need starts to be urgent and the time when its urgency is maximum. We choose this calculation method since it is independent of the way used to establish needs urgency (time functions, interdependence functions, statistics, etc.). If the time window of the plan is large, the same need can become urgent several times. This is why, this process is repeated until all possible urgency intervals are found for each need. For instance, if the plan lasts 7 hours and that *Thirst* becomes urgent every 3 hours, then the Internal State model finds 2 intervals of urgency. All of them are then sent to the Activity Scheduler. After this, the Scheduler chooses a random time called t_{Need} inside each interval. This time is used as a reference point to place the activity satisfying the regarded need but also to obtain a better diversity: needs will never be satisfied at the same time, even though the simulation is repeated with exactly the same input parameters. In the first example given in Figure 2.8, emergency intervals were found for *Thirst*, *Toilets*, *Hygiene*, and *Sport* respectively. A t_{Need} is assigned to them to illustrate their emergency time. For the second example given in Figure 2.9, emergency intervals were found for *Hygiene*, *Thirst*, and *Toilets*.

Step 4. Associating needs to activities: When the agent has finished identifying its future needs, it will try to choose an activity satisfying them. For this, the scheduler tries to place an activity satisfying a need at each occurrence of t_{Need} that belongs to this need. The activity is placed with its minimum duration. Before the simulation, needs, and activities are linked together to know which activities can satisfy which needs: each need is imperatively linked to at least one activity. However, an activity is not necessarily linked to a need, such as the activities *Waiting* and *Getting some fresh air*. To choose which activity will be chosen in the plan, the process starts with the t_{Need} that is the closest to the starting point of the plan and continues to retrieve the next ones in chronological order. This operation also ensures that two activities do not start at the same time. If several activities are possible for the same need, a random selection is used where preference parameters can be associated to put weight on the choice. When an activity

is placed, the minimum duration of this activity is removed from the remaining time budget. This step ends when there is no more t_{Need} to associate with an activity, or when the time budget is reached. In this last case, all remaining t_{Need} having no yet associated activities are deleted. Finally, in the case where the mandatory activity satisfies a need, all the t_{Need} related to this same need and close to this mandatory one are deleted to avoid redundancy. If we look at Figures 2.8 and 2.9, Drinking was associated with *Thirst*, and its start time is placed at the related t_{Need}, with its minimum duration. The same process was made to associate *Toilet Need* with *Going to the Toilets* activity, *Hygiene Need* with *Showering* activity, and *Sports* Need with *Doing Sport* activity. However, in the second example, the time budget is exceeded for the last need related to *Toilets*. Consequently, the associated *Going to Toilets* activity is not added to the plan.

Step 5. Positioning activities and adjustment of their duration: In this step, the agent adjusts the duration of activities and their start time in order to be ready at the end of its free time. Concretely, The scheduler shifts the start time and end time of the activities so that all planned activities do not overlap and exceed the time window allocated to the plan. This process is done by trying to keep the activities as close as possible to their t_{Need}. To do this, starting with the first planned activity, the algorithm first moves the activities whose start times are located before the beginning of the plan or whose end times are placed after the start of the next activity. Then, starting this time with the last planned activity, the algorithm moves activities whose end time exceeds the end time of the plan or those whose start time is placed before the end of the previous activity. With this round trip, activities are sure to start and end within the plan without overlapping with other placed activities.

After this, starting with the first planned activity, the duration of each activity is extended either until its maximum duration or the start of the next activity if the maximum is larger. In the example of Figure 2.8, Drinking overlaps *Going to the Toilets* and *Doing Sports* finishes after the end of the plan. To avoid this, when start times are analyzed, the algorithm first shifts *Going To the Toilets* to start just after *Drinking* and does not move the others since their start times are well placed. Then, when end times are analyzed, *Doing Sport* is shifted to finish before the end of the plan, and the rest is not changed since their end times are well placed. After this, *Drinking* is not extended since it is stuck with *Going To Toilets*. However, *Going To the Toilets* and *Showering* are extended to their maximum duration. *Doing Sport* activity is not extended since it finishes at the end of

the plan. The same process is shown in Figure 2.9: *Drinking* starts earlier to avoid ending up after the plan and *Showering* starts also earlier to avoid overlapping with Drinking. After this, *Drinking* is slightly extended since few minutes left before the end of the plan. The rest is not extended because the time budget is yet reached.

Step 6. Filling gaps: If the agent still has free time after scheduling activities satisfying its needs, it will include default activities, such as entertainment activities, to not stay inactive. Concretely, at this step, empty slots can be still present where no activity is scheduled. Here, gaps are considered as moments when the agent has no constraints coming from the user or the Internal State Model. Therefore, during these free-time periods, the default activities which are configurable by the user, are chosen. For instance, in our experiments, we choose to use entertainment activities such as *Watching TV*, *Computing*, or *Reading*, but also *Waiting* in case of small gaps. Figure 2.8 shows that *Reading*, *Watching TV*, and *Computing* activities are placed to fill gaps between the activities satisfying needs. Since the time budget is exhausted in the second example, no default activities are added.

Step 7. Checking Interruptions: Sometimes the agent does not have enough time to satisfy all its urgent needs. To solve this, the user can indicate which activities can be interrupted and which needs can interrupt activity in case of emergency. During this step, the scheduler first checks if the activity starting just after the plan is interruptible (this one can be the next mandatory activity or the last scheduled one when no mandatory activity is indicated). It then verifies if a need that may interrupt and already urgent before the plan has no moments of satisfaction in the plan. If these conditions are reached, the scheduler triggers an interruption process and chooses a possible activity satisfying this need. At this stage, we check if the interrupted activity has a sufficient duration to support a reduction equal to the duration of the interruption, without falling below its minimum duration. In this case, a random moment of interruption is retrieved between the start and the end of the activity. Depending on this moment value, the interruption can start before, after, or during the interrupted activity, without exceeding its initial duration. In our second example shown in Figure 2.9, *Going to Toilets* activity could not be added to the plan to satisfy the *Toilet* need since the time budget was not sufficient. Therefore, as *Watching TV* can be interrupted, the scheduler launches an interruption so that this activity can be inserted. *Watching TV*, having a sufficient duration, is then cut

in half and shortened to insert *Going to Toilets*.

When these steps are finished, the plan is delivered to the Activity Selector in order to retrieve the activities to execute. The Activity Scheduler will then wait until a message coming from the Activity Selector is sent to restart a plan in a defined time window.

2.6 Task Executor Model

The Task Execution Model is used to execute the activity given by the Activity Selector in the Virtual Environment (VE). To do this, it launches sequences of predefined tasks containing animations and moves related to this activity. The state of the activity is also returned to the Activity Selector. The External Perception Model is also used to retrieve the information about objects and places coming from #FIVE for each task to correctly perform this activity. With this, we can obtain a dynamic adaptation to environments since in global cases, the user only defines an object type or feature and the agent selects the one having this type or feature to perform an activity. For instance, to sit down, the agent could choose a *sittable* object such as a chair or a sofa, regardless of the name they can have in different environments. In addition, a specific object can also be mentioned by the user for situations requiring it. For instance, to use the laptop located on the desk, the agent has to sit down on the desk chair and not on another *sittable* object.

To concretely execute activities in the VE, each task defined in the Task Execution Model is linked to an action sequence based on Petri-Nets [140]. These action sequences are designed with the help of the #SEVEN model [49]. This scenario-based model gives a graphical representation of the Petri-Nets and each transition, which can contain checkpoints or effectors able to trigger interactions, can be configured by the user and linked to semantics coming from the #FIVE model. Execution of a task sequence can thus vary depending on context. The location of objects and appliances, as well as the current location of the agent, are automatically considered. In addition, the duration of activities is updated each time they are launched, by retrieving information coming from the Activity Selector. The object implied in the activities can also be changed, chosen randomly, or chosen with specific conditions (the closest object, the preferred one, etc.) in the input parameters of the tasks as long as they can interact in the same way with the agent. For example, the agent can randomly sit on a chair or on a sofa, can prefer to eat a

pizza rather than a sandwich, can choose between going to the toilets situated on the first floor or the one located on the ground floor according to its position, and so on. If the agent has different ways of performing the same activity (such as cooking a pizza or pasta), the choice can be made either using randomness or with another condition (such as preferences). If the agent can perform an activity in one of two rooms, then the Task Execution model can be configured to choose the closest room. When the execution of the task sequence is finished, the given activity is considered done and a signal is sent to the Activity Selector, which then selects the next one to perform.

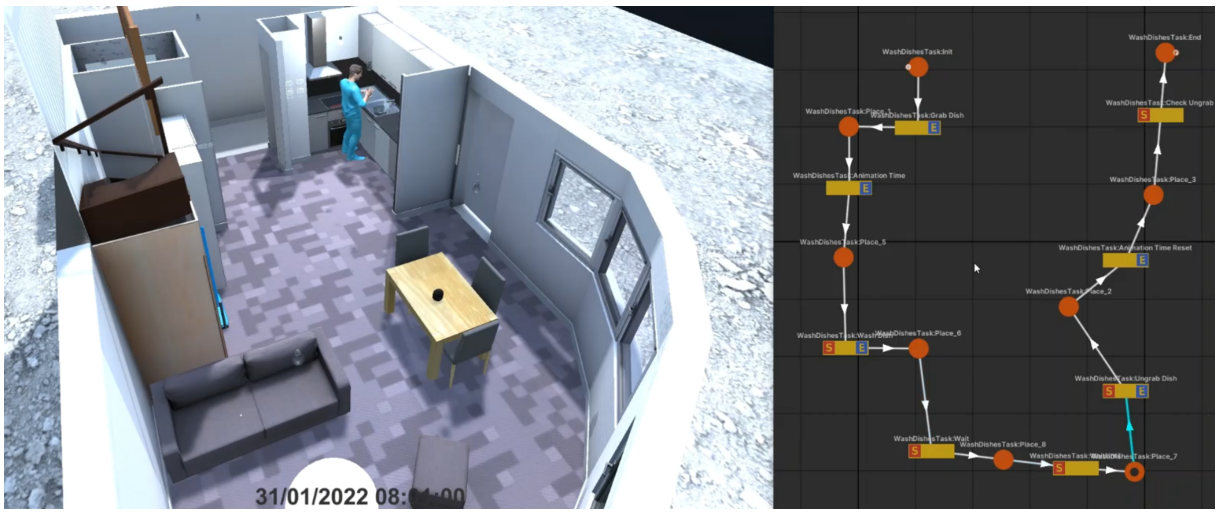

Figure 2.10 – Example of the Washing Dishes activity performed with the #SEVEN model [49] implemented with *Xareus Software* [3] (located on the right)

This approach based on Petri-nets has the advantage of being able to create a sequence of tasks and actions in a simple, fast, and accurate way. Moreover, contrary to methods that automatically build the sequence of tasks to be performed like STRIPS [70], this kind of predefined approach allows the user to better control the content of the task sequence. Moreover, it brings more diversity in activities since the user can put different objects, implement randomized choices between several possibilities or place different paths according to the situation or the agent preferences. Effectively, STRIPS-based methods always choose the most efficient way, thus limiting the diversity of results. Nevertheless, this kind of approach can still be used as an alternative to Petri-nets.

2.7 Functional Validation

After explaining the content of our agent model, we now focus on experimentation enabling the functional validation of our model. This validation uses an input-output comparison method to know whether the produced behaviors are diversified, consistent facing the same situation, and coherent in comparison to the input parameters. Since diversity, coherence, and consistency are essential parameters for assessing credibility [11], it allows us to make a first validation of the credibility of agent behaviors. As detailed in Chapter 1, the Input-Output comparison method is essential to evaluate the reliability of the generated behaviors in comparison with input parameters. This approach is used by several existing approaches such as those proposed by Trescak et al. [170], De Sevin et al. [159], or Charypar et al. [44].

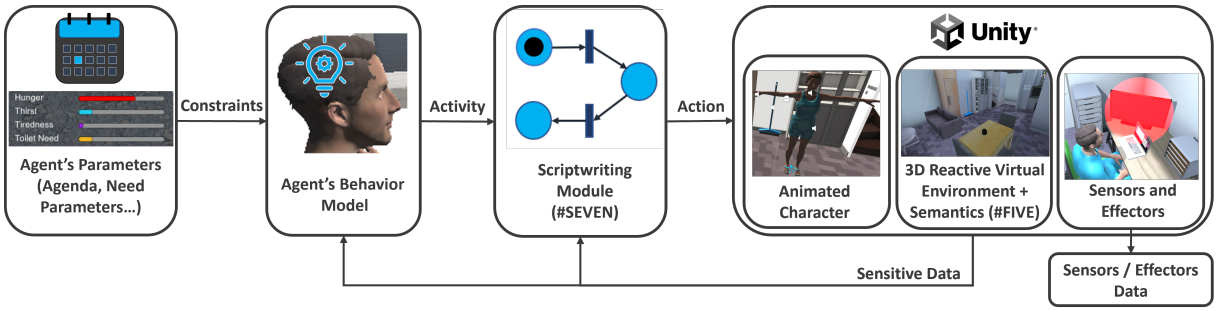

Figure 2.11 – Diagram of our Global Implementation

In our case, by comparing the input and output of our model, we check if the agent model is coherent with the input parameters and consistent faced with a similar situation. Concretely, regarding the coherence of behaviors, we evaluate if all the mandatory activities coming from the user's schedule are performed on time, whether the schedule is strict or moderate. We also check if calendar activities can be interrupted when a need allowed to interrupt is too urgent. Finally, we assess if all needs are satisfied within the expected time frame during the free-time period (average satisfaction in the emergency range of needs). Concerning the diversity and consistency of behaviors, we evaluate if, for the same input constraints, our agent model is able to produce different output activities and different moments of need satisfaction.

Table 2.1 – Activities and Needs implemented in our experimentation

Need Name	Maslow's Hierarchy (1 to 5)	Related Activities
Hunger	1	Eating
Thirst	1	Drinking
Tiredness	1	Napping, Sleeping
Hygiene	1	Using the Sink, Showering
Toilets	1	Going to the Toilets
House Hygiene	2	Cleaning
Sport	4	Doing Sport
Hobbies	5	Watching TV, Reading, Computing, Using the Phone (also used as default activities)
None	No	Getting Some Fresh Air, Preparing, Cooking, Dressing, Entering, Leaving, Working Washing Dishes, Waiting (default)

2.7.1 Global Implementation

Figure 2.11 shows the global structure of our implementation used in our experiments. First of all, the user can give initial parameters to configure the agents (needs values, preferences, available tasks, etc.). Among them, a calendar including all mandatory activities can also be given. It enables the insertion of mandatory activities to allow the user to simulate activities at specific times or to respect the same protocol of the real databases the user wishes to enrich or replicate. This is particularly our case since we want to reproduce the real Orange4Home database for validation purposes (see Chapter 4). The decision-making process is then launched and chosen activities are then executed with a #SEVEN scenario. This scenario is then sent to the 3D animated agent to perform activities in the 3D environment.

Regarding the configuration of the parameters that have been used, the preferences have not been included here to know the *generic functioning* of our model and its credibility. More than 20 daily activities were implemented and animated as well as 8 needs to represent the agent's internal states. They are summarized in Table 2.1. All the simulations last 10 simulated days. For these simulations, needs and activities were configured

as detailed in Tables 2.2 and 2.3. A video of our use case can be viewed at the following link: `https://youtu.be/v8GxXCAAV1k`. The numbers of activities and needs are for illustrative purposes: the user can add activities as many as necessary without impacting the model's working. The resources of the world are considered here as always available throughout the process.

2.7.2 Functional Validation with Full-Autonomy

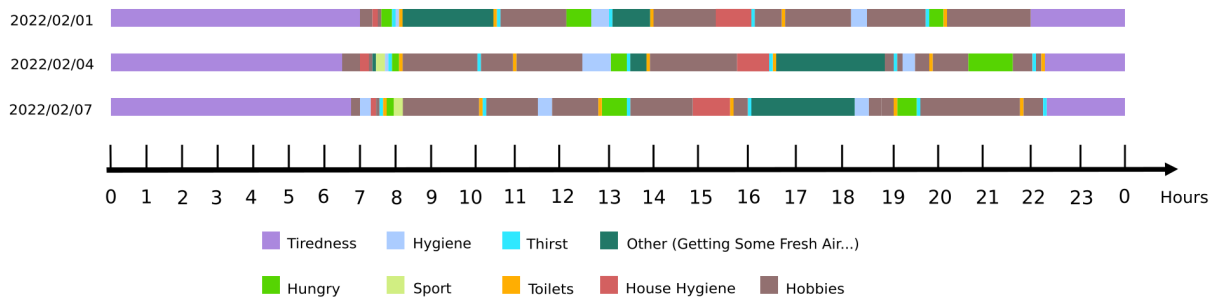

Figure 2.12 – Timeline of three-day samples showing all activities satisfying needs during the daytime. The activities satisfying the same need are put together. For instance, Hobbies contains Reading, Watching TV, Computing, and so on.

In this section, we analyze the reaction of our model when no input activity calendar is given by the user. This allows us to know how the agent is autonomous throughout the different days simulated continuously. This approach also enables us to validate the coherence of behaviors with respect to the input parameters of the needs. It also allows us to analyze the routines to see if they are diversified while being coherent with input parameters. Validation is done first on a specific day, then over the long term by analyzing several days to obtain quantitative results.

Tables 2.2 and 2.3 summarize information about the satisfaction of needs according to the initial parameters. We can see that needs are satisfied when they are urgent since the average start time of activities satisfying needs is inside the bracket between the time when needs start to be urgent and the time when their maximum urgency is reached. Better accuracy is also observed when specific hours are used rather than periods to configure the needs. Moreover, activities satisfying needs are launched each time the related need is urgent during the day. Effectively, few differences between theoretical and simulated frequencies are observed: their gap is inferior to 0.05%. The minimum and maximum

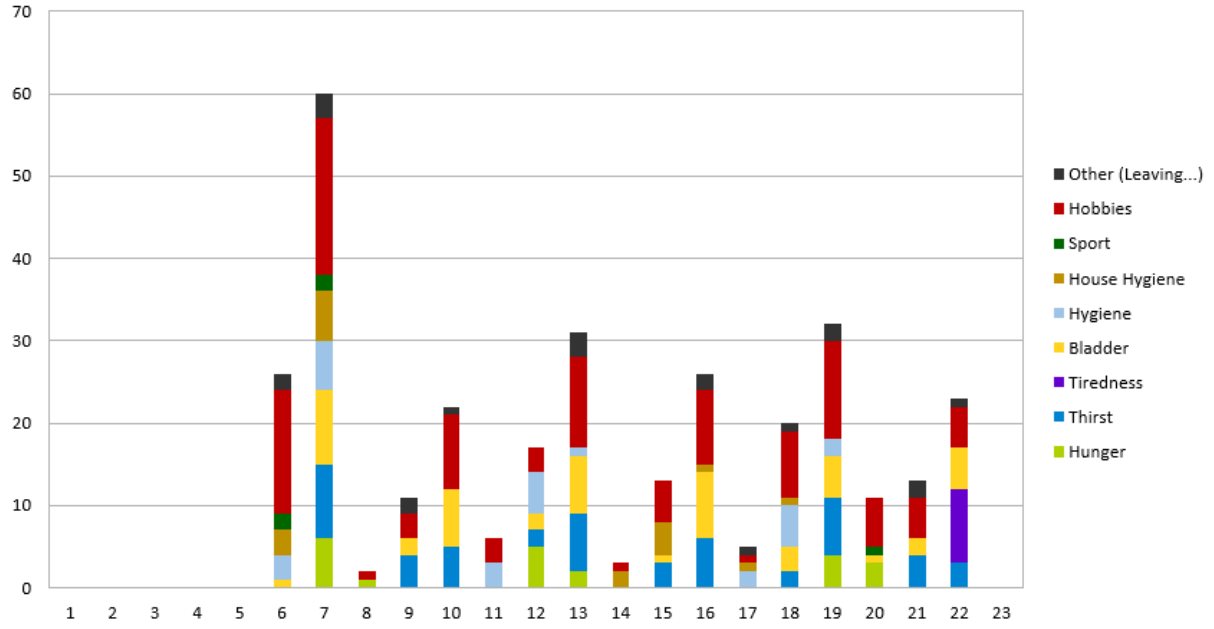

Figure 2.13 – Frequency of start time activities satisfying needs during 10 Days according to the hour of day

durations of activity satisfying needs are also strictly respected since not exceeded. The sleeping phase (between 12 a.m. and 6 a.m.) has been excluded from the calculations made in Table 2.3, since no needs can be satisfied during this period. These tables show us that behaviors produced are coherent with the input parameters since activities satisfying needs start inside the period of urgency and with respect to their minimal and maximal duration. They are also launched for all the time slots when the related need becomes urgent. In addition, the generated routines are diversified since activities do not start at the same time from one day to another day whereas the parameters of needs are the same for all days: the difference in start times can vary from a few minutes to over an hour.

Variations in duration are also observed where the gap can reach 30 minutes while respecting the maximum and minimum durations of activities. However, an exception is noted with *Sleeping*, where the duration remains at its minimum every day, reflecting a limitation of our model: activities that are really long and uninterruptible lead to an accumulation of urgent needs after them, which makes it difficult to extend their duration beyond their minimum. This same limitation is also seen in Figure 2.13 where a stronger accumulation of different activities is observed between 6 a.m. and 8 a.m. to satisfy all the needs that could not be satisfied during sleep hours. To address this limitation,

a two-speed evolution of needs could be proposed: one for the day and another for the night, in order to delay the urgency of needs during the night. Another solution could be to develop the impact of *Sleeping* and *Napping* on other needs: When executed, they could slow down the evolution of other needs, so they will not all be urgent as soon as the agent wakes up.

Regarding the timeline shown in figure 2.12, we can observe that the periodic satisfaction of needs creates a routine having some variations due to the variety of possible activities and the number of occurrences per day. For instance, *Eating* is satisfied three times per day around the same periods. The variations in the choice of activities and in their start time are interesting to produce more credible behaviors since diversity is provided while respecting coherence about input parameters. Figure 2.13 shows the start times of activities performed by the agent during 10 days according to the time of day. Judging from this figure, a routine seems to appear, since the agent eats and sleeps at regular times. All these results are encouraging in the sense that our model could create credible behavior when the agent is totally autonomous. Effectively, behaviors are coherent with the input parameters and diversified. They are also consistent: when a need is urgent, the chosen activity is always the one related to its satisfaction.

Table 2.2 – Example of needs configured with specific hours and their associated activities compared to their input configuration

Need name	Related activities	Theoretic time slot	Min/Max start time	Mean and standard deviation start time	Mean Frequency per day (theoretic)	Mean Frequency per day (simulated)	Min/Max Duration (theoretic)	Min/Max Duration and Mean Duration with standard deviation (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. ± 0h11m 12.36 p.m.± 0h27m 7.53 p.m. ± 0h37m	3	3.0	0h10m/1h	0h10m/1h 0h28m ± 0h21m
Tiredness	Sleeping	[10 p.m., 12 a.m.]	10.08 p.m./ 10.40 p.m.	10.24 p.m. ± 0h12m	1	1.0	8h/11h	8h/8h 8h ± 0m

2.7.3 Functional Validation with an Input Calendar of Activities given by the User

In this section, we analyze the reaction of our model when an input activity calendar is given by the user. This allows us to know how the agent can manage autonomy while respecting strong constraints throughout the different days simulated continuously. This approach also enables us to assess the coherence of behaviors in comparison with the

Table 2.3 – Example of needs configured with specific periods and their associated activities compared to their input configuration

Need name	Related activities	Theoretic periods	Min/Max gap between 2 satisfactions (between 6 a.m. and 12 a.m.)	Mean and standard deviation between 2 satisfactions	Mean Frequency per day between 6 a.m. and 12 a.m. (theoretic)	Mean Frequency per day between 6 a.m. and 12 a.m. (simulated)	Min/Max Duration (theoretic)	Min/Max Duration and Mean Duration with Standard Deviation (simulated)
Thirst	Drinking	Every 3 hours (urgency 2h05m)	2h13m/3h41m	2h57m ± 0h26m	6	5.75	0h01m/0h05m	0h01m/0h05m 0h04m ± 0h01m
Toilets	Going to the Toilets	Every 3 hours (urgency 2h05m)	1h06m/3h36m	2h53m ± 0h24m	6	5.88	0h05m/0h10m	0h05m/0h10m 0h08m ± 0h02m
Hygiene	Showering Using the Sink	Every 6 hours (urgency 4h02m)	4h25m/6h40m	5h29m ± 0h45m	3	3.0	0h15m/1h 0h05m/0h20m	0h15m/0h35m 0h21m ± 0h08m 0h05m/0h20m 0h14m ± 0h06m
House Hygiene	Cleaning	Every 10 hours (urgency 7h04m)	7h40m/11h26m	8h56m ± 1h13m	2	2.0	0h10m/2h	0h10m/1h50m 0h30m ± 0h28m
Sport	Doing Sport	Every 48 hours (urgency 33h56m)	34h57m/37h22m	36h10m ± 1h12m	0.5	0.50	0h15m/2h	0h15m/0h38m 0h19m ± 0h09m

user's input parameters. It also allows us to analyze the routines to check if they are diversified and coherent with input parameters during free time periods.

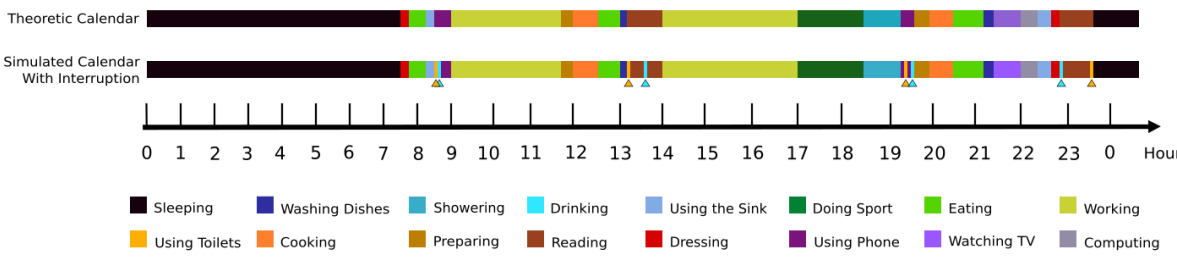

Figure 2.14 – Comparison between theoretic and simulated timelines for the same day with a strict calendar

The first result presents the case where the user provides a calendar with no free time. To illustrate it, Figure 2.14 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 *Toilet* needs have been authorized to interrupt the calendar. Some calendar activities such as *Using the Phone* or *Reading* are interruptible. These results show that mandatory activities are performed on time and with the right duration: Among activities that were not interrupted, the observed maximum deviation was for *Doing Sport* with a one-minute deviation from the imposed activity. Interruptions are indicated by blue triangles for *thirst* and yellow triangles for *toilets*. Thus, the agent is able to interrupt mandatory activities to satisfy its needs allowed to interrupt. This kind of activity interruption due to an urgent need enables the introduction of diversity when

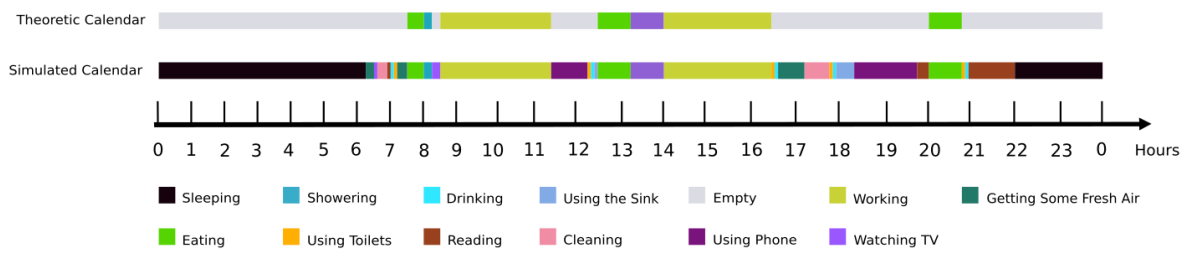

Figure 2.15 – Comparison between theoretic and simulated timelines for the same day with a moderate calendar

a schedule that does not offer free time is given. Since needs do not always interrupt at the same time, a large number of schedules can be created even with really strict activity schedules. Moreover, despite the interruption, all schedules given by the user are respected. This proves that our model is coherent with the user's requirements while offering diversity. The activity selection is also consistent when the same urgent need happens since the same set of activities is selected to satisfy it. If the agent cannot interrupt any activity to satisfy its needs, then the urgency gauges of the needs indicated in the interface will reach their maximum value, as shown in Figure 2.16. The user is then aware of the agent's incapacity to satisfy its needs with the given strong constraints.

Figure 2.16 – Interface illustrating the urgency of needs with gauges and their evolution through time. When the need starts to be urgent, a red color is indicated.

Figure 2.15 compares timelines between a day coming from a theoretic calendar containing free time with the results of the same-day simulation. In addition, Figure 2.17 shows a comparison between a user's input activity calendar with the output one performed by the agent. To handle such moderate calendars, the agent must rigorously manage its free time to be ready for the next required activity. As seen in both figures, the agent accurately schedules its activities during its free time while considering the satisfaction of its needs. The difference between the simulated and the theoretical calen-

dars is the same order as the strict calendar, proving that the agent respects the strong constraints. Figure 2.17 also shows that some mandatory activities are interrupted to manage needs that could not be satisfied before, such as *Thirst* or *Toilet* ones.

When mandatory activities are scheduled, they can sometimes satisfy needs. Redundancies may then appear between the mandatory activities and those chosen by the scheduler: for example if *Hunger* becomes urgent at the same time when *Eating* is scheduled by the user, the *Eating* activity may appear twice in a row. Consequently, as explained in Section 2.5, we make sure that the scheduler considers the effects of the next mandatory activity by removing the impacted needs that are too close. The obtained results confirm that no redundancy is observed when the calendar activities satisfying needs are positioned close to the times when these needs are urgent.

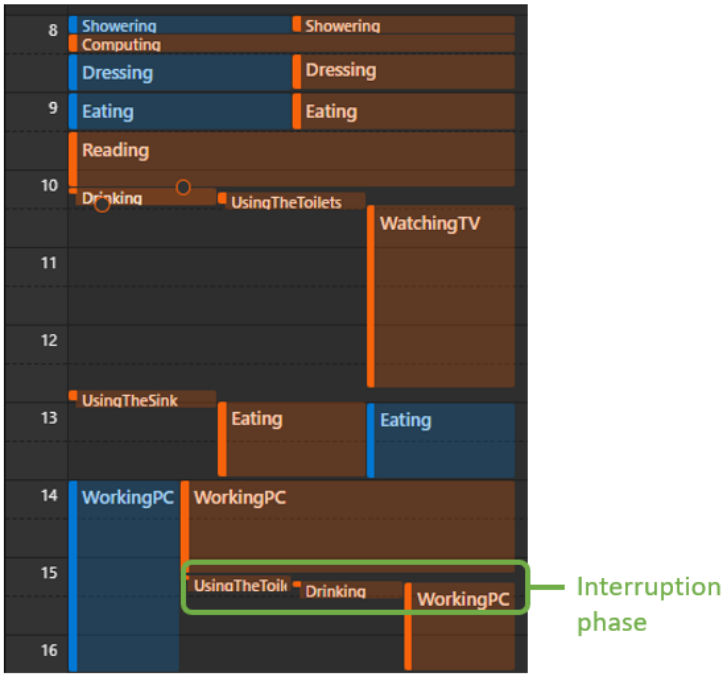

Figure 2.17 – Comparison between the user's activity calendar in blue and the activity performed by the agent in Orange

The results presented in this section show that our agent can perform activities in the VE while respecting its input constraints and its urgent needs. In this way, our model is able to adjust the level of autonomy according to the user's configurations and the agent's internal state, providing a compromise between control and autonomy.

This Input-Output comparison method allowed us to know whether our system is

coherent with the input parameters while producing diversity. We also evaluated if the system is consistent in activity selection to satisfy needs (e.g. when *Hunger* is urgent, *Eating* will be selected and not *Doing sports*). The results obtained in this section are promising and provide us a first validation of our model robustness in facing diverse situations where time and need constraints are more or less significant. Our results also show that our agent can be fully controllable, partially controllable, or fully autonomous without modifying its structure. Concerning the credibility evaluation, the results are promising since behaviors are coherent with the input parameters, diversified in most cases, and consistent for the same need satisfaction. However, as we said in Chapter 1, this comparison method is essential but not sufficient to evaluate the credibility of behaviors.

2.8 Conclusion

Our agent model was designed to address our challenges and some shortcomings of the state-of-the-art approaches related to the generation of daily activity data. In addition, this agent model version is yet sufficient to address a part of our use case: users can exploit our model to simulate situations including one occupant and to generate, replicate, or complete databases containing information about these situations. This is the case for the Orange4Home real database proposed by Cumin et al. [51] for instance which can be replicated or completed by synthetic data generated by our model. Effectively, our agent model offers a compromise between control and autonomy since the agent can strictly respect the users' constraints (such as an input activity calendar to perform) while staying autonomous to satisfy its proper objectives during free time. This controllability aspect enables the accurate replication of real datasets since we can closely follow given activity schedules that were used for real dataset collection protocols. The autonomy aspects are also essential to produce credible synthetic data since a lack of autonomy would generate rigid data with little variability, losing credibility. By balancing these two aspects, the agent can thus anticipate future constraints and adapt its free time to satisfy its proper needs while preparing itself for the following imposed activity. In addition, our agent can relax the users' constraints to satisfy urgent needs by interrupting imposed activities when permitted by users. This enables us to offer more diversity in the behaviors even in cases where too many constraints prevent the agent from satisfying its own needs. Finally, our model is compatible with data generation since the agent can execute activities in a

3D environment and can thus trigger the sensors and effectors to produce synthetic data. **With our model, the agent is thus compatible with data generation and can be fully autonomous during the simulation, fully controlled based on a schedule, or any mix of both.**

A first validation based on Input-Output comparison was also presented in this Chapter to evaluate the reliability of our agent model and the credibility of behaviors. However, as we said in Chapter 1, this kind of method is essential but not sufficient to prove credibility and fully address Challenge 3. This is why, another *a posteriori* validation approach is presented in Chapter 4 to reinforce our results and evaluate whether our agent can produce credible behaviors while respecting strong constraints.

The contents of this Chapter have been published at two international conferences. The first one, concerning the Internal State Model, was published in the MARCH workshop organized by the IEEE AIVR conference in 2021 [81]. The second one, concerning the whole agent model with the presented results, was published at the PAAMS conference in 2022 [80].

As explained before, this agent model version is yet sufficient to address a part of our use case for the reasons stated above. To go further, we propose in Chapter 3 additional models improving the adaptation to Dynamic Environments. This improvement allows users to simulate new situations including several occupants or unexpected changes coming from the environment. Users could also generate, replicate, or generate data related to these new situations. Effectively, even though our scheduler is compatible with such environments since rescheduling processes can be launched when required, some limitations can be observed regarding the management of resources and unforeseen events coming from the environments (such as managing telephone calls). These processes are essential to simulate several agents in the same environment where resources are limited and unexpected events can be triggered by other agents.

Chapter 3

Additional Model to better address dynamic environments

In the previous chapter, we explained the global structure of our agent model as well as how this model can both address our challenges and a part of our use case. In this chapter, we focus on the additional models to better address dynamic environments, since Chapter 2 was focused on managing time and need constraints, without insisting on adaptations to changes coming from the environment (resources, events from the VE, other agents). Although the scheduler is compatible with dynamic environments since it can reschedule when needed, it has limitations regarding the management of resources and unexpected events happening suddenly and coming from the environment (telephone calls, the sudden disappearance of a resource, requests from another agent, etc.). In addition, this model version cannot suddenly interrupt activities in progress if unforeseen events happen or if resources are missing. Since the virtual environment proposed in Chapter 2 to test our agent model is static (because the agent is the only one to act on it, and resources are not exhaustible), the interruption of ongoing activity caused by the environment does not occur. This was not an issue if users wants to exploit our model to simulate one-person situations and create, generate, or replicate databases related to these situations. In addition, for the part of our use case including the replication of the Orange4Home real dataset explained in Chapter 4, using this model version was sufficient since one person was considered in the real database and no unexpected changes coming from the environment were included. Consequently, issues related to dynamic environments were not fully considered previously. For all these reasons, the credibility of behaviors and scenarios related to dynamic environments remains limited to fully addressing Challenge 1. Yet, if users want to simulate several agents, trigger unexpected situations when they desire, and produce, replicate or complete multi-person databases, an agent model addressing dynamic environments is required.

According to the conclusion of Chapter 1, existing approaches rarely propose a model

simultaneously managing resources, time, unexpected events coming from the environment, agent internal motivations, and activity execution. In addition, most of them do not manage resources and unexpected events, both at the execution level (interrupting animations, resuming interrupted activities, etc.) and scheduling level (adapting the plan to disruptions, anticipating future missing resources while respecting time constraints, etc.). Moreover, most existing agent-based approaches do not provide any process to interrupt and resume ongoing activities. Concretely, current solutions rarely propose to manage the case where the agent is interrupted in a state where it could not immediately perform any other activity. For example, if the agent is interrupted while reading on a sofa, it cannot perform any other activities until it has closed the book, put it down, and stood up. In addition, solutions are also rarely proposed to return the agent to a state where it can continue the interrupted activity. For example, if the agent wants to go back to reading, the agent has to pick up the book, sit down again and open it. Finally, the control that users can have over the agents in dynamic environments is often too limited.

This lack of complete solutions in the state of the art led us to propose our solution to address dynamic environments by handling resources and unexpected events at the execution and scheduling levels while respecting other constraints such as time and needs. For this, we propose a new model designed to handle dynamic environments and made of three sub-models: **The Resource Manager** to anticipate resources during the plan creation, **the Interruption Manager** to interrupt the executed activity when a missing resource or an event suddenly happens, and **the Plan Checker** to verify if interruptions caused by missing resources or unexpected events impact the rest of the plan. This models is independent from other processes present in our agent model, and can thus be disabled or integrated into global agent structures such as BDI [162]. More concretely, we explain more precisely in section 3.1 why considering dynamic environments is relevant for our use case and to better address Challenge 1. We also develop in this section the specifications related to these environments. After this, we propose in section 3.2 an overview of the three sub-models managing resources and events both at the scheduling and execution levels. We then detail these models by starting with the Resource Manager in section 3.3. We then explain the Interruption Manager in section 3.4. After this, we study the Plan Checker in section 3.5. Finally, we propose in section 3.6 a user-experiment protocol to validate the credibility of agent behaviors in the case of dynamic environments.

3.1 Motivations and Specifications to address Dynamic Environments

In this section, we explain why managing dynamic environments is relevant to our case and what are the required specifications to well address them.

3.1.1 Motivations to address Dynamic Environments

As explained in Introduction, our aim is to propose virtual humans able to generate credible behaviors while being compatible with data generation. We also introduced our main use case aiming to offer an agent model for users wanting to simulate virtual humans in daily-living situations and generate exploitable synthetic data related to these situations. In Chapter 2, we proposed an agent model specially designed to address our main issue and some parts of our use case. We thus detail in this section why considering dynamic environment may be relevant for our use case.

3.1.1.1 Providing diversity on the synthetic data and simulated situations

The first reason to manage dynamic environments is the possibility to generate behaviors and synthetic data that are more diversified. Effectively, even when the agent configuration will be the same, each simulation will lead to more different results since unexpected events affecting the agent and not caused by its actions may occur at any time. This diversity improves the credibility of behaviors and allows us to better address Challenge 1. The diversity of synthetic data is an essential feature to limit some bias caused by insufficient data, improve the credibility of the results, and allow the consideration of more different situations.

Finally, by proposing tools to configure resources and unexpected events, users exploiting our agent model will be able to simulate new situations that were not possible before, such as simulating a device failure situation (TV that does not work anymore, the fire alarm that goes off, and so on) or a situation where a lack of resources prevents the agent from performing the desired activities (there is no food anymore, the bathroom is blocked by another agent for 2 hours, and so on).

3.1.1.2 Offering Multi-agent simulations

Managing dynamic environments is also essential for multi-agent simulations where several agents share the same environment. In this situation, their actions can impact others since they can deplete or block some required resources, or trigger events such as soliciting others. They could also synchronize to perform activities together. The capacity to manage simultaneously several agents would enable users to produce multi-person data that can be really interesting to simulate workers in a building, a family in a house, and so on. In addition, since real multi-person datasets are less frequent than real mono-person ones, the simulation of multi-agents could allow researchers to generate new datasets and improve their model for multi-occupant understanding.

The credibility of such data mainly depends on the level of interaction between them. The more precise and diversified these interactions are, the closer they will be to reality. Judging from the literature about multi-agent systems, and more particularly the survey of Risk et al. [150], several levels of interaction between agents may exist:

- **Competitive (Negative Interaction)**: This interaction happens when agents have conflicting goals. The agents could thus deliberately disturb others to achieve their goals.

- **Conflict (Negative Interaction)**: This interaction happens when there are not sufficient resources to reach all the goals. The agent can thus enter into conflict to retrieve these limited resources.

- **Coordinative (Positive Interaction)**: This interaction happens when agents complete their individual goals and work together to minimize interference. This interaction thus allows several agents to coexist at the same time. In this situation, agents mainly exchange critical information to limit the impact on other agents, such as indicating that the bathroom is occupied. However, agents are not going to help others to achieve their personal goals. Such interaction can be found in MASSHA [97] where agents can live together in the same 2D house and perform their activities separately. We also have the work of Jorgersen et al. [95], where agents can move in a city to perform their daily activities by avoiding collisions with others. However, no collaborative activities are performed with such interactions.

- **Collective (Positive Interaction)**: This interaction happens when agents share common goals (i.e. each agent contributes to this goal). However, the agent is unaware of other agents and works on the part of this goal individually.

- **Cooperative (Positive Interaction)**: This interaction happens when agents share common goals (i.e. each agents contribute to this goal). The agent is aware of other agents and communicates with them to advance these goals.

- **Collaborative (Positive Interaction)**: This interaction happens when agents do not share common goals. They help each other to fulfill their individual ones. In this situation, agents can perform collaborative activities such as helping another agent in their work or agreeing to eat together despite different schedules. Several existing approaches use collaboration in the literature for daily-activity simulations since they are the most well-suited to give credible behaviors in this case. This is the case for several reactive-based approaches such as the work of Kashif et al. [99] where agents can communicate by chat and ask for collaborative activities such as eating together, those of Schumann et al. [156] where incitement messages are sent to make collaborative activities, or those of Bogdanovych et al. [24], [170], where the agents having roles in the Mesopotamian community can exchange resources (such as pottery for fish). In the case of Planning-Based approaches, we can find the MASCARET multi-agent model proposed by Querrec et al. [145] where simulated firemen have roles to make collaborative tasks, but also GOAP tested by Bogdanovych et al. [24]. These are also some Probabilistic-based approaches BIM SIM 3D [181], where a probabilistic scheduler can synchronize collaborative activities in the plans of agents during the period when this activity is likely to occur. We also have some Learning-Based approaches such as Double DQN [93] where agents satisfy their social needs by discussing with other agents. Regarding Scheduling-based approaches, we can find the work of Meister et al. [127] where joint activities can be scheduled by a Genetic Algorithm.

To synchronize agents, decision-making can be either centralized or decentralized [150]. In a centralized decision, agents can receive instructions from a leader having a global view of the goals to reach. This leader chooses and dispatches the tasks among the agents to achieve these goals. A dashboard can also be used in a centralized approach. In this case,

the agents choose which activity displayed on the dashboard they take according to their own availability. Such centralized approaches are used for instance in Probabilistic-based approaches such as BIM Sim 3D [181] or MASSHA [97], but also in scheduling-based approaches such as the work of Meister et al. [127]. In a decentralized approach, each agent performs tasks in parallel. If they need another agent to complete their task, they communicate either through an intermediary agent knowing the status and location of others or by talking directly to the concerned agent. Such decentralized approaches are mainly used by reactive-based approaches such as Kashif et al. [99] and learning-based approaches such as Jang et al. [93] since agents can be interrupted more easily than other approaches.

In the context of daily-living simulations, using a decentralized approach is most accurate to simulate a family, workers, or citizens. In addition, enabling collaborative, cooperative, conflict, and coordinative interactions between agents are the most relevant to simulate daily-living situations for several reasons. Firstly, in contrast to collective interactions, agents are aware of other agents since their share the same environment. Secondly, daily-living situations does not induce competition between agents since agents will not deliberately try to prevent others from achieving their own objectives. In our case, the final objective would to propose an agent model able to manage these four kinds of interactions. Effectively, existing approaches cannot really all these interactions in the same simulation. More concretely existing approaches simulating collaborative interactions have simplified some aspects related to dynamic environments to facilitate the integration. For instance, some of them do not consider limiting resources such as the amount of food left in the kitchen, or do not include unexpected events (in this case, the collaboration between agents is scheduled from the beginning and thus not triggered by sudden events). Consequently, coordinative and conflicting interactions are difficult to manage since agents will not be able to adapt their actions to avoid blocking situations. **In our case, with the management of unexpected events and resources, we can simulate agents having coordinative or conflicting interactions.** The two other interactions will be considered in our future works, since integrating a communication system to synchronize agents is required to allow them to start a joint activity together.

3.1.2 Specifications related to Dynamic Environments

Regarding specifications related to the management of dynamic environments, they are based on the three challenges required to address our main issue. These specifications

complement those detailed in Chapter 2.

Specifications regarding Challenge 1 : To produce credible behaviors in the case of dynamic environment, and better address Challenge 1, the agent must respect the following features:

- **Coherence**: The agent must be coherent with the current resources of the world by respecting their availability. For instance, the agent should not start to eat if no food is available. This is why, and as a first specification, we need to have *a process automatically checking the availability of resources during activity execution and interrupting this last one if a resource is unavailable.* In the same trend, the agent must be coherent with the unexpected event coming from the environment. For this, the agent has to address these events during their period of action. For instance, if the phone is ringing, the agent could only answer during the ringing time. Events thus become a priority and consequently, agents must interrupt their activity as soon as possible to address them. In addition, they must be able to automatically resume the interrupted activity after treating events. Consequently, and as our second specification, we need to have *a process to quickly interrupt and resume executed activities when an event happens.* Another specification regards the automation of these processes: *These processes must detect and apply interruptions in an automatic and coherent way.* In other words, the interruptions cases do not have to be explicitly described by users in the activities. This means that users will not have to worry about interruption cases when they create or modify activities: everything is automatically managed by the agent model. By respecting these specifications, we could make several agents coexist.

- **Consistence**: The agent must also be consistent in its reaction when the same activity is interrupted at the same moment between different simulations (e.g. If the agent is interrupted when it is holding a plate, it must release this plate before treating the event). It must be also consistent in its reaction when the same event happens (e.g. The agent will answer the phone when this last is ringing, and will not choose to open the entrance door). Nevertheless, this does not exclude diversity to manage the same situation, just only solutions that are not related. This is why, and as our third specification, we need to have *a process producing*

consistent reactions facing similar interruptions happening at the same time for the same activity.

- **Diversity**: When several activities are possible to restore a resource or address an unexpected event, the agent does not have to choose systematically the same activity every time. For instance, if *Food* needs to be restored, the agent can choose between *Cooking* or *Shopping*. Our next specification is therefore to propose *processes capable of selecting different activities to restore the same resource or address the same event when several choices are possible.*

Specifications regarding Challenge 2 : In Dynamics Environment, the control the users can have over agents is more difficult to maintain. Effectively, more unexpected situations induce less control over resulting behaviors. For instance, if the agent answers the phone and talks for one hour, this can disrupt what the agent had planned to do afterward, especially for activities scheduled to start during this hour. The same issue happens when an activity imposed by users does not have sufficient resources to start. To keep some control over behaviors in the situation of dynamic environments, the agent must be able to anticipate its future exhausted resource. To ensure maximum compliance with the activities imposed by users at specific times, the agent must optimize its free time to restore the missing resources required by these mandatory activities. If there is any time left, the agent has to satisfy its own needs, while respecting time and current resources. This is why, and as our next specification, we need to have a *process able to anticipate future missing resources and schedule activities restoring them.* In parallel, if an activity was interrupted due to a missing resource or an unexpected event, the feasibility of the rest of the plan must be checked. This is why, the next specification induces to have *a process to adapt the rest of the plan after an interruption.* In parallel, resources and events must be configurable to enable users to have some control over their occurrence. This is why, our following specification implies having a *process letting users configure resources and unexpected events before and during the simulation.*

Specifications regarding Challenge 3 : The credibility of behaviors and resulting data has to be checked by diverse methods in the case of dynamic environments. This is why, and as our last specification, we need to propose *validation methods to check the*

credibility of the generated behaviors and data for dynamic environments.

3.2 Overview of Resource and Event Models

After describing our specifications, we now detail the resources and events that are considered in our simulation. We then explain in a general way the three processes created to better address dynamic environments.

3.2.1 Considered Resources

In this section, we detail the resource categories used in this Chapter. Judging from the literature, several categories of resources can exist [65], [120], [165] and three of them are considered in our work:

- **Consumable resources**: These resources become unavailable after their use. If the agent wants to recover a consumable resource, an activity restoring this resource must be made (such as shopping). Concretely, consumable resources are implemented with the following parameters: the name of the object (e.g. Pizza), the resource category (e.g. Foods), and the initial quantity represented by an integer (e.g. 1). When this quantity falls to 0, the resource is consumed and cannot be reused anymore. Among the resources considered consumable, we can find foods, cans, ingredients, pasta boxes, or milk. Some of them can be used several times such as milk or pasta box.

- **Reusable resources**: These resources are temporally blocked during their use and released after. Concretely reusable resources are implemented with the following parameters: the object name (e.g. computer), the resource category (e.g. electrical devices), their availability (blocked/available), and the time of use which is updated each time the resource is used by an activity. In the case of multi-agents, this time of use is shared between the agents to help their activity scheduling and avoid conflicts. These resources must be released by the same activity that blocked them. Consequently, the time of use is at most equal to the activity duration using the resource. Among reusable resources, we can find chairs, sofas, limited rooms, computers, and so on.

Table 3.1 – Examples of Resources and activities using them

Resource Category	Resource Type	Object Name	Consumed/Used by activities	Restored by activities
Foods	Consumable	Pasta, Pizza, Sandwich	Eating	Shopping, Cooking
Ingredients	Consumable	Pasta Box, Frozen Pizza	Cooking	Shopping
Drinks	Consumable	Soda Can, Milk	Drinking	Shopping
Plates	Renewable (Dirty, Clean Filled)	Plate1, Plate2, Plate3	Eating, Cooking	Washing Dishes
Battery Devices	Renewable (Charged, Discharged)	Phone	Using Phone	Charging
Other Dishes	Renewable (Dirty, Clean Filled)	Sauce Pan, Colander	Cooking	Washing Dishes
Glasses	Renewable (Dirty, Clean Filled)	Glass1, Glass2	Drinking	Washing Dishes
Sittable Objects	Reusable	Sofas, Chairs, Desk Chair, Bed	Eating, Computing, Using Phone, Watching TV, Reading	Same activities (since free after use)
Layed Objects	Reusable	Bed	Sleeping, Napping	Same activities (same reason)
Limited Rooms	Reusable	Toilets, BathRoom, Office, BedRoom	Napping, Sleeping, Computing, Showering, Using The Sink, Going To Toilets	Same activities (same reason)
Book	Reusable	Book1, Book2	Reading	Same activities (same reason)
Sinks	Reusable	Kitchen Sink, Bathroom Sink	Using the Sink, Cooking, Going to the Toilets, Washing Dishes	Same activities (same reason)
Electrical Devices	Reusable	Computer, Hood, Oven, TV, HotPlate	Cooking, Computing, Watching TV	Same activities (same reason)

- **Renewable resources**: Renewable resources have several states that can be modified by activities. Each defined state is reversible and can be met again by executing one or more activities. Contrary to consumable resources, renewable resources stay in the world, and activities just impact their current state: their change is thus reversible. Concretely, renewable resources are implemented with the following parameters: The object Name (e.g. Plate1), the resource category (e.g. Plates), and the possible states the resource can take with the transitions between each state (e.g. Clean, Filled, Dirty). These transitions will be defined in the activities acting on these resources: these activities indicate as input the state of the resource they use and as output the state they produce. If renewable resources are not in the requested state, they cannot be used. It will then be necessary to bring them into the desired state with the help of other activities. For instance, if *Eating* requires a clean plate to be executable and all plates are dirty, it cannot be launched. To solve this, *Washing Dishes* can be executed before to get clean plates. Among the possible renewable resources, we can find rechargeable batteries, dishes (which become clean or dirty), etc.

To become a resource, the objects located in the environment can be associated with one of these three categories. For instance, all Food-type objects can also be considered as *Consumable Resources*. Users can turn any object into resources. Table 3.1, summarizes all the resources managed in our use case. This table also shows the activities using these resources and those restoring them.

3.2.2 Considered Events

We detail in this section the unexpected events that can be considered during our simulation. These events are currently related to home environments but they can be enriched at any time, as with resources. In Table 3.2, we stored all the unexpected events considered in our simulation as well as the possible activities to manage them. The duration of activities indicated in the Table are given by way of example and may be modified at any time. Events can be triggered by another agent, by a script indicating the moment when they happen, or by users who could use an interface to trigger them at any time. Users can thus schedule these events in advance if they wish, or trigger them in real-time. In any case, these events are unknown to the agent since they will not be

in its plan: The agent will therefore know neither when nor which event will be triggered during the simulation. The time required to resolve the event is very short since these events have a short duration. For instance, the phone only rings for a few seconds. After this short duration, the event cannot be addressed anymore. This is why, **the agent must be able to quickly pause the activity currently performed to address the event before its expiration**. Users can change the event duration as they want.

Table 3.2 – Exemple of events that can be managed with the activities managing them

Event Name	Max Event Duration	Event Activity	Min/Max Event Activity Duration
Phone Ringing	30 seconds	Answering the Phone	10 min / 1h
The bell Ringing	5 min	Welcoming the person	5 min / 20 min
Receiving an SMS	20 min	Replying to the SMS	2 min / 5 min
Solicitation	10 min	Talking to the person	15 min / 1h
Fire Alarm Triggered	10 min	Disabling Fire Alarm	2 min / 3 min

3.2.3 Models to manage Resources and Events

In this section, we explain the sub-models added to our agent model to fully address dynamic environments. Effectively, we explained in the conclusion of Chapter 1, there is a lack of complete solutions to manage Resources and unexpected events both at the scheduling and execution levels. More precisely, Reactive-based approaches and Learning-Based approaches are well suited to manage dynamic environments since the agent can quickly react to unexpected events. Nevertheless, they cannot anticipate future missing resources and time constraints. In contrast, scheduling-based approaches can anticipate such constraints, but they have difficulties in quickly reacting when unexpected events occur and disturb the rest of the plan. Regarding Planning-Based approaches such as STRIPS [70], HTN [39] or GOAP [24], they can manage resource constraints and adapt the plan when an unexpected event happens. This is why STRIPS is used in robotics for instance, as it can handle the unexpected events of the real world (a missing object, a human cutting the path, etc.). However, they have difficulties anticipating future time constraints and they cannot prepare activities providing resources in advance to avoid future resource depletion. For instance, with these approaches, *Shopping* could be scheduled just before *Eating* to provide food, but it could not be scheduled two hours earlier to

provide food in advance and enable the performance of *Eating* later. Finally, even though Probabilistic-based approaches and Mix-based approaches could manage these kinds of environments, they rarely focus on managing resources and events both at the execution and scheduling levels. Generally, most existing agent-based approaches do not provide any process for stopping an activity in progress where animations could generate errors if stopped abruptly. Moreover, few of them provide both a process to return the agent to a state where it can perform other activities and a process to resume an interrupted activity. Finally, the control users can have over agents in dynamic environments is often limited.

For all these reasons, we propose new models to address dynamic environments. The specifications described in our previous section are the starting points of the conception of these models. According to these specifications, we first need to have a model to anticipate future missing resources. In parallel, we need to have a second model to interrupt and resume activities when they occur. Finally, we need to have a model to adapt the rest of the plan to the consequences of the interruptions. We explain below how these three models are integrated into agent models. It should be noticed that these models are independent of other models already present, and can be disabled for static environments.

The first model located at the scheduling level is the **Resource Manager** that acts during the creation of activity plans to anticipate future missing resources. This process can work closely with a time scheduler (such as our reactive scheduler) to restore missing resources while respecting the remaining time. The second one is the **Interruption Manager** located at the execution level. It was set up to interrupt ongoing activities when a missing resource or an unexpected event happens. It also ensures that the agent can continue other activities or resume the current one after this sudden interruption. The last one is the **Plan Checker** located at the scheduling level. It was done just before the selection of the next activity to verify its feasibility. This checkpoint is essential to verify if the plan is always feasible after a possible interruption. Figure 3.1 shows our updated agent model where these three models are integrated to fully address dynamic environments. We present below their global processes as well as the modification made in the External Perception Model to record changes in the world state:

External Perception Model: At the initialization phase, the External Perception Model retrieves all resources to store them in the World Database. In parallel, all events happening in the environment are also stored in this database. Each time a resource

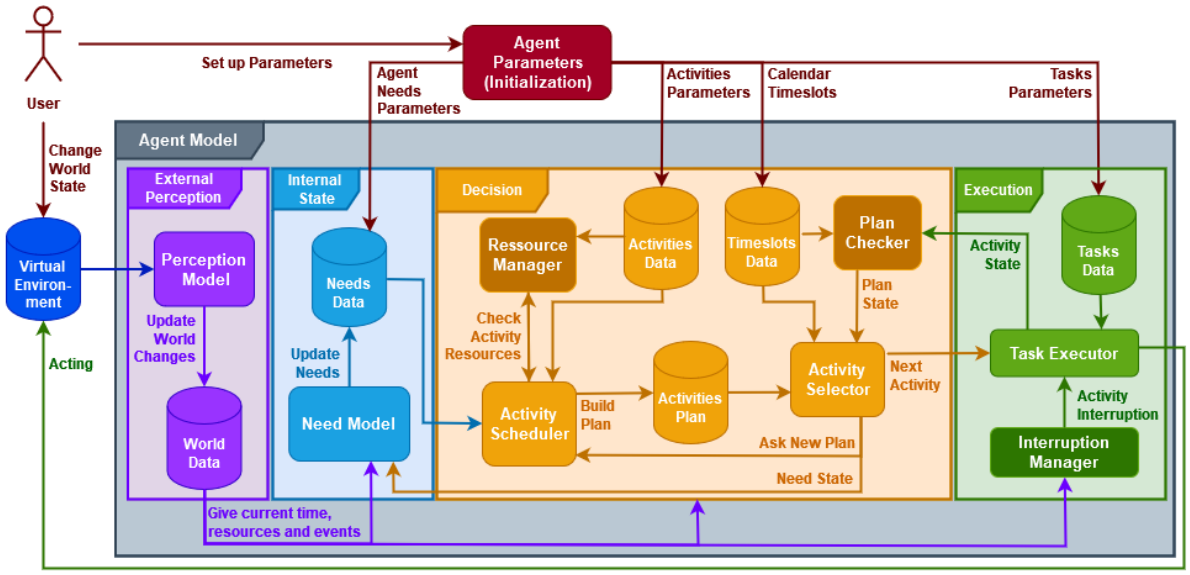

Figure 3.1 – Diagram of our Agent Model for Dynamic Environments

changes or an event happens, the External Perception Model will register the changes and send them to the different models.

Resource Manager: The resource manager is used to detect and restore the missing resources related to each activity that the agent wants to schedule. To restore them, a process is launched to select activities producing the missing resources. Consequently, the manager works closely with the Activity Scheduler which is a reactive scheduler in charge of building the plan of activities. Concretely, we thus have the reactive scheduler to manage the time constraints given by users and the agent needs constraints given by the Internal State Model. In parallel, we have the Resource Manager to ensure that each scheduled activity is compatible with the currently available resources. To know this current state, the Resource Manager retrieves this information from the External Perception Model. In addition, it also retrieves information from the Activities Database to know which resources are consumed and produced by which activity. During the creation of the plan, the Scheduler sends to the Resource Manager each activity that should be scheduled as well as the remaining time. In return, the Resource Manager will send an ordered list of activities restoring resources to schedule before the concerned activity. To find this list, the Resource Manager checks the availability of the resources consumed by the input activity. If some resources are unavailable, the process explained

above is launched to find activities restoring these resources while respecting the remaining time. The final plan is then created by the Activity Scheduler and sent to the Activity Selector. More details of this step are given in section 3.3.

Interruption Manager: After receiving the plan, the Activity Selector sends the next scheduled activity to the Task Executor to be launched. During execution, some resources can suddenly be exhausted or an event can occur. For instance, if during *Eating*, another agent eats the food before its use, then *Eating* cannot continue. The Interruption Manager was thus conceived to address these unexpected changes and manage interruptions. The Interruption Manager concretely receives as input a notification from the External Perception Model indicating that an unexpected change happened. The Interruption Manager then works in collaboration with the Task Executor to stop the current activity in a safe way (i.e. without generating animation errors). An process is then launched by the Interruption Manager to let the agent return to a state where it can perform another activity. To do this, *Undo* actions cancelling the reversible changes caused by the current activity are executed. If necessary, a process can also be launched to return the agent to the state where the interrupted activity was stopped. In this case, actions applying again the reversible changes caused by the interrupted activities are executed. With this, the agent is able to resume interrupted activities. In the case of unexpected events, the Interruption Manager also launches the activity addressing this event. When the interruption process is finished, the Task Executor is notified by the manager to either continue the paused activity or finish it prematurely. The Task Executor then notifies the Plan Checker that an interruption has occurred. More details about the Interruption model are explained in section 3.4

Plan Checker: The Plan Checker checks if the rest of the plan is always feasible after the execution of any activity. For this, the availability of resources that are consumed by the next activity is checked. If an interruption notification is also received, the current time is also checked to know whether the agent is early or late. Depending on the impact of these interruptions, the Plan Checker can either make some adjustments to the plan or request a rescheduling by emptying the rest of the plan. The modified plan is then sent to the Activity Selector which either selects the next activity or requests a new plan to the Activity Scheduler if the modified one is empty. More details are given in section 3.5.

3.3 Resource Manager

In this section, we detail the process of the Resource Manager to anticipate future missing resources and to create a plan being coherent with the currently available resources. To reach this, the Resource Manager works with the Activity Scheduler to check if the solutions proposed to restore missing resources are compatible with the remaining time allocated to the plan. The priority is first to schedule activities restoring the missing resources used by the future mandatory activity indicated in the user's calendar. Actually, doing everything possible to avoid their failure is essential to respect user constraints as much as possible and address our Challenge 2. After managing the mandatory activity, both models work together to schedule activities that can satisfy urgent needs while respecting time and resource constraints. If there is still time to place other activities, default activities are scheduled in the plan to keep the agent busy. To prevent these activities from disrupting those already scheduled, default activities only depend on reusable resources. After this, relaxing user constraints is possible to satisfy an urgent need by interrupting the mandatory activity if resources and time constraints are respected. To illustrate these changes, we proposed an example in Figures 3.2, 3.3, 3.4, 3.5, and 3.6. The whole process of plan creation including resource constraints is given below:

1. Retrieve time budget and current resources: The Activity Selector gives to the Activity Scheduler the time window of the plan and the next mandatory activity happening at the end of the plan (if there is one). The scheduler then turns this time window into a time budget. In parallel, the Resource Manager retrieves the current state of resources from the External Perception Model. All the missing resources are stored in a smaller database called *RMissing*. This database is essential to know this since the scheduled activities consuming resources will impact the rest of the plan. *RMissing* is updated each time an activity is scheduled in the plan to consider the consumed and produced resources. In the example shown in Figure 3.2, we have a 3-hours time budget to schedule and *Eating* as the next mandatory activity. In parallel, the Resource Manager detects that Food, Cleaned Plates, and Cleaned Glasses are exhausted and that the Bathroom is blocked until 6 p.m. These missing resources are thus stored in *RMissing*.

2. Filter activity according to time and resource constraints: In this step, we seek to reduce the range of possibilities by keeping only activities whose minimal duration is below the time budget. This filtering process is made by the Scheduler to exclude any

activity that cannot be scheduled during the plan. For instance, if the time budget is equal to 3 hours, the *Sleeping* activity is automatically excluded since its minimum duration is equal to 8 hours. After this, all these possible activities are sent to the Resource Manager to categorize them according to the resources they produce. This classification is used to facilitate the selection of activities to restore a missing resource.

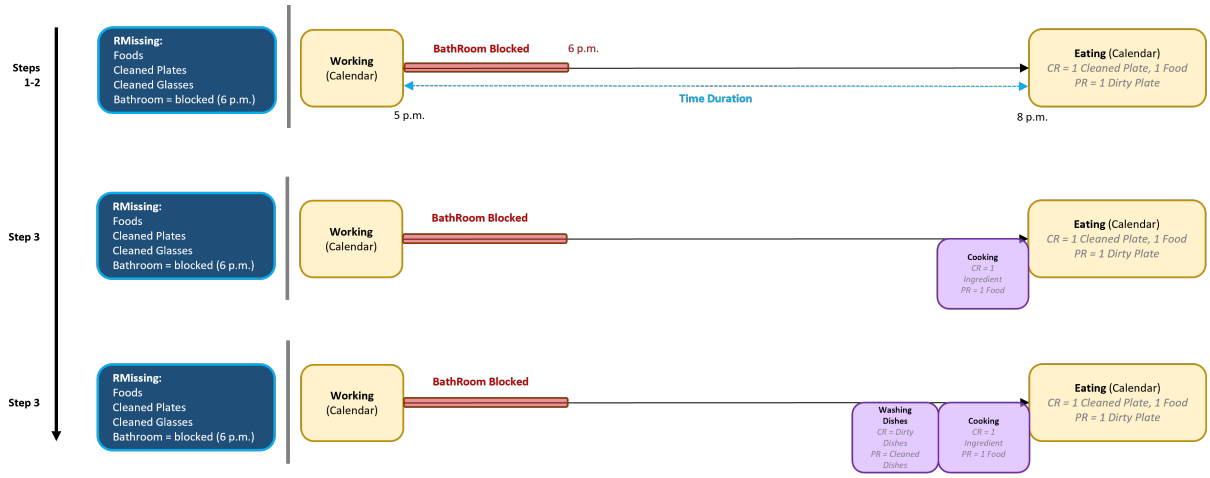

Figure 3.2 – Example of plan creation with resources (steps 1,2, and 3). CR means Consumed Resources, PR means Produced Resources

3. Managing the Resources of the Mandatory Activity: Before considering activities satisfying needs, the agent has to ensure that the resources of the Mandatory Activity are available. If they are unavailable, they must be restored during the plan by scheduling activities producing them. If no mandatory activity is indicated, this step is skipped. To do this, the mandatory activity as well as the time budget is first sent to the Resource Manager. In return, the Resource Manager sends to the Scheduler the list of activities restoring the required missing resources. They are then scheduled just before the mandatory activity and the time budget is updated by subtracting their duration. If no resource is required, the Resource Scheduler just sends a notification to continue. In the case where no suitable activity can restore one of the missing resources, the Resource Manager asks the Scheduler to postpone the mandatory activity. With this, the time budget is extended to offer more possibilities to restore the missing resources. The mandatory activity can only be postponed until the start time of the mandatory one scheduled after. If postponement is impossible, the mandatory activity is canceled. We

give below the process made by the Resource Manager to select the activities restoring the missing resources required by the mandatory one:

a. *Checking reusable resources*: The Resource Manager first checks the reusable resources used by the Mandatory Activity. If one of them is blocked, the manager asks the Scheduler to postpone the Mandatory activity at the time when the resource is available again.

b. *Checking missing resources*: After this, the renewable and consumable resources are also checked. Since the Mandatory Activity is scheduled at the end of the plan, its produced resources are not considered. For each missing resource required by the mandatory activity, the Resource Manager tries to find activities restoring it. These last ones are first filtered as follows: the priority is given to those consuming no missing resources. After this, the manager keeps those consuming only one missing resource without putting priority. To limit the complexity, the Manager ignores those having a duration exceeding the time budget, those consuming more than one missing resource, or those requiring one blocked resource. After this, the Resource Manager randomly selects a remaining activity by putting a weight on those having a priority. If the selected one consumes missing resources, the same process is made to find an activity restoring these resources and consuming no missing resources. If no such activity is found, the Resource Manager selects another activity and tries again the process. If no solution is found either, the Resource Manager asks the Scheduler to postpone the Mandatory Activity. When activities restoring a missing resource are definitively selected, their consumed and produced resources are stored temporally and considered for the next selections. This avoids interlocking and conflicts with the activities selected to restore other missing resources. Each new activity is ordered before the ones yet placed. When all the missing resources are treated, the Resource Manager sends to the Scheduler the ordered list of activities restoring the missing resources. This list is also kept by the Resource Manager to check if the Mandatory Activity is always executable, even when activities satisfying needs are scheduled before.

In the example stated in Figure 3.2, *Eating* requires one renewable resource (Cleaned Plate) and one consumable resource (Food) to be executable. However, Foods and Cleaned Plate are not available in the current state. This is why, the Resource Manager first looks for an activity restoring Food. Among the possible activity producing Food, the

Manager can find *Shopping* and *Cooking*. Since both of them have no missing resources, the Manager randomly selects one. *Cooking* is thus chosen and placed before *Eating*. Since *Cooking* needs to have 1 Ingredient to be executable, this resource is temporally stored to be not used by the activities that will be selected to restore other missing resources. After this, the Resource Manager looks for an activity producing Cleaned Plates. Among the possible activities, we have *Washing Dishes* which does not use missing resources and washes all the dirty dishes. *Washing Dishes* is thus selected and placed before *Cooking*. In the end, *Cooking* and *Washing Dishes* are sent to the Scheduler. They are then scheduled just before *Eating* and the time budget is reduced by the duration of both activities.

4. Anticipating Future Urgent Needs: As explained in Chapter 2 section 2.5, the scheduler first asks the Internal State Model to give the list of future needs that will be urgent during the plan with their urgency interval. A time of urgency called t_{Need} is then extracted from intervals to help the placement of activities satisfying needs. The Resource Manager is not involved in this step. In our example shown in Figure 3.3, four urgent needs are found: *Hygiene*, *Sport*, *Thirst*, *House Hygiene*, and *Toilets*.

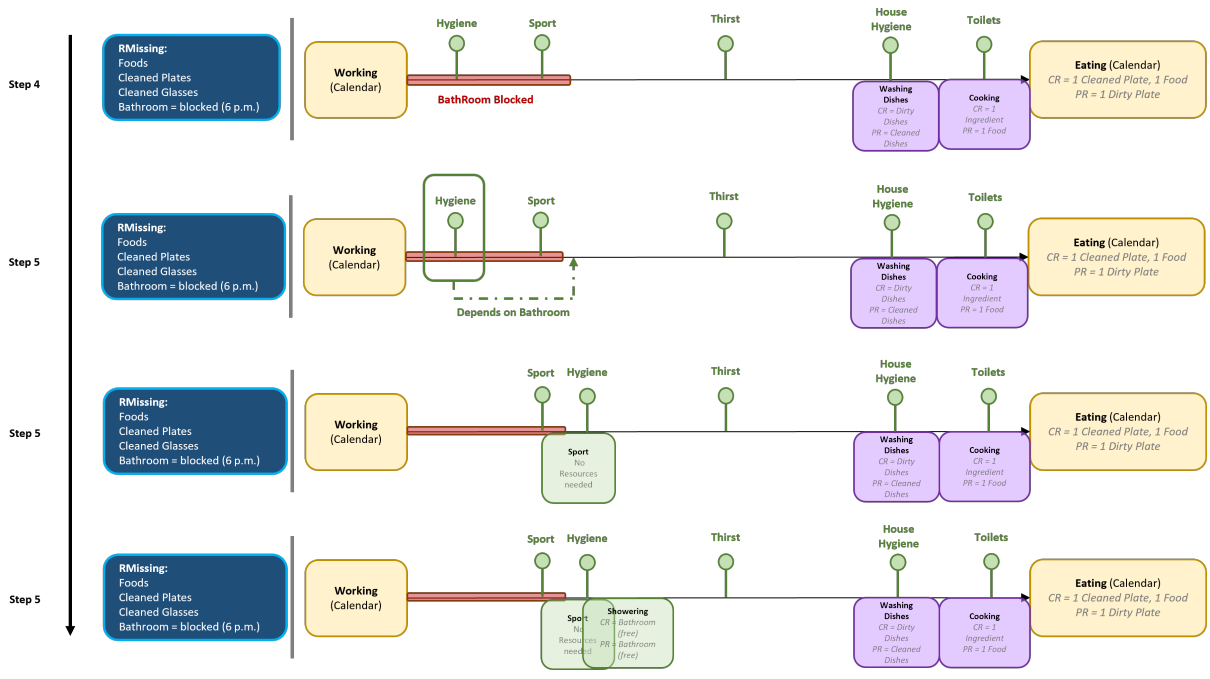

Figure 3.3 – Example of plan creation with resources (steps 4 and 5). CR means Consumed Resources, PR means Produced Resources

5. Associate Needs to Activities: In this step, activities satisfying needs will be scheduled if they respect the time and resources constraints. By starting with the t_{Need} closest in time, the Scheduler first sends to the Resource Manager the remaining time budget, the current t_{Need}, and the possible activities to satisfy the related need whose duration is below the time budget. Since activities and needs are linked at the beginning of the simulation, the scheduler can quickly retrieve them. In return, the Resource Manager sends the activity selected to satisfy the need with the list of activities restoring the missing resources consumed by this one. The Scheduler then places these activities just after t_{Need} and reduces the time budget by their minimal duration. To avoid useless redundancy, a list of activities to delete can also be sent if the found activities restore some missing resources used by the Mandatory Activity. If no activities are found, the Manager sends a notification to either postpone t_{Need} or to delete it, depending on the cause. The scheduler then applies the request, chooses the next t_{Need}, and sends the new activities. The t_{Need} can only be postponed until the end of the plan. If postponing is impossible or if the need can be satisfied later by another occurrence in the plan, then the t_{Need} is deleted. The process used by the Resource Manager to select activities according to their resources is given below:

a. *Choosing an activity satisfying the need*: Since the Scheduler can send several activities to satisfy a need, one of them must be first chosen by the Resource Manager. To do this, they are first ordered according to the number of missing resources that are required: The highest priority is given to those not using missing or blocked resources. After this, we find those using only one missing or blocked resource and finally those requesting more than one missing and blocked resource. The Resource Manager selects the less restrictive activity and then performs steps *b* and *c* in the case where some missing resources are requested by this activity. If no activity is found to restore the missing ones, a second activity able to satisfy the need is tested. If the process fails again, the manager sends a message to cancel the t_{Need}. The limitation in the number of tested activities before abandoning is voluntarily restricted to limit the complexity of our solution. However, this number can be modified if desired.

b. *Checking missing resources*: The Resource Manager first checks the missing resources used by the activity satisfying the need. Activities able to restore each missing resource are then searched. The process is the same as the one explained

in Step 3. The activities found to restore a missing resource are placed before those yet placed, and their consumed resources are temporally memorized to avoid conflicts with the future ones selected to restore other missing resources. In the case where no activity is found to restore one of the missing resources, the Resource Manager returns to step *a*.

c. *Checking reusable resources*: The Resource Manager checks the reusable resources used by the activity satisfying the need. If one of them is blocked, a message is sent to the Activity Scheduler to postpone the t_{Need} at the time when the resource is available again.

d. *Detecting conflicts with the Mandatory Activity*: After selecting the activity satisfying the need and the potential activities restoring its missing resources, the Resource Manager checks the resource states of the Mandatory Activity. To do this, *RMemory* is updated by applying the effects of these new activities. In the case where *RMemory* changed, the manager first detects if some resources used by the Mandatory Activity are restored by these effects. If some of them are effectively restored, then activities scheduled in step 3 restoring the same resource are placed in a delete list to avoid redundancy. After this, the manager detects if some resources used by the Mandatory Activity are now missing due to these new effects. If some of them are effectively missing and cannot be counterbalanced by the activities yet scheduled in step 3, then the activity satisfying needs cannot be placed and the process returns to step *a*. *RMissing* is then reset. Otherwise, the activity satisfying the need is sent to the Scheduler in addition to the list of activities restoring its missing resource and the list of activities to delete. In this case, *RMissing* keeps the changes.

In the example shown in Figure 3.3, the *Hygiene* t_{Need} is postponed after 6 p.m. because all the activities satisfying *Hygiene* use the bathroom, which is blocked until this time. Since *Hygiene* is postponed, the Scheduler takes the next one (*Sport* need). The Resource Manager then selects *Doing Sport* which has no missing resources. This activity is then placed at the *Sport* t_{Need}. After this, *Hygiene* is selected and *Showering* can be now placed since the Bathroom is now available. The process continues with *Thirst*, where *Drinking* is found to satisfy it (shown in Figure 3.4). However, *Drinking* requires a Cleaned Glass which is missing. Therefore, a second activity must be found to

produce a Cleaned Glass. *Washing Dishes* is thus selected since all its consumed resources are available. At this step, the Resource Manager detects that Cleaned Plates are also restored by this new *Washing Dishes*. Consequently, one of the missing resources related to *Eating* is restored. This is why, the *Washing Dishes* previously scheduled to restore a resource of *Eating* is not useful anymore and thus deleted. *Washing Dishes* and *Drinking* activities are then placed at the *Thirst* t_{Need}. After this, *Cleaning* is placed to satisfy *House Hygiene*. Finally, *Going to the Toilets* activity, requiring no missing resources, is found for the last need and scheduled.

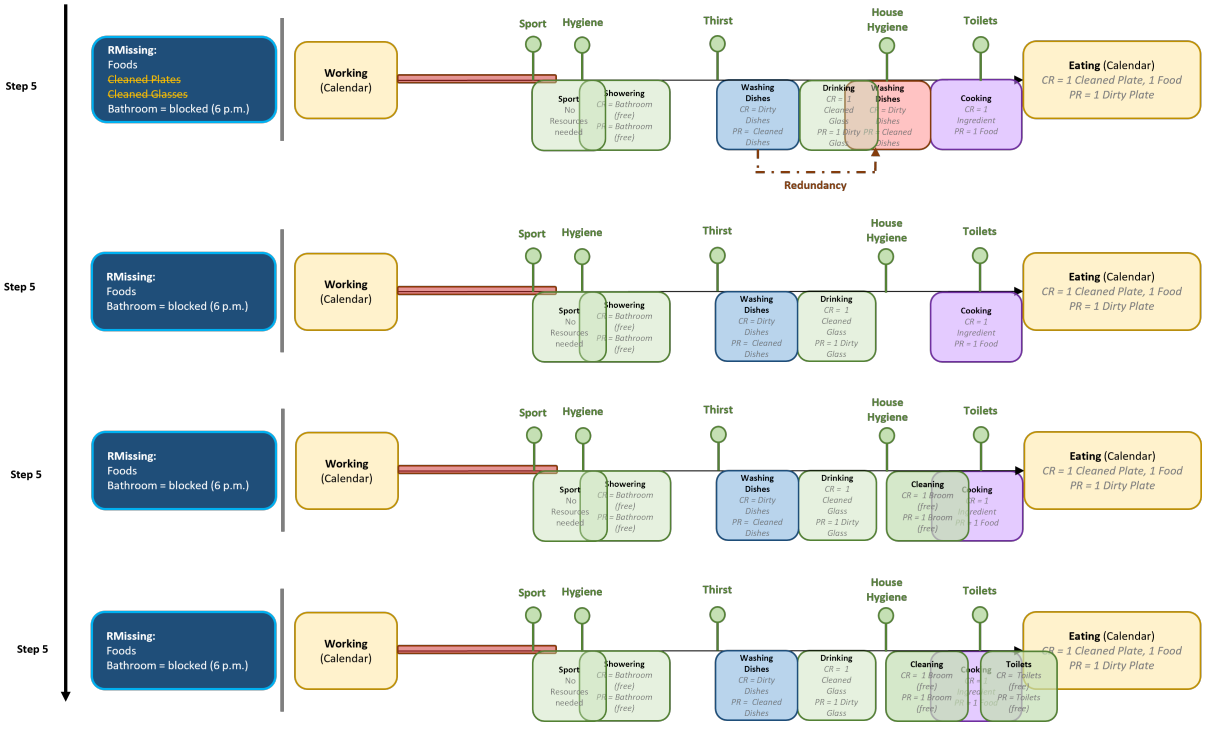

Figure 3.4 – Example of plan creation with resources (continued step 5). CR means Consumed Resources, PR means Produced Resources

6. Positioning Activities and Adjustment of their Duration: When all activities satisfying needs are placed in the schedule, their start time and end time will be shifted by the Scheduler to avoid overlapping. Concretely, the scheduler first starts at the beginning of the plan to check if start times are well positioned. The scheduler then starts at the end of the plan to check if end times are well positioned. After this, the Resource Manager is solicited to check if the activity impacted by blocking resources starts at the time when

these resources are available. It should be noticed that the activities placed at the end of the plan to restore the resource of the mandatory activity are not concerned by this shifting step. In the end, the shifted activities ending after those linked to the Mandatory one are deleted. In the example given in Figures 3.5, *Showering* is first shifted since the start time is before the end of *Doing Sport*. The other ones are not shifted. The process then checks the end times. *Going to the Toilets* and *Cleaning* are first shifted to start earlier since they overlap *Cooking*. This move implies shifting the previous activities to avoid overlap. However, due to this shifting, *Showering* starts earlier and now begins when the Bathroom is blocked. This is why, the Resource Manager notifies it must be shifted again to start at the right times. This move implies shifting the next activities to avoid overlap. Consequently, *Going to the Toilets* exceeds the time limit and is deleted from the plan.

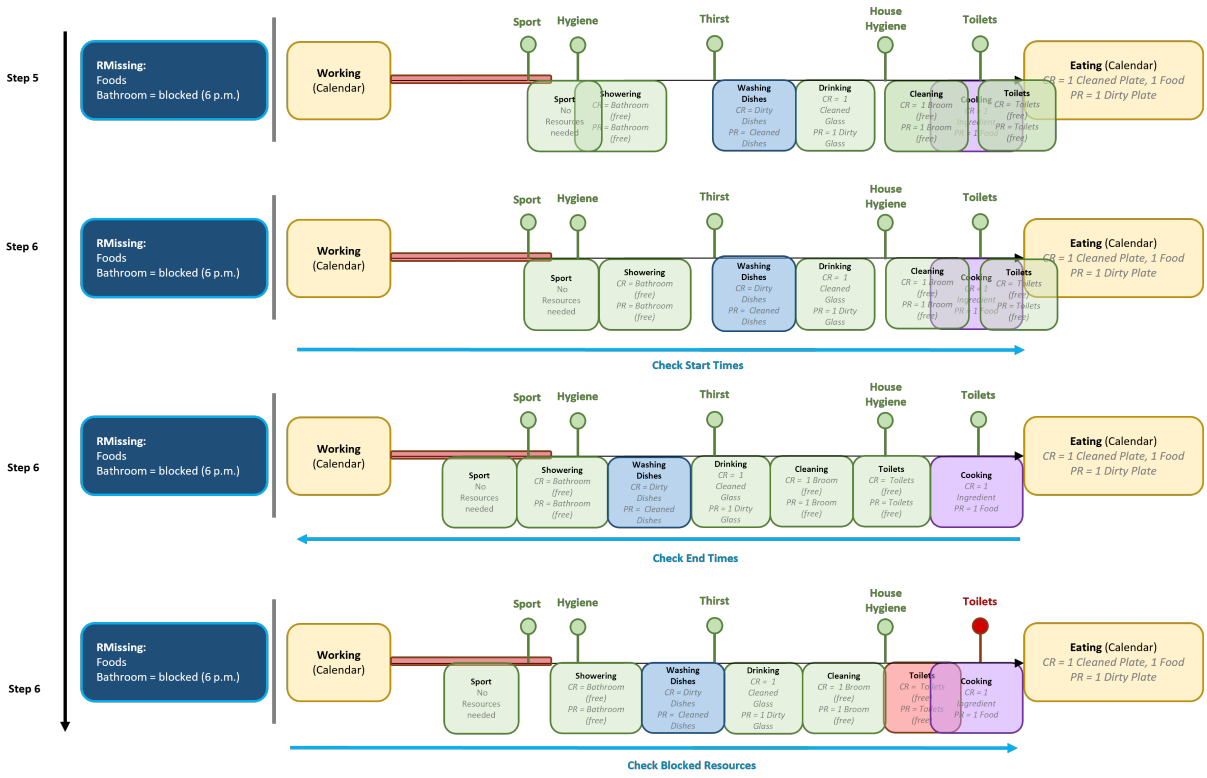

Figure 3.5 – Example of plan creation with resources (step 6). CR means Consumed Resources, PR means Produced Resources

Activities are then extended by the Activity Scheduler in the same way explained in Chapter 2. The small change is that activities having blocked resources during the plan

which cannot start earlier to avoid conflicts. However, they can be extended to finish later if there is free time after it. In the example shown in Figure 3.6, *Doing Sport* and *Drinking* have free time around them, they are thus extended until their maximum time. After them, *Washing Dishes* is also extended until *Drinking*.

7. Filling Gaps: Default activities can be selected by the Scheduler to fill the free time periods between activities. Since these activities only use reusable resources, the Resource Manager only checks if the used resources are not blocked at the desired time. The Scheduler also checks if their minimum duration is inferior to the free time period. In the example detailed in Figure 3.6, there is a gap between *Doing Sport* and *Showering*. Consequently, the *Watching TV* default activity is placed with the duration of this gap.

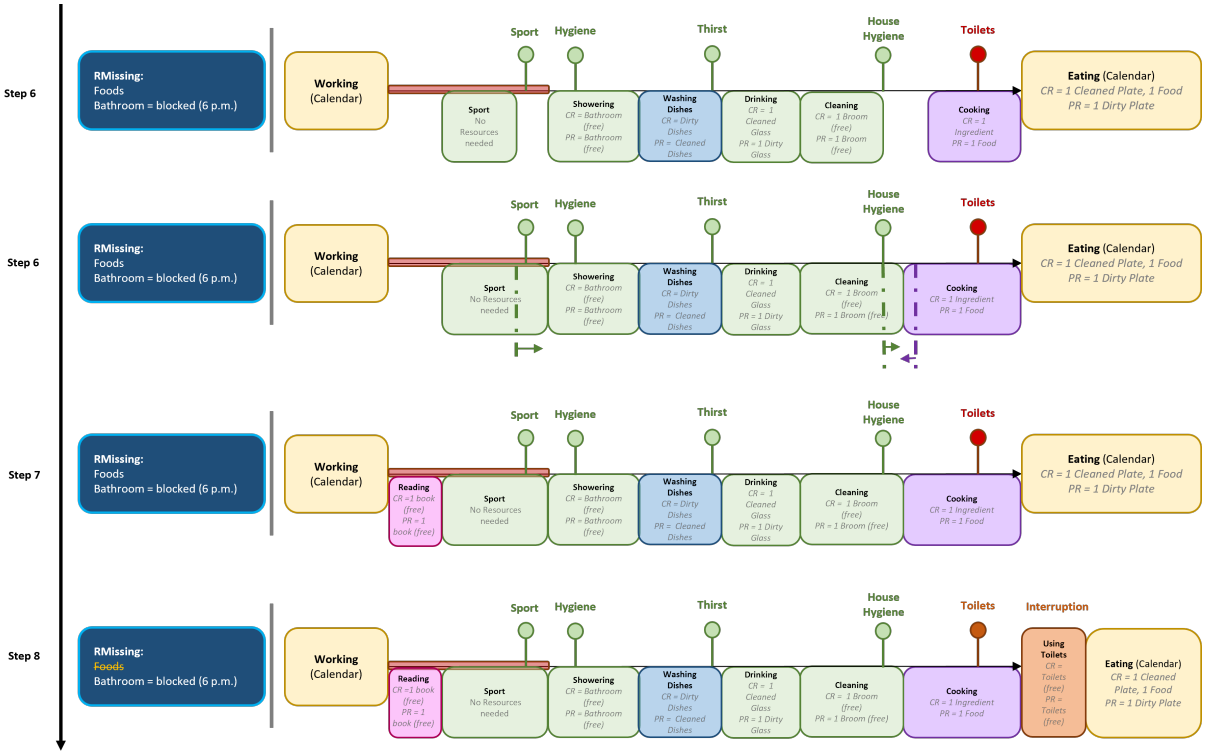

Figure 3.6 – Example of plan creation with resources (Steps 7 and 8). CR means Consumed Resources, PR means Produced Resources

8. Checking interruptions: This interruption step is used to relax the duration of the mandatory activity in the case where an urgent need could not be satisfied. The activity satisfying the urgent need can only interrupt if its minimum duration is included

in the duration of the interrupted one and if this last one is interruptible. In addition, to be able to interrupt, all the resources of the activity must be available and the resources used by the interrupted one must not be consumed. The Scheduler will be in charge of integrating this interruption and the Resource Manager is also implicated to check the resource constraints. In the example shown in Figure 3.6, the *Toilets* need could not be satisfied, and *Eating* was considered interruptible. Since *Going To the Toilets* activity has not missed resources and does not use the resources consumed by *Eating*, it is then scheduled with its minimum duration. *Eating* is then reduced by this duration.

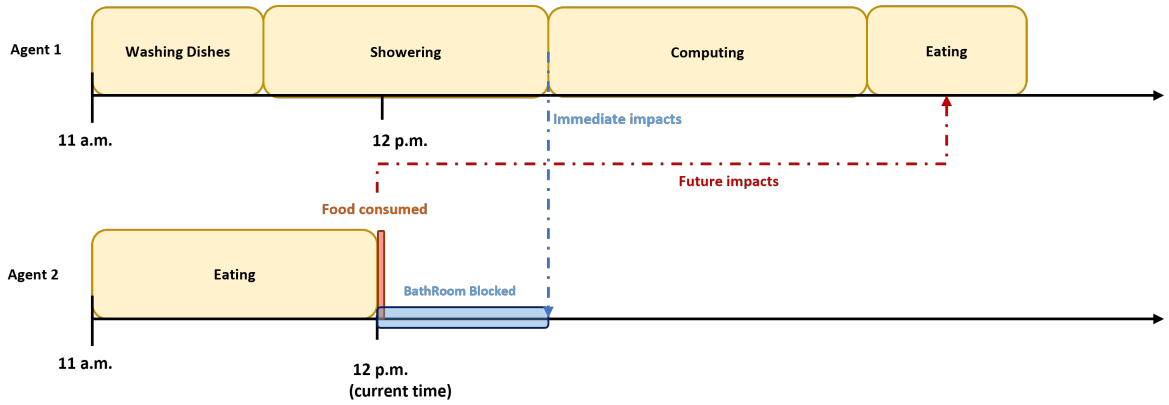

Figure 3.7 – Example of situations the actions of one agent impact the other

In this section, we shown an example of a plan construction in the case where resources are considered. The Resource Manager is essential for multi-agent situations to simulate coordinative and conflicting interactions as said in section 3.1.1.2 since this model can provide plans in accordance with the resources yet used by other agents. An example of a situation where the agent actions can impact other agents can be shown in Figure 3.7 where two agent calendars are shown in parallel. In this figure, we are at the point where Agent 2 has ended its last scheduled activity (*Eating*) and is preparing to build a new plan. Since *Eating* was executed, there is no available food anymore. In parallel, Agent 1 keeps performing its proper scheduled activities: at midday, Agent 1 is still performing *Showering*. Consequently, the Bathroom is blocked when Agent 2 starts to schedule its new plan. In this situation, any activity using the Bathroom such as *Showering* or *Using the Sink* will thus not be scheduled by Agent 2 during the blocking period. Finally, since Agent 2 has consumed the available food, this missing resource will disturb Agent 1 plan in the future since *Eating* is scheduled later. This situation will be managed by the Plan

Checker at the moment when Agent 1 will end *Computing*.

3.4 Interruption Manager

In this section, we detail the Interruption Manager used to manage resources and events at the execution level. Effectively, even though the plan is created in a coherent way and checked each time after an executed activity, unexpected events or missing resources can suddenly happen. Unfortunately, we concluded in Chapter 1 that existing approaches rarely propose solutions to manage interruptions for 3D environments. This is why, we offer a process to interrupt and allow the agent to recover a state where it can correctly perform any other activity. In addition, we also suggest a process to allow the agent to return to the state where it can resume the interrupted activity.

By using the Interruption Manager, we also want to automatically detect and treat interruption cases. When interruptions are considered in the literature, users often have to manually add interruption cases inside their activities [182] when Automatons such as Petri-nets or Finite state machines are used. This means that every time they want to create or modify a new task, they must imagine all the interruption cases and implement new branches to consider interruptions. More concretely, if they want to create a *Filling a glass* task requiring cleaned glass, a breakpoint must be created by hand to consider the case where no cleaned glass is present. They also have to manually indicate any undo tasks to be done when interruptions happen, so that the agent returns to a state where it can perform any other activity. For example, if the agent is holding a glass and is sitting down when the phone is ringing, users have to manually create a breakpoint and add the *Put the glass down* and *Stand up* actions so that the agent is ready to answer the phone. Consequently, creating or modifying an activity becomes a very cumbersome task. This is why, the idea behind the Interruption Manager is to automatically take care of all interruption processes when Automatons are used so that users can concentrate solely on the case where the activity works normally. This model is thus flexible since it automatically adapts to the situation in which the interruption occurs. It is also independent of the executed activity: any activity can be interrupted automatically without manually adding any specific code or actions. Our model was inspired by GOAP and STRIPS [70], but we propose in our case a solution compatible with Automatons. This compatibility is also useful for our use case since the #SEVEN model [115] implemented in the *Xareus*

Software[1] and based on Petri-Nets is used for activity execution. In addition, we wanted to develop a solution that was easier for users to configure compared to STRIPS or GOAP where preconditions and effects must be added for each activity. To do this, the manager concretely records the reversible changes caused by the executed activity and executes actions canceling them when an interruption happens. More generally, two different processes can be launched by the Interruption Manager to handle interruptions, depending on whether the interruption is due to exhausted resources (section 3.4.1) or unexpected events (section 3.4.2).

3.4.1 Managing Resources at the execution level

During the activity execution, the requested resources can be suddenly unavailable. To manage this situation, the process illustrated in Figure 3.8 is launched:

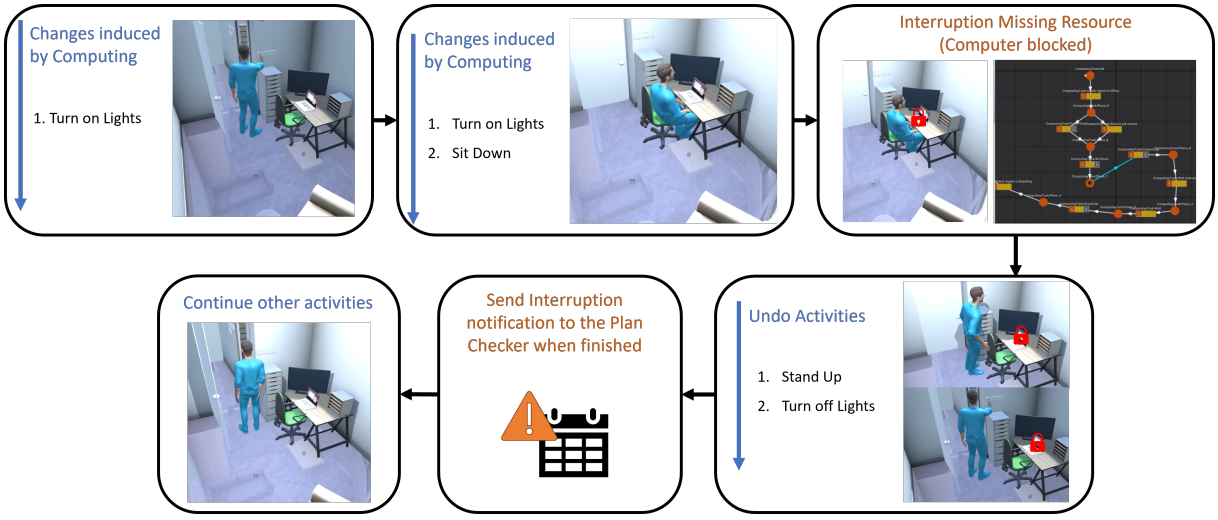

Figure 3.8 – Example of missing resource management at the execution level

1. Stopping the current scenario: When an activity is launched, the associated sequence of tasks is launched to execute actions in the 3D environment. In the case where a task consumes a resource, the External Perception Model is automatically notified to give the state of this resource. If this resource is not available, the condition required to execute the tasks is not reached and the scenario is blocked. The Interruption Manager

1. *Xareus Software*: `https://team.inria.fr/hybrid/xareus/`

is then noticed and the scenario is stopped. In our example shown in Figure 3.8, the agent started to perform *Computing*. However, when the agent has to interact with the computer, this last one is unavailable. Consequently, *Computing* is stopped by the Interruption Manager before the use of the computer. Concretely, the token moving in our #SEVEN scenario based on Petri-Nets is stopped by the resource sensor detecting that the computer is not available.

2. Undo process to cancel reversible changes: When an interruption happens, the agent can be in a state where it cannot perform any other activities. For instance, the agent may be sitting while reading a book. Consequently, it cannot immediately perform another activity if no actions to release the book and get up are launched. To solve this, the Interruption Manager has memorized in a stack all the reversible changes caused by each executed task. These changes concern the blocking of a reusable resource (chairs, books, etc.), the agent body state (sitting, laying, etc.), the objects taken in hands, and the devices turned on. The irreversible changes such as the consumed or renewable resources are not considered. Effectively, renewable resources take too much time to return to their previous state (ex: the agent must perform *Washing Dishes* to turn a dirty plate into a clean plate). After stacking the changes, an Undo process is launched to return the agent as quickly as possible to a state suitable for other activities. For each unstacked change, actions canceling the reversible changes are launched. For instance, if an object is grabbed, then an action to release this object is launched. These actions are selected by starting with the one solving the last change. The treated changes are then removed from the stack, and when this stack is empty, the process is stopped. In our example, *Computing* ended due to a missing resource (computer blocked). Before this interruption, Reversible actions such as *Light turned on* and *Agent is seated* were registered when the related tasks were executed. When *Computing* is interrupted, the Interruption Manager unstacks reversible changes and executes the corresponding Undo Actions: The first one is *Getting Up* to cancel *Sitting Down* and the next one is *Turning Off Lights* to cancel *Turning On Lights*. These Undo actions are made through #SEVEN scenarios in our implementation.

3. Notify Interruption: When this process is finished, The Task Executor is notified and an interruption notification is sent to the Plan Checker. In our example, a notification indicating that *Computing* ended prematurely is sent to the Plan Checker.

3.4.2 Managing Events at execution level

Managing events at the execution level is challenging since they can happen at any time during an activity execution. This is why, we need to have a process able to interrupt activities at any time while avoiding animation errors caused by a sudden stop. This process only concerns executed activities that are indicated as interruptible by users: if an event happens during a non-interruptible activity, then this one is not considered. *Showering* is an example of such activity: if the phone is ringing, the agent cannot answer since it is under the shower.

When an event happens, the External Perception model detects the event and sends it to the Interruption Manager. This last one then checks if the executed activity is interruptible. In this case, the process illustrated in Figure 3.9 is launched as followed:

1. Pausing after the current task: In this step, the Interruption Manager identifies two cases: the case where the current task contains an animation lasting a few seconds and the case where the animation is repeated to last several minutes. When the animation lasts a few seconds, the activity is stopped after this animation to avoid errors due to no ended movements. This is the case for grabbing an object, turning on a device, etc. When an animation is repeated during a determined duration, the duration is shortened to the minimum to let the animation finish its last loop. The scenario is then stopped just after. In contrast to the resource case, the scenario stays active to enable relaunch after the event. In Figure 3.9, the agent is performing *Cooking* when the Phone Rings. This event happens at the moment when the agent is picking up the pasta box. The Interruption Manager then waits for the end of *Grabbing Pasta* action before pausing *Cooking* by stopping the token of the #SEVEN scenario.

2. Undo process: To address events, the agent must be in a state where it can launch any other activities. Consequently, the same Undo process explained in section 3.4.1 is also used in this step. Concretely, by launching Undo actions to cancel reversible changes caused by executed tasks, the agent can reach this state. In the case of events, the reversible changes caused by the interrupted activity are kept in memory after executing undo activities since they will be applied again to resume the activity, as explained in step 4. The localization of the agent is also memorized to allow the agent to return to the room after addressing the event. In the example shown in Figure 3.9, the reversible changes made by *Cooking* are *Turning on Plates* and *Grabbing Pasta*. Undo actions are

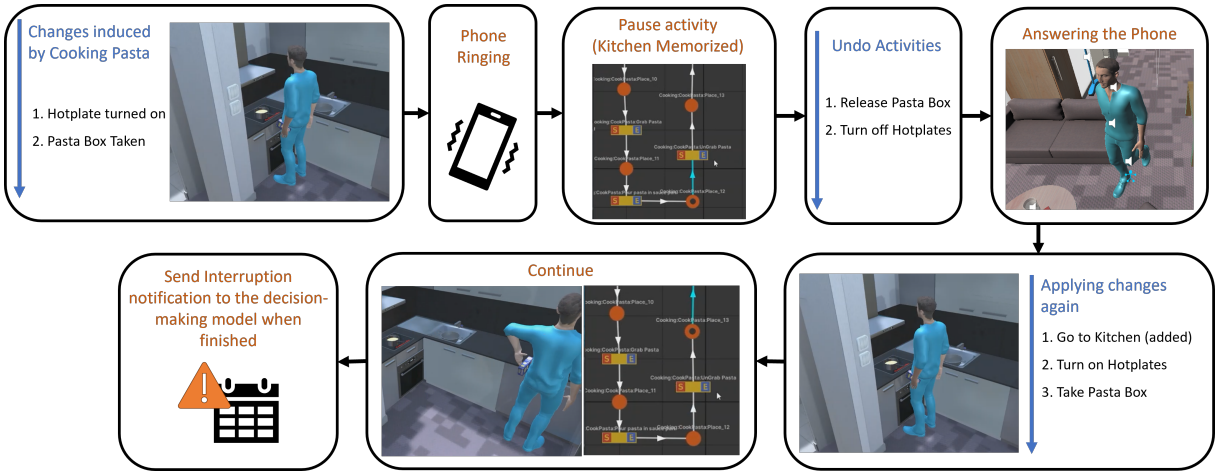

Figure 3.9 – Example of an event management at the execution level

thus executed to release the pasta box and turn off the hotplates before answering the phone. In addition, the Kitchen room is also memorized.

3. Launching the scenario addressing the event: When the activity is paused and the undo actions are executed, the Interruption Manager then executes an activity addressing the event. As default activities, these activities can only exploit reusable resources to avoid resource conflicts after. In the example shown in Figure 3.9, the agent launches *Answering Phone* to address the event.

4. Resume the interrupted activity: When the event is managed, the Interruption Manager executes actions to apply again the list of changes that were caused by the paused activity. This process enables the agent to be in the same situation as the moment when it was interrupted. The activity can be thus continued when it was stopped. After executing these actions, the Task Executor is notified to resume the paused activity. Before this, the Task Executor checks if the current time exceeds the end time of the paused activity. In the case where the time is not exceeded, the paused activity is relaunched with a reduced duration to end at the expected times when it is possible (by respecting the minimal duration limit). Otherwise, the new duration will correspond to the minimum duration minus the duration already spent by the activity before the interruption. When the duration is established, the activity is resumed with the newly allocated duration. In the example illustrated in Figure 3.9, the agent hangs up the phone and uses this process

to go back to the kitchen, turn on hotplates, and take the pasta box. The token of the #SEVEN scenario is then released and the agent can now pour the pasta in the saucepan.

5. Notify Interruption: When the activity is finished, the Task Executor sends an interruption notification to the Plan Checker.

3.5 Plan Checker

When an executed activity ended, the Plan Checker verifies if the rest of the plan is still feasible. In the case where the activity ended normally, only resources used by the next activity are checked. Otherwise, an interruption happened and the Plan Checker verifies whether the agent is now behind or ahead of schedule. After this checking, some adjustments can be applied by the Checker to adapt the rest of the plan. If the changes have too much impact, rescheduling is required. This section describes more precisely how the Plan Checker checks the next activity resources (section 3.5.1) and the impacts of interruptions (section 3.5.2).

3.5.1 Checking Plan Resources

Whether the executed activity was completed normally or affected by interruptions, the Plan Checker always verifies the resources of the next activity. Effectively, even though the plan is coherent with the resources available at the beginning of this plan, resources can independently evolve from the agent actions. Consequently, some resources that were not used by the executed activities could also disappear during their executions, without impacting them since they are not used. However, these new missing resources could impact the next activity scheduled after. For instance, if the agent is performing *Cooking*, the fact that another agent uses the computer is not an issue. However, if the next scheduled activity is *Computing*, this last one will be impacted if the computer is still being used by the other agent when this activity has to start. This is why, it is essential to set up a process checking the activity resources just before its launching.

Each time a resource changes, the External Perception Model updates the world database. To check the next activity scheduled in the plan, the External Perception Model is requested to know if the resources related to this activity are always available. If the answer is negative, the activity cannot be performed.

To limit the use of rescheduling processes, the Plan Checker first checks if this failed activity can be replaced by another activity producing the same kind of resources. If none of them is found, the Plan Checker tries to replace the failed activity with the activity scheduled just after. To do this, the required resources of this last activity are checked as well as its maximum duration to know whether the activity can be extended to take the time slot of the failed activity in addition to its initial one. If it is possible, the duration of the next activity is extended to start now and end at the expected time.

Otherwise, a process of rescheduling is launched: The rest of the activity plan is dropped and the Plan Checker sends the empty plan to the Activity Selector. Since no activity is scheduled, the Activity Selector asks the Scheduler to produce a new plan for the remaining period of the failed plan.

In the example shown in Figure 3.10, *Food* is exhausted when the agent is executing *Washing Dishes*. This situation can happen if users voluntarily drop all the available foods or if another agent decided to eat this food for its proper objectives. Since Food is not required by this current activity, the agent can finish normally *Washing Dishes*. After this, the Plan Checker checks the resources of *Eating*. However, since Food is missing, this activity could not be performed anymore: a plan adjustment is required. The Plan Checker thus checks if the *Computing* activity situated after *Eating* can start now. Since *Computing* can be extended and executable with the current resources, the plan is adjusted to start *Computing* now. The rest of the plan is not changed.

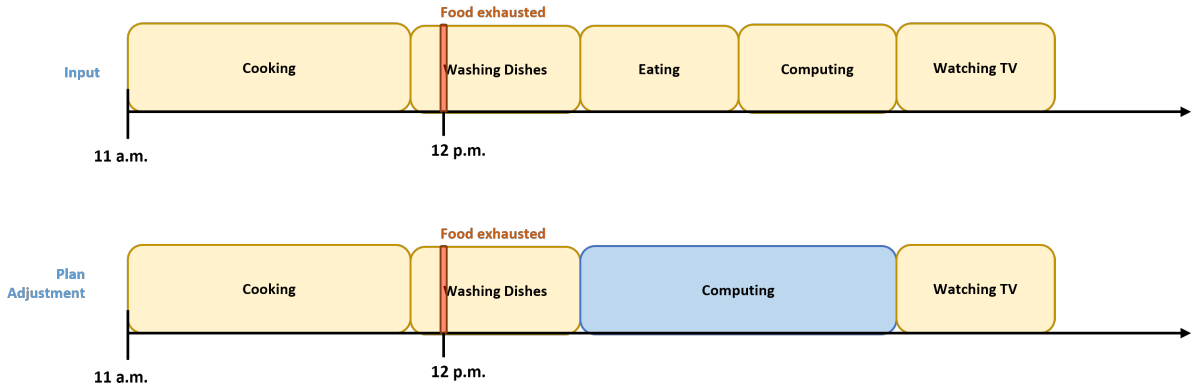

Figure 3.10 – Example of a plan adjustment made by the Plan Checker after a missing resource: above, the initial plan, and below, the adjusted one.

3.5.2 Checking Plan after Interruption

In the case where the Plan Checker receives an Interruption notification from the Task Executor, a deeper verification is launched to assess whether the agent is now behind or ahead of schedule. Before checking resources, the Plan Checker will first compare the current time with the one that was supposed to be reached at the end of the executed activity. Three cases can thus happen:

The interrupted activity ended in time: This situation can happen when the interruption caused by an unexpected event was sufficiently short to not delay the interrupted activity. The process to check resources detailed in the previous section is thus launched. In the example given in Figure 3.11, *Reading* was interrupted by *Receiving an SMS*. Since *Answering an SMS* is a short activity, *Reading* ended at the scheduled times. The rest of the plan is thus not impacted by this interruption.

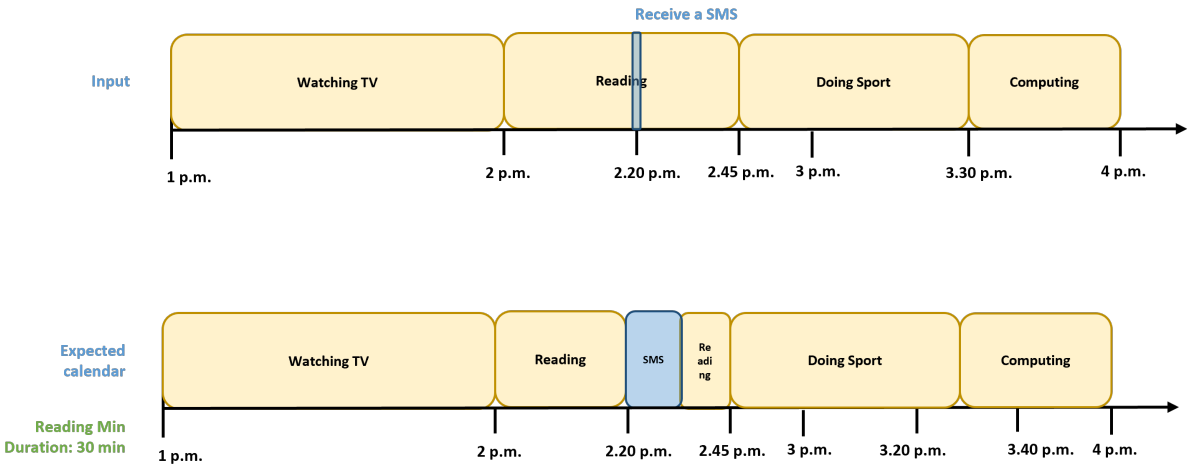

Figure 3.11 – Example of a plan adjustment made by the Plan Checker after a short Event: above, the initial plan, and below, the adjusted one.

The interrupted activity ended earlier: An activity can finish earlier if a required resource is suddenly missing during its execution. Consequently, the interrupted activity could not be completed, and some effects may not be realized, such as resource production. For instance, if *Cooking* was suddenly stopped because the Pasta Box was empty, the Food resource was not restored during the execution. This may have an impact on the next activities, such as *Eating* if *Cooking* was prematurely stopped. This is why, the Plan

Checker first verifies whether the next scheduled activity can still be performed despite the early termination of the current activity. If possible, the duration of the next activity is extended to start now in the case where its maximum duration is not exceeded. Otherwise, The Plan Checker verifies if another activity producing the same kind of resources can be scheduled during the gap by checking its minimum duration and the availability of its required resources. In the case where no activity is possible, a default activity is put to fill the free time period. In the example shown in Figure 3.12, the ingredients were exhausted during the execution of *Cooking* before their use. This exhaustion could be due to another that already used the ingredient just before or due to the users' actions voluntarily wanting to drop ingredients to observe the agent reaction. Users can effectively interact with the simulator to trigger events or deplete resources at any time through the use of an interface or specific keyboard keys. Due to this, *Cooking* ended earlier and could not produce any food. Nevertheless, a similar activity called *Shopping* is found to produce Food. Since *Shopping* can be performed with the current resources and with the imposed duration, it is scheduled in the plan to start now. The rest of the plan is not changed.

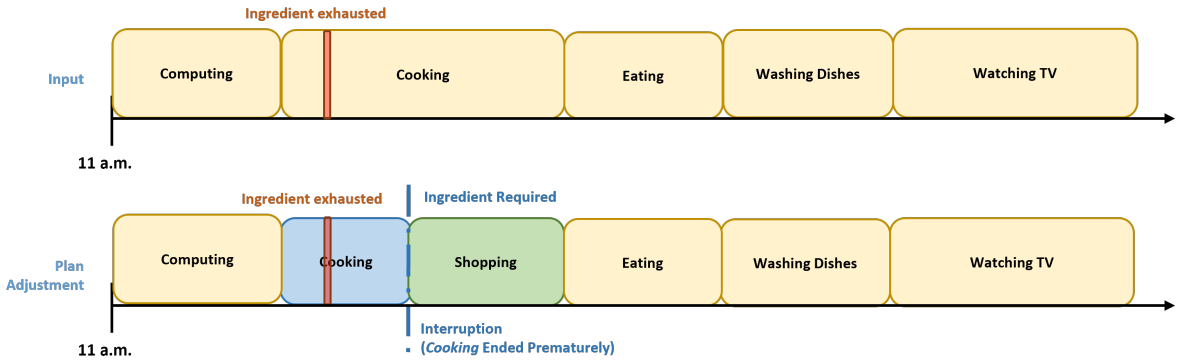

Figure 3.12 – Example of a plan adjustment made by the Plan Checker after a resource interruption: above, the initial plan, and below, the adjusted one.

The interrupted activity ended later: This situation can arise when the interrupted activity was delayed due to an unexpected event taking too long time to address. If the current time is within the time slot of the next activity, the Plan Checker tries to cancel the delay by reducing the duration of this next activity by the amount of lost time. This solution works only if the minimum duration of the next one is sufficient to support this time diminution. Otherwise, the plan is considered compromised: The rest of the plan is dropped and the empty plan is sent to the Activity Selector to trigger a rescheduling.

Actually, this rescheduling is used to limit the delay impacts over the next mandatory activities. In the specific case where mandatory activities had to start before the current time and were prevented from doing it due to the event, the plan is dropped. The mandatory ones impacted by this delay are then consecutively added to the empty plan with their minimum duration if they have sufficient resources. The plan is then sent to the Activity Selector to be executed. In this way, the delay should be limited while avoiding the cancellation of many mandatory activities. In the example given in Figure 3.13, *Reading* was interrupted by the *Phone Rings* event. Since *Answering the Phone* is a long activity, *Reading* ended later and impacted the next activities: *Doing Sport* and *Computing* are strongly compromised. In the case where they are not mandatory, a rescheduling process is chosen: the plan is emptied. Otherwise, if *Computing* is mandatory, then the rest of the plan is emptied, and *Computing* is added to the plan to start now with its minimum duration.

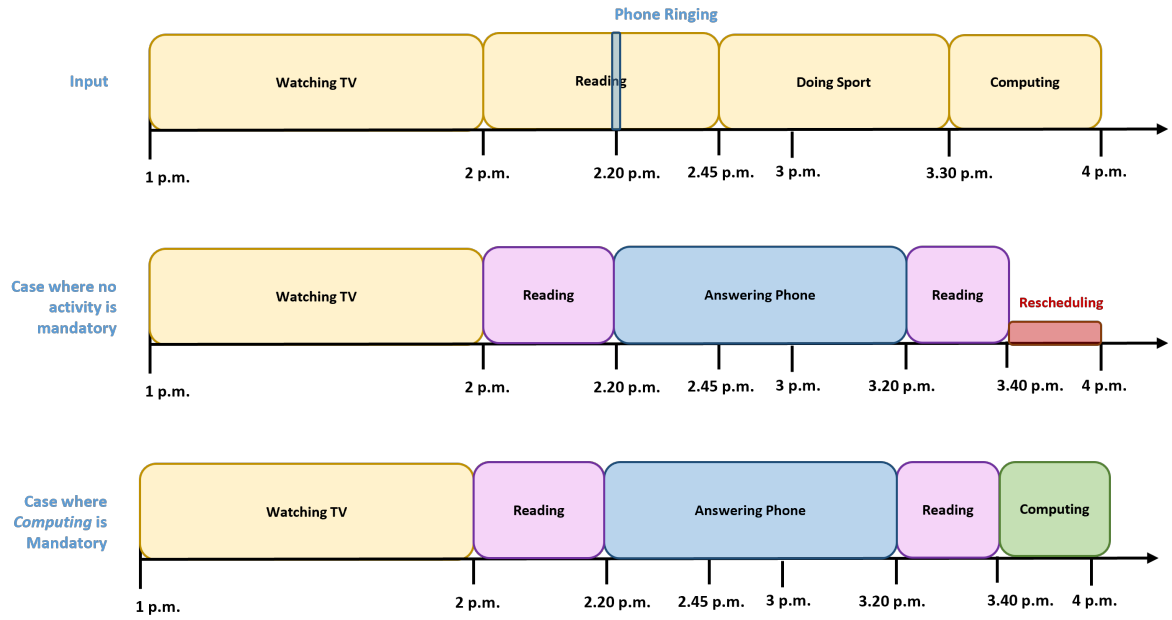

Figure 3.13 – Example of a plan rescheduling made by the Plan Checker after a long Event: above, the initial plan, and below, the impacted one.

3.6 User-Experiment Protocol

In this section, we propose a preliminary version of a user-experiment protocol to validate our agent model for dynamic environments. **This protocol may therefore evolve before making the experiment.** In addition, pilot tests will be set up to check and adjust the parameters proposed in this experiment protocol, such as the number of videos, the number of questions, the duration of activities, and so on. The main question related to this experiment is the following one: Does our agent model produce behaviors close to what a human would do when confronted with dynamic environments?

Global process: This experiment will be split into two parts. The first part, explained in section 3.6.1, is designed to evaluate the credibility of the agent behaviors at the execution level. In this step, participants will be invited to analyze the credibility of the agent reactions in the situation where this last one is interrupted by an unexpected event when an activity is currently executed. The second part, explained in section 3.6.2, is set up to assess the credibility of the agent behaviors at the scheduling level. In this part, participants will be invited to schedule an activity calendar after an interruption. This calendar will be then compared to the one produced by our agent through the use of objective validation methods such as statistical analysis. This protocol was inspired by several existing works detailed in Chapter 1 proposing user questionnaires or comparison methods between agents and humans to validate the credibility of the generated behaviors. For instance, the work of Jorgensen et al. [95], those of Park et al. [139], those of Darty et al. [53], and those of SMACH [156], propose the protocols that are particularly relevant to evaluate behavior credibility.

Participants: This experiment is not submitted to any specific restrictions regarding participants. Ideally, we would like to have at least twenty participants. We also want to cover different age brackets (from 18 years old), respect gender parity, and have people working from different fields (not just those working in IT or research). The same participants will be used for the both following parts.

3.6.1 Part 1: Evaluate behaviors at the execution level

This first experiment part aims to show that our agent can interrupt and resume activities in a credible way from a human's point of view when an unexpected event

occurs. With this, we could have a credibility assessment concerning the agent behaviors in the case of dynamic environments. The collected data are the participants' answers of a questionnaire retrieving their feelings. The main hypothesis of this first part is the following: *Participants have the feeling that the agent reacts as a human would facing an unexpected event.* To do this, we propose the following protocol:

Setting up the Experiment: Before making the experiment, several video showing an activity execution will be created to retrieve the different feelings of participants for the same situation. We decided to choose 3 activities that are relevant for interruption cases since they impact a lot the environment with reversible and non-reversible changes. These activities are *Cooking (Pasta)*, *Cooking (Pizza)*, and *Washing Dishes*). In addition, we also choose 3 non-interruptible activities to retrieve the participants' opinions regarding situations where activities are not interrupted during an event. These non-interruptible activities are *Showering*, *Going To The Toilets*, and *Dressing*. In parallel, 4 events are chosen for the experiment to diversify the kinds of interruption: *Fire alarm activated*, *Phone ringing*, *Receiving SMS*, and *Doorbell ringing*. After this, Each activity will be simulated with one of the 4 events (for instance *Wahsing Dishes* will be interrupted by *Phone Ringing*). These events will be programmed before the simulation to happen precisely at the desired moment. We will finally obtain 6 different videos (1 for each activity) that will be used for the experiment. Videos should be between 2 and 5 minutes: the simulated time is effectively faster that the real time to prevent participants for watching too long.

1. Video Observation: Participants start the experiments by watching the 6 videos showing the 6 different activities disturbed by one of the 4 unexpected events described above. Each video will be shown in a precise order: The order of the videos will be the same for all participants, alternating between interruptible and non-interruptible activities. After each video, the questionnaire detailed in step 3 is given. Viewing and answering questions should take about 40 minutes.

2. Active interaction with the simulator: In this part, participants will directly interact with the simulator to trigger real-time interruptions during the execution of an activity. **Only one interruption will be authorized by activity**. An interface is first shown to choose between the 6 activities listed above. Once an activity is chosen, the

activity will start whenever the participant decides. At any time during the simulation, the participant can trigger one of the 4 events listed previously. If participants have not triggered any interruptions, the trial is not counted and they will be asked try again. Otherwise, after each interruption simulation, the questionnaire detailed in step 3 is given. This step should also take about 40 minutes.

3. Questionnaire: After watching a video or interacting with the simulator, a questionnaire is given to the participants to retrieve their feeling about the credibility of the agent reactions facing unexpected events. These questions listed below are inspired by those proposed by Schumann et al. [156], Renoux et al.[148], Jorgersen et al.[94] and Darty et al. [53] to evaluate the credibility of their agent behaviors during activity execution or activity scheduling:

1. *In your opinion, was the agent in a situation where it could interrupt its work?* The answers can be *Yes*, *No*, *I do not know*.
2. *In your opinion, did the agent react to the unexpected event in a reasonable time?* A scale from 1 to 5 is given as the answer. 1 corresponding to *Too Slow* and 5 corresponding to *Too fast*.
3. *In your opinion, did you find the agent address the situation in a credible way?* A scale from 1 to 5 is given as the answer. 1 corresponding to *Not at all credible* and 5 corresponding to *Very credible*
4. *In the case of interruption, did you find the agent resumes the activity in a credible way? (Excluding animation)* A scale from 1 to 5 is given as the answer. 1 corresponding to *Not at all credible* and 5 corresponding to *Very credible*
5. *In the case where the agent's behavior does not seems credible to you for handling unexpected events, would you have any suggestions to improve this behavior?* The answer is a paragraph written by the participant

Analysis: A statistical evaluation using means and standard deviation will be performed to study the answers.

3.6.2 Part 2: Evaluate behaviors at the scheduling level

This second part of the experiment aims to show that the schedules produced by the agents after an interruption are close to those produced by a human. In this way,

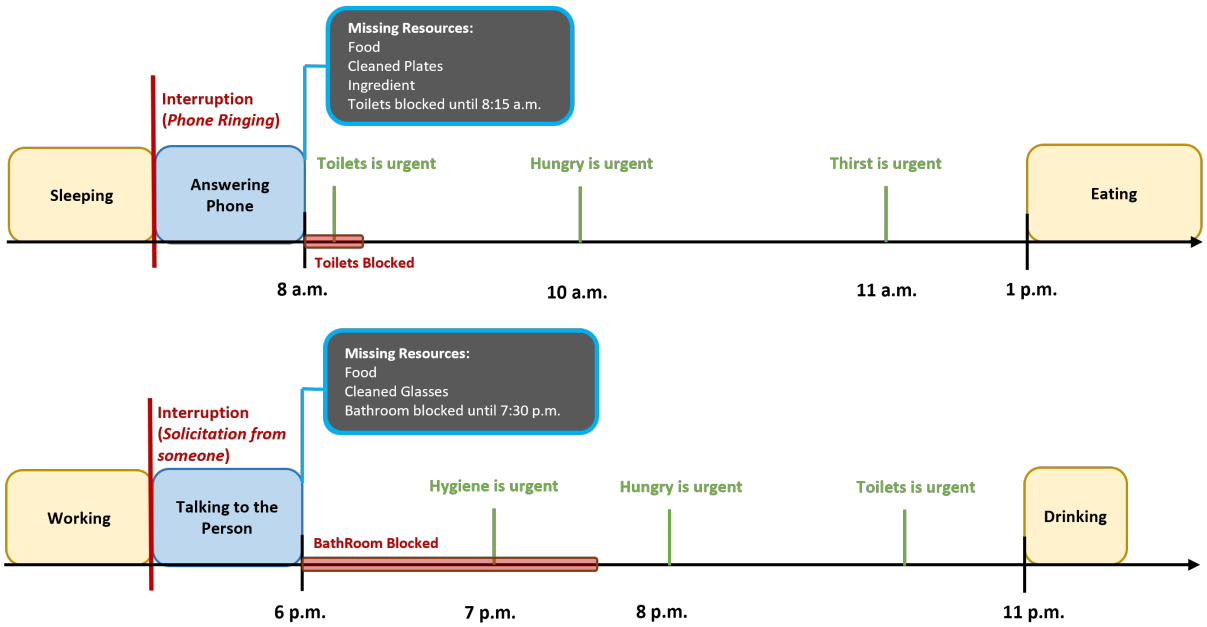

Figure 3.14 – Calendars that will be used for the user Experiment

we could have a credibility assessment concerning the agent behaviors produced at the scheduling level in the case of dynamic environments. The main hypothesis of this part is the following: *There is little difference between an activity schedule made by a human and one made by an agent after an interruption.* To do this, the following protocol is proposed:

Scheduling a Calendar: In this second part, the same participants will then complete two already-started calendar (one for the morning and one for the evening) that were stopped just after the management of an interruption. The calendar to complete ends with a mandatory activity having not sufficient resources to be executable in the current situation. The objective is therefore to fill the schedule between the current moment situated just after the interruption and the moment when the future mandatory activity starts. The two schedules given to the participant is shown in Figure 3.14. To place activities in the plan, Participants will interact with a 2D interface illustrated in Figure 3.15 containing the plan to be completed at the top, the activities currently possible at the bottom, and the missing resources on the left side. The participant can drag and drop activities into the schedule and can also adapt the activity duration between a minimal and a maximal value. Activities that cannot be performed due to missing resources are

greyed out. The moment where needs start to be urgent are also displayed to users to provide the same information level that agents can have during their scheduling. The participant can choose from 15 activities indicated in Figure 3.15. This second part should last around 30 minutes. Before the experimentation, the agent will be simulated in the same situation and the output calendar will be retrieved.

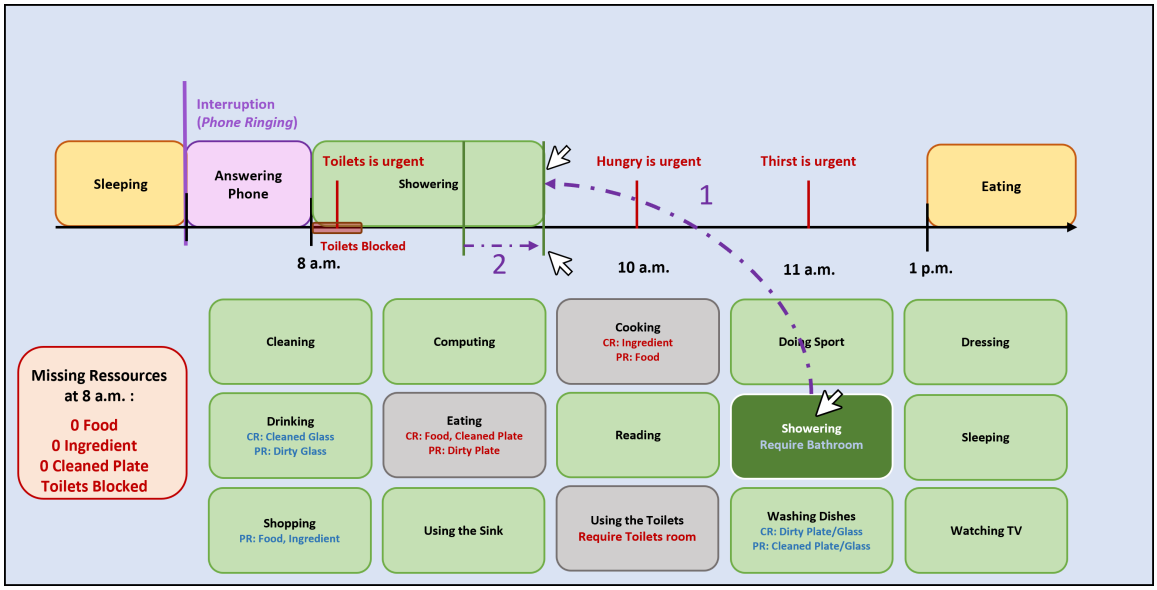

Figure 3.15 – User Interface prototype to schedule daily activities. CR means Consumed Resources and PR means Produced Resources. (1) shows how users can drag and drop activities on the plan and (2) shows how users can extend durations

Analysis: We propose objective validation methods to assess the credibility of agent behaviors. For this purpose, the participants' schedules will be compared with the agent one through the following metrics:

- The activities chosen by both participants and the agent, and their occurrence.
- The mean duration and start time retrieved for each activity chosen by participants and the agent.
- The activities only chosen by humans and those only chosen by agents.
- Error rate related to the case where the mandatory activity scheduled at the end cannot be performed with the proposed plan because resources were not restored.
- The number of satisfied needs for the agent and participants
- The sequences of activities both found in the participant and agent calendars (e.g. *Cooking* -> *Eating* -> *Washing Dishes*) and their occurrence.

By using these metrics, statistical analysis, as made by Jorgersen et al. [95], or clustering methods, as made by Darty et al. [53], can be performed to analyze results and check our hypothesis.

3.7 Conclusion

During this chapter, we propose a new model made of three sub-models to better address dynamic environments. The first is the Resource manager used to anticipate future missing resources and produce a plan coherent with the currently available resources. The second one is the Interruption Manager used to manage resources and unexpected events at the execution level. The last one is the Plan Checker used to check whether the rest of the plan is always feasible after an interruption. These models allow the agent to consider resources and unexpected events at both scheduling and execution levels while managing the other constraints described in Chapter 2 (time, needs, etc.). We also detailed in section 3.6, the preliminary version of a user experiment protocol to validate the agent model in the case of dynamic environments through the inclusion of missing resources and unexpected events impacts.

With this new agent model version presented in this Chapter, new scenarios and situations are accessible for users wishing to use our model to generate data, populate environments, or enhance their existing simulators. For example, they can now simulate several agents simultaneously, whereas it was previously complicated to do this due to the impact of each one on the activities of others. Multi-agent data can therefore be created. They can also simulate scenarios specific to dynamic environments such as the case where there no food is available, the bathroom is occupied this morning, and so on. In addition, with the ability to schedule events in advance or trigger them during the simulation, they can also simulate situations implying unexpected events such as the telephone ringing at 12 p.m., an object breaking down during the evening, the fire alarm going off during *Cooking*, and so on. Actually, if we have simulated such situations with the version proposed in Chapter 2, the agent would have ignored the unexpected event and continued as if nothing had happened.

Regarding the three challenges that we want to address, these 3 models enable us to better address Challenge 1 related to the credibility of behaviors since environmental impacts are now handled. Regarding Challenge 2 requiring a compromise between control

and autonomy, our models are designed to respect the constraints given by users as long as they are compatible with the environment state. However, in order to address Challenge 3 related to validating the credibility of behaviors and data, we must perform the user-experiment protocol proposed in this section as well as some Input-Output comparisons checking the respect of input parameters. Following this idea of strengthening validation, we propose in Chapter 4 another validation method that compares the effectiveness of simulated data with real ones to train machine learning algorithms.

Chapter 4

Methods to validate the credibility of the data generated by the Simulator

Chapters 2 and 3 showed how we can address the first and the second challenges through the design of an agent model that can give control and autonomy over behaviors and execute daily activities in a 3D environment. In particular, the ability to perform activities in a 3D Environment allows the generation of accurate synthetic data, as explained in our Chapter 1. Now, we present our *a posteriori* validation method to meet the third challenge regarding the way to validate the credibility of the synthetic data generated by the agent. To address this challenge, we propose to replicate a real data collection experiment in a virtual home, using our agent model in place of a human subject. To find out whether our simulated data can be used in the same way as real data, **we propose to evaluate the credibility of simulated data by using them for context-aware machine-learning models.**

To evaluate this, we have chosen two human understanding tasks: current activity recognition and future activity prediction tasks. According to chapter 1, several possible validation methods can be used to compare simulated data with real data for activity detection or prediction, as shown in the survey of Chaquet et al. [42] which summarizes the existing approaches using synthetic data for activity detection. Among the existing *a posteriori* validation methods for activity detection and prediction, we have the comparison of the performance rate obtained from the algorithms between the case where synthetic data are used and the case where real data are used. This is the type of validation that we use here to compare the credibility and effectiveness of our synthetic data. The final objective of this credibility assessment is to show whether simulated data can replace real ones to train machine-learning models that can recognize and predict real situations.

Activity recognition described in section 4.2 allows us to assess the credibility of the

interactions performed by the agent in the environment as well as the sensor behaviors. By comparing the performance gap in activity recognition, we can find out whether the agent interacts with the environment in a coherent and accurate way, and whether the performed tasks are close to those recorded in reality. We can also know if the diversity of performed tasks is sufficient for the detection algorithm to be robust when analyzing real activities. This also enables the detection of missing or imprecise interactions between the agent and the virtual sensors/effectors. In parallel, we can also assess if sensors located in the VE can capture the agent interactions in an accurate way.

Activity prediction described in section 4.3 allows us to assess the credibility of the agent decision-making model. By comparing the gap in performances of prediction between the use of real and synthetic data, we can evaluate whether the agent can schedule activity routines close to real ones. Effectively, since this algorithm predicts the future activity in relation to the activities performed just before, it can identify logical patterns and habits in the choice of activities. For example, if a person has prepared food and cooked, then the *Eating* activity is likely to happen after. By analyzing the performance of the prediction algorithm according to whether they have been trained with real or simulated data, we can check if the agent's choices are sufficiently coherent and diversified to produce routines giving enough robustness to this algorithm to predict real routines with similar performances. We could then reinforce the assumption that the routines produced by the agent are sufficiently credible to substitute real data in this domain.

Whether it is activity detection or prediction, our aim is not to outperform the result obtained with real data, but rather reduce the gap between them. For example, if the successful prediction rate is 20% when the algorithm is trained with real data, then we also aim to have a prediction around 20% when this algorithm is trained with synthetic ones. For fair comparisons between real and simulated data, it is important to replicate as closely as possible the real home, the real sensors, the real subject, and the real experimental protocol used to collect the real dataset. Section 4.1 thus describes how the real environment and experimental protocol have been replicated.

It should be noted that the agent model used for these validation experiments is the one presented in Chapter 2. The additional model presented in Chapter 3 is thus not used here. Effectively, they were not required to replicate this real database since unforeseen events coming from the environment or resource management were not considered in the real experiment protocol (the resource reloading was done outside the data collection periods).

4.1 Replication of a Real Database

In this section, we detail the real Orange4Home dataset [51] that we propose to replicate as well as the details about the replication of the home, sensors, effectors, and experimental protocol used for this specific dataset. It should be noticed that the results presented in this section are related to only one real database: testing our approach on other databases will thus be necessary in the future to validate the credibility of our model in a robust way. For the moment, this experiment essentially serves as a preliminary test to know whether substituting real data with simulated one is really possible in a specific case.

4.1.1 Introducing Orange4Home

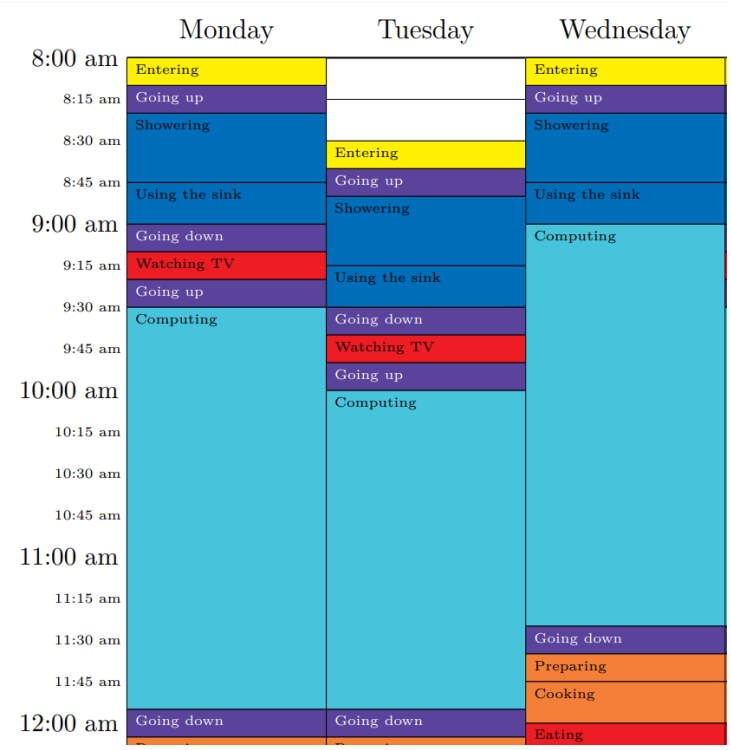

Figure 4.1 – A sample of the input activity calendar that must be followed by the real participant in order to create the real Orange4Home database.

Orange4Home is a database that was created by the Orange company where the initial goal is to test its developed activity recognition and prediction machine-learning algorithms (These algorithms will be the ones used in this chapter). A two-story apartment

was thus made available to one participant to do the experiment for 4 weeks. This person was asked to follow an input activity calendar containing mandatory activities that must be performed at specific times. A sample of this calendar is shown in Figure 4.1. Real data contained in the Orange4Home database were created by the sensors located in the flat and by the effectors triggered by the participant. Concretely, The occupant had a mobile app to indicate when an activity started and when it finished. In addition to the mandatory activities, the *Going to the Toilet* activity, which was not originally scheduled in the initial calendar, can also be indicated by the participant during the experimentation. This unscheduled activity was introduced to generate more diversity in the real data. More details about the creation of this database can be found in the work of Cumin et al. [51] and more explanations about the use of Orange4Home to train the activity prediction and detection algorithms can be found in the next work of the same author [50], [52]. During this thesis, we extended these results by testing our synthetic data on these algorithms in order to compare the obtained results with the previous ones based on real data. This analysis allows us to know if the produced synthetic data can be as efficient as the real ones in our specific case. We can thus check whether our agent model could be used for data generation. Figure 4.2, shows in parallel the occupant performing an activity and our agent performing the same activity.

Orange4Home dataset [51] was chosen in our experiments for multiple reasons:

- Orange4Home is freely open to researchers [1], which improves the reproductibility of our experiments.

- Orange4Home comprises 4 weeks of labeled daily-living activities at home. We will thus judge the credibility of varied activities simulated by our approach on relatively long time scales.

- Orange4Home was recorded in a two-story home comprising 8 different rooms, instrumented with 236 heterogeneous sensors and effectors. Our virtual environment will thus replicate a realistic home setting, and we have a large selection of sensors and effectors to select for replication. These sensors and effectors are detailed in section 4.1.3.

- We worked with the authors who created the Orange4Home database and machine learning algorithms to develop these new results. This enables more accurate

1. See: `https://amiqual4home.inria.fr/orange4home/`

replication of the dataset and thus fairer comparisons between real and simulated data.

Figure 4.2 – Illustration of an occupant and a virtual agent performing the same activities. On the top, the performed activity is Computing, and on the bottom, the performed activity is Cooking

4.1.2 Replication of the Real Environment

The home occupied for the recording of Orange4Home was recreated in our virtual environment, following the plans established for the real apartment, reported in Figure 4.3. This replication also includes most of the furniture, objects, and appliances that the real subject interacted with during Orange4Home's collection. In particular, special care was taken for the replication of elements that are instrumented by sensors or effectors, which will thus impact the quality of data generated when interacting with them. In Figure 4.4, we present a side-by-side view of the living room captured from a similar point of view, in both the real apartment and our simulated replication. Photorealism is

not sought-after since our data are not impacted by this.

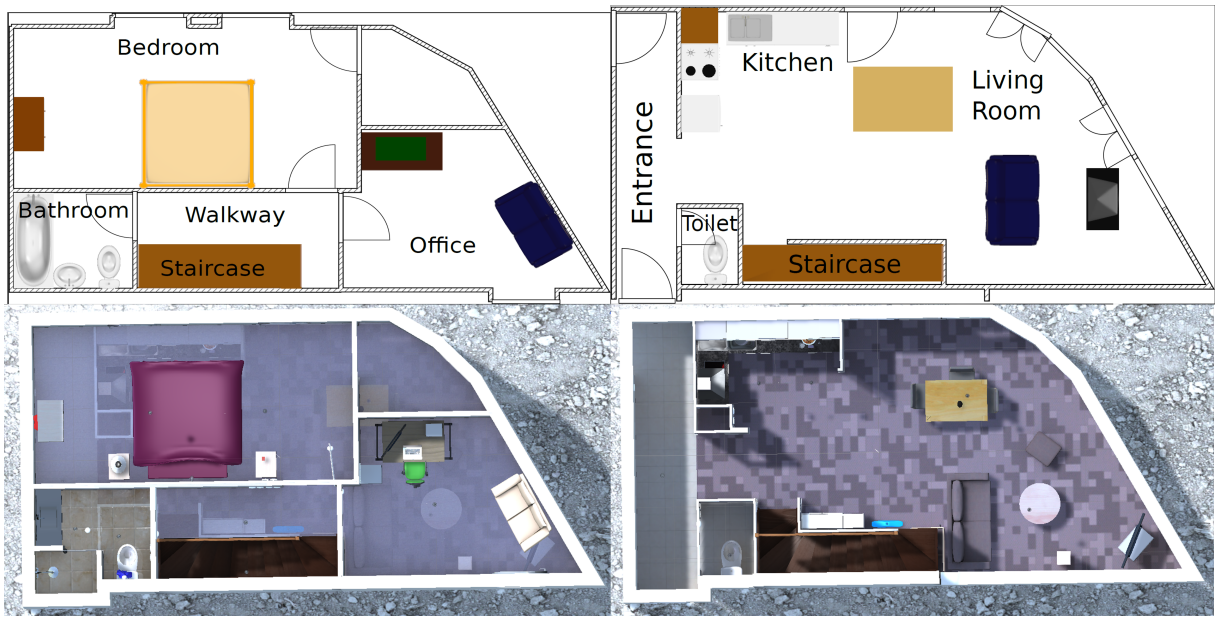

Figure 4.3 – The plan of the ground floor (left) and first floor (right) of the home used in Orange4Home and replicated in the virtual environment shown below.

4.1.3 Replication of Real Sensors and Effectors

Simulating credible data requires that virtual sensors and effectors have credible behaviors. Real sensor data include artifacts, rather than being the result of perfect sensing capabilities. For example, real presence sensors often do not detect seemingly motionless people, whereas a perfect simulated presence sensor would always detect movement regardless of amplitude. Replicating credible behavior thus requires the replication of artifacts and quirks that occur with real sensors. In this section, we present the general principles for simulating sensors and effectors in our VE as well as those used for our validation experiments.

There are in general two ways to replicate credible sensor behavior from real sensor behavior. First, manufacturer data can be used to know how each sensor or effector generally behaves. For example, we can obtain information on the field of view of a presence sensor, the threshold at which a pressure sensor is triggered, or the frequency of data collection. With this data, we can replicate the general behavior of sensors. However, such documentation is not always available for specific device models or versions, or we

Figure 4.4 – Sections of the Office (right corner), Kitchen (top left corner), and living room (bottom left corner), as seen in the real Orange4Home apartment (left part) and in our virtual environment (right part).

sometimes do not know which device model or version we want to specifically replicate. We generally cannot replicate artifacts in data from these specifications.

To overcome this issue, the second way to replicate credible behavior is to observe the behavior of real sensors and effectors in collected datasets. We can then extract behaviors from statistical analysis, which can include their artifacts. This approach requires that there is enough representative data for the device we seek to replicate. Ideally, both approaches should be combined to replicate sensors and effectors behavior with maximum credibility, but this is not always feasible.

Most of the simulated sensors in our VE rely on the analysis of both manufacturer and statistical data. Among these sensors, two categories may be distinguished:

- Sensors notifying when their state changes. Among these sensors, we can find the devices sending a notification when they are turned on or off such as ovens, microwaves, hoods, and so on. We have also door-opening sensors (opened/closed), lights (on/off), and shutters (opened/closed) that send a notification when they are triggered. Finally, we have switches sending a notification when pressed (on) and then a notification when released (off).

- Sensors notifying when their state changes, and then sending a notification of their state with a regular frequency. For example, smart TV sends a notification when turned on or off, and a notification of its state (on/off) every minute as well.

Some sensor behaviors in our VE are based only on statistical analysis, performed on

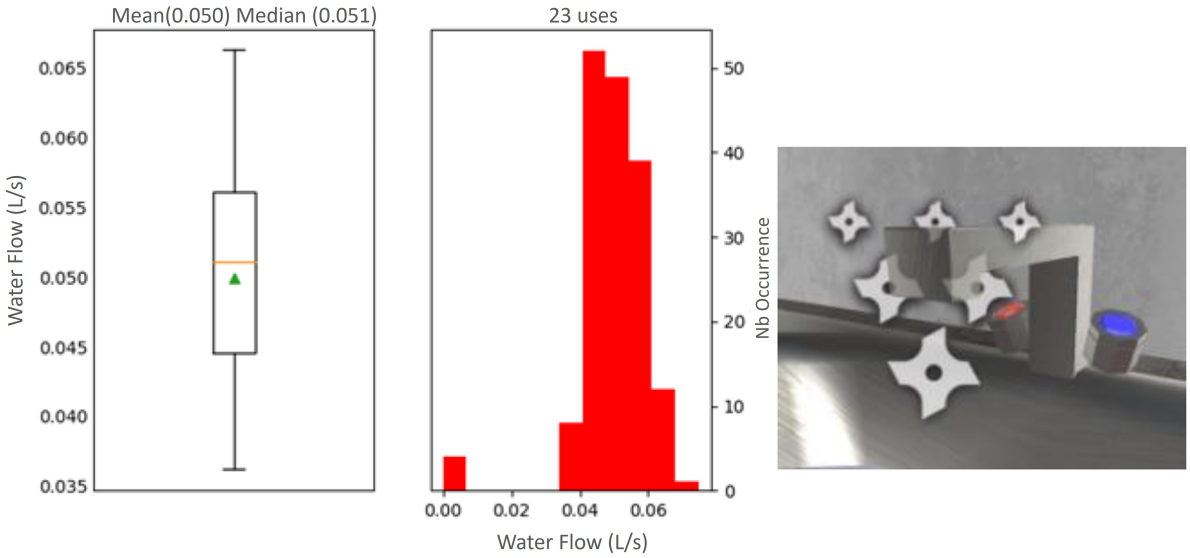

Figure 4.5 – Real data retrieved from the real water sensor placed on the real kitchen tap. On the left, a box plot of the real water flow values; on the middle, the number of real flow occurrences according to their value; on the right, an illustration of the simulated water sensor placed on the virtual kitchen tap.

the Orange4Home dataset [51]:

- Water flow sensors, notifying when their state changes as well as the flow of consumed water at a specific frequency while active. Water flow sensors are used for taps (for both hot and cold water independently) and toilet flushes. Virtual sensors measure water flow based on a Gaussian distribution, parameterized based on the analysis of real water flow sensor behavior, which is illustrated in Figure 4.5.

- Presence sensors notifying when the agent is detected in its field of view. This latter is simulated by Unity *Colliders*[2], detecting when the agent enters into its conical detection area (110°) as shown in Figure 4.6. We observed in real data that these sensors have a 10-second window of blindness where no new notification is sent after detecting a presence. In addition, the person is often undetected after a certain duration of immobility and then detected again. To replicate this behavior, the simulated sensor does not detect the agent if it does not move during a given period of time. After this period, if the agent moves, the sensor detects it again. Otherwise, the sensors detect the immobile agent after a random duration

2. *Unity Collider*: `https://docs.unity3d.com/ScriptReference/Collider.html`

(corresponding to the detection of small movements that occur in real situations).

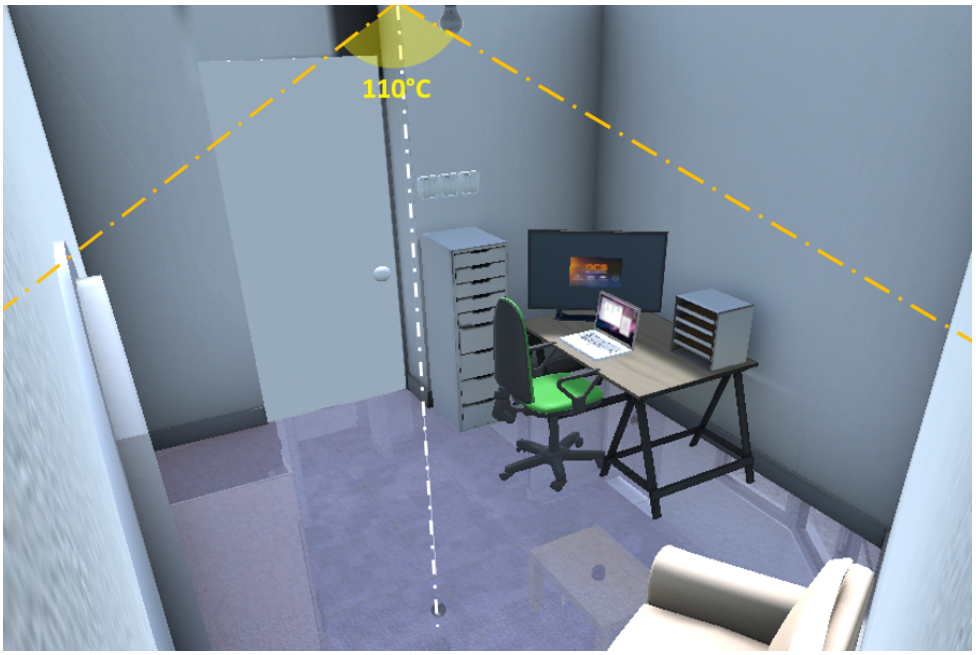

Figure 4.6 – Example of a range detection sensor (110°)

For our experiment, synthetic data are generated from the selection of sensors and effectors we have replicated based on the original Orange4Home experiment, which contains 236 real sensors. In close collaboration with the authors who created the Orange4Home database, we have replicated 63 of these sensors in the VE which seemed to be useful to recognize activities since they are not redundant and are frequently triggered by the agent during activities: lights, door and cupboard openings, presence detection, switches, electricity instantaneous consumption, hot and cold water instantaneous consumption, shutters, and TV statuses. These virtual sensors were placed accurately to their corresponding real equivalent. Functional dependencies for certain sensors were also properly replicated: links between switches and lights, between water consumption and faucets, door opening sensors and doors, and so on. Sensors that were not replicated fall into one of 2 categories:

- Sensors for which we do not have a virtual equivalent yet: temperatures, CO2 levels, weather information, noise levels, etc. Replicating some of these sensors accurately requires specific simulations of environmental variables, which we have

not investigated in this experiment.

- Sensors that correspond to settings, are redundant or seldom triggered: heater settings, total accumulated consumptions, unused switches, unused appliances, etc.

4.1.4 Replication of the Experimental Protocol

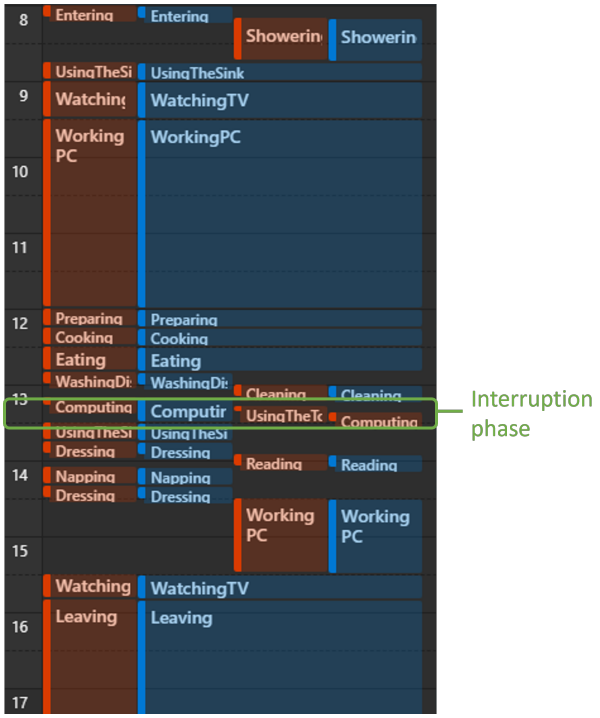

Figure 4.7 – Sample of a one-day Orange4Home calendar that was given in input (Blue one) and performed by the agent (Orange one). An interruption was also triggered by the agent to satisfy *Toilet* need

To match Orange4Home's experimental protocol, we generated 4 weeks of simulated activity data, excluding evenings, nights, and weekends since they were excluded in the real dataset. The agenda that the real participant had to follow in order to create Orange4Home was given as an input calendar in the agent model. Figure 4.7 shows a one-day sample of this calendar that was performed by the agent and whose results are presented in the format of an output calendar. In total, 23 different activity classes were simulated (24 including moments during which no activity occurs, labeled *none*)[1]. The *Cleaning* activity was simulated only in the kitchen, living room, bathroom, and office. Orange4Home activities, sensors, and effectors that were replicated can be found in Table 4.1.

Since the activity schedule of Orange4Home is strict, little room for maneuvering was left to the occupant. He could therefore interrupt an activity to satisfy his own needs such as going to the toilet for example. To have an agent able to reproduce such behaviors, the introduction of interruption mechanisms, when a need becomes urgent, is thus necessary for our use case. The schedule was voluntarily strict to obtain well-structured data for the training of future activity prediction and current activity recognition algorithms. In addition, the first two weeks were very similar between days whereas the last two were very different. With this, the first two weeks can be used as training data and the last two weeks containing less predictable activities can be used to test the efficiency and robustness of these machine-learning algorithms.

4.2 Validation with Activity Recognition

As said before, activity recognition allows us to know if the tasks and interactions executed by the agent are sufficiently coherent and diversified to look like what could happen in real life. With this experiment, we can thus check the credibility of our Task Executor model by verifying whether the agent can interact with enough devices and effectors, and can be detected by enough sensors compared to a real person. We could then deduce if simulated data can be exploited in the same way as real data. More concretely, the comparison between the performances obtained in the case where the detection algorithm is trained with real data and the case where it is trained with simulated ones will allow us to know if the simulated data give enough contents (devices, effectors, triggered sensors) compared to the real ones and thus conclude whether they are exploitable. In parallel, the ability of simulated sensors to capture data in a credible way can also be assessed by checking their occurrence during a specific activity compared to real ones for the same activity.

The activity recognition used in this Chapter to compare the efficiency of synthetic data is based on a Multi-Layer Perceptron (MLP) model which is a category of Artificial Neural Networks. The input to this MLP is a vector of the last reported value of each sensor and effector. For instance, if the agent turned on the TV, the vector will contain this notification with the time of day. In output, we obtain the name of the recognized activity that will be then included in the calculation of the overall performance rate. For instance, if the agent was detected in the bathroom (by the presence sensor) and the shower tap was turned on, the algorithm can deduce that *Showering* is performed. **Our**

goal in this experiment is not to obtain the best performance but rather to get close results between the case where real data are used for training and the case where synthetic data are used. For a fair comparison, we restrict real data to sensors that were replicated in the simulated environment (see Section 4.1.3). Both the real and simulated data thus share the same input feature size.

Table 4.1 – Activities, sensors and effectors implied in the Orange4Home dataset that were replicated

Locations	**Simulated Orange4Home Activities**				**Simulated Sensors and Effectors**
Entrance	Entering	Leaving			Door, Switches, Lights
Kitchen	Preparing	Cooking	Washing Dishes	Cleaning	Sink tap, Switches, Lights, Oven, Hood, Cupboard Doors, Cooktop, Presence
Living Room	Eating	Watching TV	Computing	Cleaning	Presence, TV, Shutters, Switches, Lights
Toilet	Using The Toilets				Switches, Lights, Toilet Flush, Door
Staircase	Going Up	Going Down			Switches, Lights
Office	Computing	Watching TV	Cleaning		Presence, TV, Door, Shutters, Switches, Lights
Bathroom	Using The Sink	Using The Toilets	Showering	Cleaning	Presence, Switches, Lights, Shower tap, Sink tap, Door, Toilet Flush
Bedroom	Dressing	Reading	Napping		Lights, Shutters, Switches, Door, Drawer Doors, Closet Door

Following the experimental protocol set in [52], the first two weeks of data are used for the training phase, the third week as validation data to optimize training parameters, and the fourth week for testing (which are the results reported here), for both real and simulated data. Concretely, the algorithm was first trained with the first three weeks of Orange4Home data, to be then tested on the last week of real and synthetic data. After this, the algorithm was trained with the first three weeks of our synthetic data, to be then tested on the last week of real and synthetic data. These four cases and the obtained performance rate are given in Table 4.2 and studied in the next section.

4.2.1 Recognition Performances on Real and Simulated Data

We report in table 4.2 the recognition accuracy of the MLP on simulated and real data. We see that the model is quite accurate (88.70%) at recognizing activities from real situations when it was trained on real data, with only the selected subset of real sensors also replicated in our VE.

Table 4.2 – Activity recognition accuracy using simulated and real data.

Training data	Test data	Test accuracy
Real	Real	88.70%
Simulated	Simulated	79.40%
Simulated	Real	80.10%
Real	Simulated	55.00%

We see that the recognition accuracy in simulated situations is also high when it was trained on synthetic data, although lower than in real situations (79.40% compared to 88.70%). This shows that virtual activities are globally recognized with simulated training data, even if some are harder to identify than with real instances. Several reasons can explain these difficulties: the agent might interact in simplified movements, triggering less useful sensor events compared to a real occupant. Another possibility is that virtual sensors necessary to identify a specific activity might not be as informative as the real ones.

Comparing performance in different activity classes can help identify which parts of the simulation can be improved. For example, the *Using the sink* activity in the bathroom reaches an accuracy of 37.14% on real data, compared to 3.91% on simulated data. In this example, we noticed that most of the misclassifications concern the *Showering* activity, which occurs in the same place. Therefore, we can assume that some human actions related to both activities are important to distinguish them, and were not integrated into our agent. We can also assume that the replicated bathroom and sensors might not capture all significant information related to these activities.

4.2.2 Using Simulated Data for Real Situations

Ideally, simulated data could reduce the need for labeled datasets collected in real situations to train machine learning models. In this case, simulated data must be representative of real situations, so that a model trained on simulated data can provide

accurate recognition when processing real situations. To illustrate this case, we train our activity recognition model with simulated data by following the same process, and we test it on real data (using the last real week). The resulting model, as reported in table 4.2, reaches a recognition accuracy of 80.10%. These performances, although inferior to the ones obtained when real data are used for training (88.70%), **show that it is possible to reach high recognition accuracy of real situations, from purely simulated training data.**

When the detection algorithm is trained with simulated data, the same detection mistakes are observed when it is tested on simulated data as on real data. For instance, the success rate reached with *Using the Sink* is as low on simulated data as on real data. This shows that simulated data still lack the accuracy required for the algorithm to be effective in detecting some activities. We can also observe that the activities being well detected in the simulated data are also well detected in the real ones. For these activities, the synthetic data are sufficiently rich and accurate to enable the algorithm to identify them. All of this shows that our simulated data are generally usable since most activities are well recognized when the detection algorithm is trained on simulated data. However, some adjustments still need to be made for some activities to improve the success rate and come closer to the one reached with real data.

For completeness sake, we also tried to recognize simulated data with the algorithm trained on real data. We observe a large drop in performances compared to the case where it was trained on simulated data, as reported in Table 4.2 (55.00% vs 79.40%). Activities that were well detected when the model was trained and tested on the same type of data (real and synthetic) exhibit very low accuracies in this case. This global drop is not observed across all activity classes: some well-classified activities in the previous cases stay well-classified. One explanation could be that real data contain richer information than our simulation. Effectively, when we created the task sequences, we asked the authors of Orange4Home to give us the main tasks performed by the participant in the real house. This gave us an overall idea of which main sensors and objects were used in each activity. However, our simulated activities stay approximate since we did not use real videos to create our activities: we just used a list of sensors and objects implied in activities. This inevitably leads to approximations in the simulated data, since some participant behaviors that could trigger additional sensors containing no indispensable information for recognition are not simulated. This could explain why we get a good result when our algorithm is trained on simulated data and tested on real data: information stored in the

simulated data are sufficient to distinguish activities, but we probably do not include all events triggered by a real human. Consequently, when we train the algorithm on real data to test it on simulated data, information that is not essential to detect an activity can become a key selection criterion for the algorithm. For instance, if the location of an object is not the same between real and virtual environments, the participant could take a different path from what the agent would take and thus trigger additional sensors (other presence sensors, additional cupboard opening, etc.). To summarize, the algorithm is more demanding when trained on real data, since it has access to a greater amount of information. More key criteria are thus considered than in the case where synthetic data are used for training: if all the sensors considered as recognition criteria are not triggered in the simulation, the activity performed by the agent will be harder to identify. Conversely, when synthetic data are used for training, fewer key criteria will be found, but all of them will be present in the real data, enabling accurate activity detection.

4.2.3 Comparing Performances on Real Data with all Sensors

In previous experiments on real data, we have limited ourselves to only using the list of sensors that were replicated in the simulation (see Section 4.1.3), thus ensuring fair comparisons to be able to use our model trained on simulated data for real test situations (and vice-versa). However, it can be valuable to compare these performances obtained on real data using all available sensors in Orange4Home. This model reaches a recognition accuracy of 93.20% when it was trained and tested on real data, which is slightly more accurate than the model trained with the limited list of sensors (88.70%), as expected.

We see that the gap in performance is weak, showing that the list of sensors we have chosen to replicate in the simulation provides most of the required information to recognize human activities accurately. This experiment also allows us to identify certain sensors that ought to have been simulated: for example, the model trained with all sensors is very accurate for class *Napping* (99.19%), compared to the model trained with limited sensors (46.13%). This large discrepancy helps us identify *a posteriori* that the bed pressure sensor, which was not replicated, is in fact essential to correctly classify this activity.

4.3 Validation with Future Activity Prediction

In order to judge the credibility of simulated activity routines, we propose to test the use of simulated data for the task of future activity prediction. For this task, we seek to predict the next activity, using previous activities and other high-level contextual information such as place and time. Therefore, the main factor in the quality of simulated data for this task is the agent model. As said at the beginning of this chapter, since this algorithm predicts future activity in relation to activities performed just before, it can identify logical patterns and habits in the choice of activities. Effectively, prediction is based on the assumption that people have daily routines that are repeated over time and can therefore be detected and used to predict what will happen once this routine has begun. Of course, isolated activities that occur in reaction to a situation, or those chosen by default to keep busy cannot be easily predicted. To ensure that the algorithm is able to detect routines, the initial Orange4Home calendar, followed by the participant, deliberately introduces the sequences of activities that are repeated every day to create routines (such as the *Preparing*, *Cooking*, *Eating* sequence). Since the agent follows the same calendar as the real person, it can theoretically produce routines with the same precision. By comparing the performance obtained when the algorithm is trained with simulated data and when it is trained with real data, we can find out whether the agent can choose activities in an accurate and coherent way to give this algorithm enough robustness to predict real routines with similar performances. **These experiments are thus an indicator of the agent model's ability to generate credible activity routines**. The credibility of the agent interactions and the quality of virtual sensors are less important here since we do not use them directly to predict future activities: The data used here are the output calendars containing all the activities performed either by the participant or the agent, with their durations and locations.

To compare behaviors of activity prediction on simulated and real data, we use the model proposed in [52], which was previously evaluated with the Orange4Home real database. This model is a Dynamic Bayesian Network (DBN) in which nodes representing activity, room, hours of the day, and day of the week, are represented for both current and future situations. This model is used to predict the future activity that will occur. Cumin et al. [52] proposed different ways to improve activity prediction. One of those propositions is that sequences of human activities do not satisfy the Markov property. In other words, the choice of a human to perform an activity does not only

depend on their current activity but can also depend on other past activities as well. In our experiment, we will focus on testing different non-Markovian depths for prediction, as in [52]. For example, a non-Markovian depth of 3 indicates that the model uses the current activity as well as the 2 previous activity instances for prediction.

Following the experimental protocol set in [52], we use the first two weeks of data for the training phase, the third week as validation data to optimize training parameters, and the fourth week for testing (which are the results reported here), for both real and simulated data.

4.3.1 Impact of Non-Markovian Depth

Table 4.3 – Prediction accuracy with varying non-Markovian depth on simulated data.

Non-Markovian depth				
1	2	3	4	Average
79.12%	85.71%	89.01%	89.01%	85.71%

Before comparing the obtained performances between simulated and real data, we first study in this section the impact on performance induced by the initial configuration of the algorithm. More specifically, we study this impact when the algorithm is configured with different non-Markovian depths. We thus present in Table 4.3 the prediction accuracy obtained with different non-Markovian depths when the prediction algorithm was trained on simulated data. A depth of 1 implies that only the current activity (with the nodes of place, the hour of the day, and the day of the week) is used to predict the future activity, whereas a depth of 3 implies that the current and previous two activities are used to predict the next activity.

We see that performances are higher with a depth of 3 or 4 (89.01%), and lower with a depth of 1 or 2 (79.12% or 85.71%). This behavior was also observed on 5 real datasets as reported in [52], where multiple datasets reached their highest performance with a depth of 3, except for one dataset (the least predictable) reaching a depth of 1.

These experiments show that the sequences of daily activities generated through our agent model exhibit similar non-Markovian behaviors to those of real datasets reported in previous works [52]. Whatever the data used (real or synthetic), the past sequence of 3 activities provides the most information on which next activity will occur for this dataset. Only using 1 past activity is generally not informative

enough, whereas using sequences longer than 5 activities introduce noise and unneeded complexity to our predictive model.

4.3.2 Predicting Performances on Real and Simulated Data

After studying the impact of different non-Markovian depths, we can now compare the performance obtained when the algorithm is trained with simulated data and when it is with real ones. To do this, we choose a depth of 3, which is generally the most effective as detailed in the previous section. We report in Table 4.4 the prediction accuracy of this model on simulated and real data. We see that real data from Orange4Home is quite predictable (91.18%) since the occupant followed a strict schedule of activities. To evaluate the efficiency of simulated data, two cases were examined: the case where the agent had the right to interrupt an activity to satisfy the *Toilet* need, and the case where it could not. The first case is interesting since it enables us to introduce variability into the data in a controllable way to make the predictions more robust. In addition, the agent is placed in the same situation as the participant who had the right to interrupt activities to go to the toilet. For the second case, where the agent cannot interrupt, we wanted to know whether the simulated data could already provide high prediction performance when the agent is not disturbed by an unexpected event. Effectively, these events will automatically reduce the level of prediction since they are not linked to routines but rather to a reaction caused by a sudden change. According to Table 4.4, when our agent cannot interrupt activities, this leads to very predictable activities (91.86%). This first result thus indicates that the agent model is able to reproduce routines with similar levels of predictability compared to real data.

We also obtain slightly less predictable activity sequences when our agent model enables interruptions (89.01%). This performance rate thus remains close to the one reached with real data. Such interruptions, allowing the satisfaction of needs when there is not sufficient free time, can thus be used to introduce variability in generated data, in a controlled manner. In this way, our agent model could generate different datasets of the same environment using different degrees of activity predictability, which is valuable to design and improve such predictive models. Prediction accuracies obtained in those first experiments are all significantly higher than a random predictor (5%, assuming identical predictive randomness across the 20 activities).

Table 4.4 – Activity prediction accuracy using simulated and real data.

Training data	Test data	Test accuracy
Real	Real	91.18%[3]
Simulated	Simulated	89.01%[3]
Simulated, without interruptions	Simulated, without interruptions	91.86%[3]
Simulated	Real	82.35%[3]
Real	Simulated	82.42%[3]

Non-Markovian depths used for each reported result are indicated with superscripts.

4.3.3 Using Simulated Data for Real Situations

Ideally, simulated data could reduce the need for labeled datasets collected in real situations to train machine learning models. In this case, simulated data must be representative of real situations, so that a model trained on simulated data can provide accurate predictions when processing real situations.

To illustrate this case, we trained the activity prediction model on simulated data (following the same protocol used previously), and tested it on real data. The resulting model reaches a predictive accuracy of 82.35% (with a non-Markovian depth of 3) as shown in Table 4.4. Even though it is lower than a model trained directly on real data (91.18%), these performances are still very high and vastly superior compared to a random model, **indicating that simulated data can thus be very useful to train these kinds of models.** This gap between performances may be due to several factors. Firstly, the person did not interrupt the activities at the same time that our agent (and often the real interrupted activities were different from the simulated ones). The location of *Cleaning* sometimes changed and was not necessarily the same for the agent and the person at the same time. Finally, since activity duration was also an input parameter for the algorithm, it could be used as a prediction criterion. Consequently, even if most of the durations are close, time differences are sometimes observed. Effectively, the person sometimes shortened some activities (such as *Showering*, which was sometimes shortened by 20 minutes) and lengthened others to compensate (such as Working, which was lengthened by 1 hour). The agent, on the other hand, adhered strictly to the times indicated in the schedule, leading to these gaps. It should be noted that in this experiment, we

wanted the agent to strictly respect the duration of mandatory activities. However, our model is able to apply variations in the duration of mandatory activities. For instance, we could change the duration or start time of the next mandatory activity before launching the scheduler. The planner could then be executed normally, with just a different time budget.

For completeness sake, we also trained our activity prediction model on real data and tested it on simulated data. The resulting model reached a predictive accuracy of 82.42% (with a non-Markovian depth of 3) as shown in Table 4.4. There are still performance gaps to a model trained on simulated data (89.01%), **but performances are again far beyond random prediction, which reaffirms that simulated data are in some way representative of real data.** The gap in performance found here is due to the same reasons as the case explained just above.

4.4 Conclusion and Discussions

In this Chapter, we propose a validation approach to better address our Challenge 3 regarding the way to validate the credibility of generated behaviors and data. For this purpose, we replicated a real collection experiment in a virtual home, using our agent model in place of real occupants. The real database used for replication purposes is the Orange4Home database proposed by Cumin et al. [51]. The credibility of synthetic data was then evaluated in two human understanding tasks: Activity Detection and Activity Prediction. To do this, we used two machine learning algorithms specialized in these both tasks to compare their performance rate between the case where they are trained on real data and the case where they are trained on synthetic ones. The obtained results show that these rates are relatively close, despite some inaccuracies, demonstrating that it is possible to use synthetic data in place of real data without strongly dropping performance in at least this specific case.

4.4.1 Insights on replicating Datasets

In this chapter, activity recognition and prediction models were used as indirect measures to assess the credibility of simulated data and agent behaviors. Our focus on these human understanding fields has multiple justifications. First, we wanted to evaluate the interactions made by the agent to know if it can execute activities in a credible way. In

addition, we wanted to have a way to evaluate the credibility of the sensors capturing data. Activity recognition is a task that encompasses those elements and is thus suitable for this evaluation. Second, we wanted to evaluate the credibility of activity routines generated by our agent model. Future activity prediction allows comparisons between the predictability of real and simulated routines. It is thus suitable for this evaluation. We have shown in this chapter that data generated by our simulation can be effectively used as training data for activity recognition and prediction tasks applied to real situations in Orange4Home. The results obtained in our experiments are convincing, **showing that using simulated data as training sets to counterbalance the lack of real data is a promising research avenue.** Our experiments also allowed us to identify the improvements we could make to refine the results.

Among the improvement to make, some specific activity classes are not well classified when using simulated training data, compared to real data. This indicates that credibility discrepancies exist, depending on which activity class is simulated. One source of discrepancies we have discussed in Section 4.1.3 is the actions of the agent which can be not enough accurate compared to humans. Concretely, since we simulated activities in a generic way, some tasks or interactions that could trigger additional sensors were sometimes not simulated. When these missing sensors are essential to correctly detect activity, the performance rate is impacted in all the studied cases. Further analysis is therefore required to find out which important interaction was missing. When the missing sensors are not indispensable to correctly detect activity, some limitations can be observed in the case where the algorithm, trained on real data, tries to identify simulated activities. Effectively, it has access to more information during its training and becomes more demanding in its classification criteria. Regarding activity prediction, the differences between the moments of interruption, the durations, or the start times have an impact on performance. **To summarize, these experiment allows us to identify which parts of the simulation model are not credible enough and need to be improved.** Another source of discrepancies can be the behavior of virtual sensors. Our efforts on replicating realistic behaviors of sensors have helped limit the drop in performances, but not completely remove it. In the general case, we might not have access to the behavior of all sensors of the environment we are trying to replicate. Limiting such discrepancies would thus prove difficult. This experiment also allows us to know which sensors and effectors are actually essential to correctly detect some activities.

In future work, all this information can be used to enrich our agent and our simulation.

Regarding the agent, this allows us to detect missing interactions that are essential to detect activities. We can also see what limitations exist in the choice of activities to produce more credible routines. Regarding the simulation, this information can be used to enrich our simulation with missing sensors. Conversely, we could use our replication strategy to check the importance of different sensors (measured by the drops in performances), for applications aiming at reducing the number of redundant sensors in smart environments.

4.4.2 Generalization to Other Contexts

Our credibility validation approach, which uses activity recognition and prediction models, has only been tested on the Orange4Home dataset. As it stands, we cannot extrapolate our results to other datasets or generalize our observations to any smart environment. It is required to test this simulation and validation approaches in various other contexts that offer accurate real data for comparison.

The validation methods presented in this chapter are a further step to address Challenge 3 concerning the way to validate the credibility of simulated human behaviors. The results reported in this chapter are promising and show that simulated data could be used in place of real data without a drastic loss in algorithm performance, for at least the specific cases presented here. These results also reinforce the ones presented in Chapter 2, where an input-output comparison of the agent model was implemented. In addition, since we have shown that simulated data could substitute real ones in at least one case, this also shows that our agent model can be compatible with data generation and could produce sufficient credible behaviors to be exploitable. However, testing our solution against many more varied datasets stay required to ensure that generated behaviors and data will be credible enough to be exploitable in any virtual environment and situation.

Chapter 5

Conclusion

During this thesis, we address the simulation of virtual humans compatible with daily activity data generation. To reach this objective, we explained that three challenges must be addressed. The first one involves proposing a human behavior model able to generate credible behaviors. This implies having an autonomous virtual human producing behaviors that are coherent, consistent, and diversified. The second one involves proposing a model providing a compromise between control and autonomy in order to be compatible with data generation. Finally, the last one involves implementing validation methods to check the credibility of the generated behaviors and synthetic data. These validations are effectively essential to know whether synthetic data can replace real ones for various human understanding tasks.

In an age where collecting data has become a major challenge in many domains, we hope that proposing new approaches able to generate exploitable synthetic data could enrich existing real databases, create new ones for situations that are difficult to achieve in real life, and also limit the collection of data coming from real users. In parallel, addressing Challenge 2 led us to develop an approach having the advantage to be controllable, configurable, and transparent in its functioning. This goes against the flow of the new generation of machine learning algorithms, where it becomes difficult to control output behaviors and to know how these behaviors were obtained. In addition, our approach has the advantage of not relying on any data to operate. With these advantages, we can address several use cases: we have shown in this thesis that our agent model can be applied for data generation, but it can also be used for other purposes, such as populating environments.

Contributions

In Chapter 1, we presented the existing works related to agent behaviors models and data generation. We first explored approaches simulating humans to generate daily activ-

ity data. We stated that agent-based models were the most suited to address the challenges stated above. We also concluded that 3D environments are well-suited to simulate any sensors capturing data (particularly environment-dependent ones), but also to produce any daily activity data, and achieve greater granularity in the simulated activities. In the second part of our related works, we explored the existing agent models, the ways to interact with virtual environments, and the existing validation methods. We finally noticed that among existing agent models, some limitations were observed to address our three challenges at the same time.

In Chapter 2, we thus proposed an agent model to address our three challenges. This model, inspired by the BDI architecture that was enriched with a reactive scheduler, is made of the following parts: The **External Perception Model** to retrieve information about the environment, the **Internal State Model** to manage the autonomous part by simulating the agent motivations coming from needs (Hunger, Thirst, etc.) and preferences, the **Decision-Making Model** to make a compromise between control and autonomy in behaviors, and finally, the **Task Executor Model** to execute activities in a 3D environment and allow their interruptions. In addition to these four systems, a module called **Agent Parameters** enables users to configure the model, such as providing an input activity calendar to impose activities at a specific time. In the Decision-Making model, we propose a reactive scheduler to create activity plans according to constraints, and an activity selector to select the activities of the plan while synchronizing the scheduler with the rest of the system. A validation method was also proposed to check whether our model can respect the input constraints and the agent needs. The results were encouraging and showed that our model can cover a part of our use case and address Challenges 1 and 2. With this version, users can effectively use our agent model to make one-person simulations and generate, replicate, or complete databases related to these situations.

In Chapter 3, we proposed an extension of this model to better address Dynamic Environments, our use case, and Challenge 1. Effectively, the management of such environments, which can change independently of agent actions, allows users to obtain more diversified data and simulate several agents simultaneously. To manage Dynamic Environments, a new model made of three sub-models is proposed to manage resources and unexpected events at scheduling and execution levels. Among these sub-model, we have **The Resource Manager** to anticipate resources during the plan creation, **the Interruption Manager** to interrupt the executed activity when a missing resource or an event suddenly happens, and **the Plan Checker** to verify if interruptions caused by missing

resources or unexpected events impact the rest of the plan. We these sub-models, the produced activity plans are coherent with the constraints of time, needs, and now resources. In addition, the plan can be revised to know whether the remaining activities can still be performed after an interruption. The agent can also interrupt the current executed activity when an unexpected event happens or a resource is missing. With this process, the agent can take the necessary actions to stop the activity and be ready to perform another one. In addition, it can also perform the required actions to resume interrupted activity if needed. A preliminary version of a user-experiment protocol was also proposed to validate the proposed model. With this new version, users can exploit our agent model to simulate multi-person situations or situations where unexpected changes happen in the Environment. They could also generate, replicate, or complete databases related to these situations.

In Chapter 4, we proposed a validation method to better address Challenge 3 and to know whether our agent model can produce synthetic data sufficiently credible to substitute real data, at least for activity detection and prediction in a specific context. For this purpose, we replicated the real Orange4Home dataset by using a digital twin of the real environment that is composed of the apartment geometry and the exact positions of its sensors and effectors. We used our virtual agent model to simulate an occupant, and we followed the same experimentation protocol used for the creation of Orange4Home. We then tested our synthetic data on the training of two machine learning algorithms used for human-understanding tasks: Activity Recognition and Activity Prediction. We compared their performance rates obtained with the case when they are trained on real data. The obtained results are promising and allow us to know that simulated data could be used instead of real data, at least in a specific case. However further validations must be set up to know whether our model can produce synthetic data exploitable in more global cases.

Perspectives

Short Term Perspectives

Regarding our future work in the short-term, several axes could be explored:

Model Completion and Authoring Tools: Since we want to validate our system in other use cases (such as buildings or factories) and propose an agent model ready for use by users coming from various fields, our system will be confronted with new types of environments and new activities. In parallel, we also want to increase the number of activities and tasks to enable the agent to perform more diversified actions. Consequently, designing authoring tools will be essential to facilitate the initial configuration of the agent, as well as the creation of activities and animations. Regarding activities authoring, offering this kind of tool would enable users to create their own animations without needing any knowledge of the field. In addition, we could quickly obtain specific animation sequences that are not available in animation banks. To allow this, it could be interesting to propose a VR tool to create the animations and activities. Some existing approaches such as the work of Lécuyer et al. [115] propose interesting methods to create VR scenarios without using any code. With this approach, we could yet produce the sequence of tasks contained in the future activity. However, animations contained in each task must be yet implemented. This is why, we could go further by proposing methods inspired by *Inverse Kinematics*[1] to track the users' controllers and headsets and construct the animation performed during a new interaction. In addition, visual feedback could be used to allow users to observe their new animations or even their new activity by seeing in VR the agent performing it. They could then fix some parts by doing again the concerned animations. Regarding the agent configuration, a user interface could be developed to configure each agent in an accurate way. Existing approaches such as SMACH [156] and the work of Bogdanovich et al. [23] propose some interesting possibilities to configure several agents at the same time, such as using Genetic Algorithms to automatically generate diverse crowds or the use of interfaces to configure each agent.

Additional Validation Methods: We also would like to set up an Input-Output comparison method to check whether the output of the model detailed in Chapter 3 is coherent with the state of the environment (missing resources, occurred events, etc.) and the initial agent parameters. We could thus assess whether our model can adapt to unexpected changes caused by Dynamic Environments while respecting users' constraints as soon as possible. In addition, we would like to apply the user experimentation protocol proposed in Chapter 3 to make an additional validation. With this, Challenge 3 will be better addressed since the case of a Dynamic Environment will be considered. We also want to

1. *Inverse Kinematics*: `https://docs.unity3d.com/Manual/InverseKinematics.html`

improve the results obtained in Chapter 4 by deeply analyzing the activities that were not well predicted and detected to know which interactions and sensors must be added or improved. To fully address Challenge 3, extending our validation to other databases, environments, and contexts is essential. We thus would like to replicate other databases and their validation methods to know if our agent model can be used in different use cases to generate synthetic data credible enough to replace real ones. We could start with the real *CASAS*[2] dataset since experimentation was yet performed by Cumin et al. [52] to assess the performance rate of their Activity prediction and Activity detection algorithms that are already used in Chapter 4.

Generating more credible behaviors by simulating needs Interdependence: We also would like to go further with the needs by proposing a greater interdependence between them. For example, satisfying *Thirst* may also affect *Toilets*, satisfying *Sports* may affect *Tiredness*, and so on. This interdependence will allow us to improve the credibility of generated behaviors and thus better address our Challenge 1. Considering these new aspects will have an impact on our scheduler since needs will be impacted by two parameters: the time of day and the effects of other needs. Consequently, their moment of urgency will be more difficult to anticipate and the method used to calculate the level of their urgency will therefore have to evolve. One possibility would be to use a confidence interval regarding the evolution of needs. Concretely, the scheduler will consider the worst-case scenario to optimize the chances of planning correctly. If, despite this, the need reaches its maximum level of urgency at another time, an event is triggered. Another solution would be that satisfied needs would increase or decrease the value of other needs by a fixed percentage (for example, *Doing Sports* would increase *Tiredness* by 10%). By using a constant effect, urgent needs can still be anticipated by requiring the scheduler to not delete any scheduled activity satisfying needs. Starting with the urgent need closest in time, the scheduler will gradually readjust the need urgency each time they are impacted by the schedule of a new activity.

Generating more diverse behaviors through preferences and emotions: We finally would like to test more precisely the impacts of preferences over the agent model and develop some other cognitive processes such as emotions or personality to diversify behaviors. Effectively, some preferences have been developed to impact the choice between

2. *CASAS dataset*: `https://casas.wsu.edu/datasets/`

two activities satisfying the same goal (to satisfy *Hobbies*, the agent could prefer to choose *Watching TV* rather than *Reading*) or between the different ways to perform an activity (between *Eating a Sandwich* or *Eating Pasta* for instance). To avoid the systematic selection of the preferred choice over others, the agent randomly chooses one of them by putting a stronger weight on the preferred one. However, testing the impact of these preferences must be carried out to ensure that they effectively influence the final plan. For other cognitive processes, further studies must be done to determine which emotions and personalities are relevant to simulate in the case of daily life. Regarding emotions, they could be treated as unexpected events: for example, if the agent is stressed, some effects could happen such as dropping objects, forgetting mandatory activities given by users, or performing activities incorrectly (e.g. sleeping badly), and therefore not fully achieving what was planned. However, such changes could have a major impact on the scheduler and the control over the agent behaviors. Consequently, further thoughts will have to be done to ensure a certain level of control.

Generating more credible and diverse behaviors through environment and social factors: We would like to implement new kinds of needs that can be impacted by the environmental state such as Temperature, Humidity, and so on. With such features, the agent will be sensitive to environmental changes. Considering these needs are essential for users wanting to observe the impacts of the environment on human behaviors, by retrieving data about energy consumption for instance. In contrast to precedent needs, they will have two tolerance thresholds bounding a comfort zone [159]. The goal of the agent would thus be to stay inside this zone by performing activities either increasing the value or decreasing it. In addition, they do not evolve according to time but according to the environment. Since they are unpredictable, one solution to integrate them without impacting the scheduler is to consider them as events when they become urgent. A second kind of need will be influenced by social factors. Social needs could be used to motivate the agent to interact with others when the need is urgent. As with the previous ones, they have a comfort zone to target. Above this zone, the agent needs to talk with others or participate in collaborative activities. Below this zone, the agent needs to stay alone and will refuse any solicitation made by others. These needs increase over time and decrease when social activity is performed: they can thus be scheduled as other needs. With this type of requirement, we could trigger interactions between agents and offer them the possibility of accepting or rejecting proposals.

Long Term Perspectives

Regarding our future work in the long-term, we would like to integrate deeper changes:

Generating other kinds of data and exploring other use cases: Our credibility evaluation approach could be extended to a large array of different human context analysis tasks. For example, tasks such as room occupation detection or occupant positioning could be used as another way to evaluate the credibility of synthetic data. By adding virtual cameras in the simulation, we could retrieve videos to train visualization-based algorithms such as Recognition algorithms in order to check the efficiency of our synthetic data, as shown by Puig et al. with Virtual Home [144]. However, a great improvement in the agent animations would be needed to avoid mistakes in activity recognition.

Developing multi-agent simulations including collaboration and cooperation: We would like to implement collaborative activities between agents to allow them to interact with each other and share activities. Developing multi-agent simulations are useful for generating multi-person data since real activity data involving several people is rarer than those involving a single person. To synchronize agents, we could develop some activities triggered by social needs (such as *Talking with others*) to plan future collaborative activities (such as eating together, going out at a specific time, etc.). These collaborative activities would then be scheduled as mandatory activities to be sure to be performed. The agents could also negotiate between them to find a timeslot suiting their needs and imperatives. In addition, having a multi-agent model will be useful for users wishing to simulate larger environments (city, buildings, etc.) to either generate data, populate them or enhance their existing simulators. After making our model compatible with multi-agent, we would like to validate the generated multi-person data with a similar process made in Chapter 4. Concretely, we will compare the performance obtained when the same algorithms are trained on synthetic multi-person data with those obtained when trained on real ones. As multi-person real databases, *CASAS* could be used since some data concern several people. Additional results could be thus obtained regarding the credibility of behaviors when several agents are implied to better address Challenge 3.

Human-Agent interaction: As an ultimate test of the robustness of our model facing unexpected events, we could go further than multi-agent situations by testing our model in a situation where the agent performs collaborative activities with a real human controlling

an avatar. Our agent will thus be aware of the existence of this avatar. We could also make an experiment where participants in VR can impact the environment to disrupt the agent, such as removing resources, changing the temperature, or triggering events. This would allow us to finalize the validation of our agent model in the case of Dynamic Environments. To go further, this kind of collaboration opens up new perspectives for users wishing to populate environments since humans and agents sometimes share the same environment in these use cases.

Checking the impact of bias on credibility evaluation: Finally, regarding the method used for validation purposes, we believe that the use of machine learning models specialized in human understanding is a promising avenue of research to assess synthetic data. We also believe that designing meaningful yet non-biased credibility evaluations for simulated data is difficult to achieve. Instead, further work should focus on evaluating the impact of such biases on credibility evaluation. To do this, some checking approaches could be developed such as cross validations [17] to determine whether the machine learning algorithms used for our validation could induce *Overfitting* (i.e. the algorithm will have too great capacities to capture information, and will have difficulties generalizing and obtaining good performances on new data: even a small fluctuation or noise in the trained data will be considered an important feature) or *Underfitting* (i.e. the algorithm will not be able to capture enough information to offer good performance, even with training data.). We could also assess whether combining real data with synthetic ones during the training phase could reduce these biases, by assuming that simulated data could provide more diversity in the input ones.

PUBLISHED PAPERS

Articles in Peer-Reviewed Conference Workshops

L. Gramoli, J. Lacoche, A. Foulonneau, V. Gouranton, and B. Arnaldi. Needs Model for an Autonomous Agent during Long-term Simulations. *In the Modeling and Animating Realistic Crowds and Humans (MARCH) Workshop, proposed by 2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR) (pp. 134-138).* **2021, November**.

Articles in Peer-Reviewed Conference Proceedings

L. Gramoli, J. Lacoche, A. Foulonneau, V. Gouranton, and B. Arnaldi. Control Your Virtual Agent in its Daily-activities for Long Periods. *In Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection: 20th International Conference, PAAMS 2022, L'Aquila, Italy, July 13–15, 2022, Proceedings (pp. 203-216). Cham: Springer International Publishing.* **2022, October**.

Articles in Submission

L. Gramoli, J. Cumin, J. Lacoche, A. Foulonneau, V. Gouranton, and B. Arnaldi. Generating and Evaluating Virtual Data of Daily Activities with an Autonomous Agent in a Smart Home. *Submitted to the Special Issue on Realistic Synthetic Data: Generation, Learning, Evaluation, proposed by the ACM Journal on Transactions on Multimedia Computing, Communications, and Applications (ACM TOMM). (In Review)*

Articles in Preparation

L. Gramoli, J. Lacoche, A. Foulonneau, V. Gouranton, and B. Arnaldi. Managing Resources and Events Coming from a Dynamic Environment at Scheduling and Execution

Levels. *(In preparation)*

Other Articles

A. Cheymol, G. Fouché, **L. Gramoli**, Y., Hirao, E., Hummel, M., Mavromatis, Y. Moullec, F. Argelaguet, and F. Nouviale. The rubber slider metaphor: Visualisation of temporal and geolocated data. *In 2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW) (pp. 904-905). IEEE.* **2022, March**

Résumé Etendu en Français

Introduction

Comprendre le comportement humain est essentiel pour développer des systèmes autonomes et adaptatifs tels que des environnements connectés. Pour parvenir à cela, des données réelles sont alors nécessaires, mais elles sont rares et coûteuses à produire. Une solution consisterait à générer des données synthétiques en simulant les environnements et leurs occupants. Cependant, nous devons nous assurer qu'elles sont suffisamment crédibles pour être utilisable et cette crédibilité va notamment dépendre de celle des humains simulés et de leur capacité à interagir avec des environnements virtuels pouvant produire de telles données. Cette approche soulève alors la question suivante: **Comment simuler des humains virtuels capables de produire des comportements crédibles tout en étant compatible avec la génération de données ?** Être compatible avec la génération de données signifie que l'agent doit avoir la capacité d'interagir avec des environnements virtuels qui peuvent générer des données simulées en capturant leurs actions. De plus, pour vérifier la crédibilité des données et des comportements, des méthodes robustes de validation devront être appliquées. Pour répondre à cette problématique, nous proposons dans nos travaux de thèse un modèle d'agent autonome capable de fournir des comportements à la fois autonomes et contrôlables tout en permettant l'exécution d'activités dans des environnements 3D. Notre modèle s'adressera aux personnes (chercheurs, développeurs, etc.) qui souhaiteraient utiliser des humains virtuels pour générer des données synthétiques, peupler des environnements ou améliorer les performances de leurs algorithmes déjà existants. Nous nommerons ces personnes *Utilisateur* tout au long de cette thèse.

Pour tester l'efficacité de notre modèle d'agent et voir s'il est compatible avec la génération de données, nous allons utiliser des simulateurs 3D comportant des environnements connectées dotés de capteurs capables de générer des données en capturant leurs actions. Les capteurs à l'intérieur de ces environnements sont non-intrusifs car ce ne sont pas des micros ou des caméras. Ce sont plutôt des capteurs qui récupèrent des données sur le bruit ambiant, les mouvements, des débits d'eau, les ouvertures de portes ou de placard,

ou encore l'activation ou non d'un interrupteur. Figure 5.1 montre un exemple d'une de nos simulation 3D où un agent autonome exécute des activités quotidiennes comme *Manger* ou *Travailler*. L'agent est autonome dans sa prise de décision et dans l'exécution de ses activités. De plus, il peut être configuré par les utilisateurs à l'initialisation pour générer les comportements souhaités. Une interface montrant l'état des besoins de l'agent (faim, soif, etc.) est aussi indiquée. La date et l'heure exacte de la simulation est aussi affichée et peut aussi être modifiée initialement.

Figure 5.1 – Vue d'ensemble de notre simulateur 3D et de l'agent perfomant des activités

Contexte Industriel

Cette thèse a été réalisée en partenariant avec *Orange*[3], une entreprise française majeure spécialisées dans les télécommunications, réseaux, et services numériques. Plus récemment, Orange propose de nouveaux services en lien avec les environnements connectées. Pour étudier et simuler ces services, Orange a développé de nombreux simulateurs et plateformes, dont certaines d'entre elles ayant des besoins liés à notre problématique sont transverses à notre thèse. Notre thèse quant à elle fait partie d'un des projets interne qui étudie le potentiel des jumeaux numériques pour les futurs services de l'entreprise.

Afin de pouvoir améliorer ses services, Orange avait besoin de connaissances sur les comportements humains dans la vie quotidienne. Les chercheurs avaient donc développé

3. *site web Orange*: `https://www.orange.fr/portail`

des algorithmes d'apprentissage automatique spécialisés dans la compréhension du contexte humain. Cependant, puisque ces algorithmes avaient besoin d'une quantité importantes de données labellisées et que les données réelles étaient rares, coûteuses et pouvaient poser des problèmes de vie privées, l'entreprise a souhaité explorer la génération de données synthétiques. Des environnements 3D ainsi que des humains virtuels ont été conçus par l'entreprise pour répondre à ces besoins de nouvelles données. En effet, les environnements 3D ont l'avantage d'offrir une granularité plus fine des activités réalisées mais aussi de permettre la simulation de certains capteurs qui sont dépendants de l'environnement, comme des capteurs de présence. Figure 5.2 donne un exemple de ces environnements 3D qui ont été créés par les chercheurs d'Orange. Cependant, ces humains virtuels qui étaient fournis avaient des comportements prédéfinis étaient très limités par rapport à ce qu'un humain pourrait faire. C'est dans le but d'améliorer le comportement de ces humains virtuels que la thèse a été conçue.

Figure 5.2 – Exemples d'environnements virtuels 3D crées par les chercheurs d'Orange afin de générer des données synthétiques. Les deux images situées en haut à gauche représentent les deux étages d'un même appartement virtuel répliqué depuis celui d'un réel. Les maisons intelligentes présentés en haut à droite et en bas à gauche sont fictives. En bas à droite, une image d'un bâtiment intelligent répliqué depuis un bâtiment réel.

Autres contextes d'application

Les humains virtuels sont aussi importants pour d'autres domaines de recherches et d'autres cas d'usage que le contexte de l'entreprise. En effet, ils peuvent servir pour d'autres domaines de recherche souhaitant générer des données d'activité quotidiennes. comme le bâtiment et l'énergie qui souhaitent récupérer des données sur la consommation énergétique des foyers. Les chercheurs se penchant sur le changement climatique pourraient aussi utiliser les humains virtuels pour tester certaines situations. Ils peuvent aussi être utilisés pour peupler des environnements comme c'est le cas dans le domaine du jeu vidéo, de l'héritage culturel ou encore la formation. Au sein du laboratoire Inria/IRISA où la thèse se déroule, les humains virtuels sont utiles pour peupler des environnements de Réalité Virtuelle (RV) afin de faire des applications de formations mais aussi pour simuler des foules ayant des comportements individuels.

Enjeux

Afin d'obtenir des humains virtuels produisant des comportements crédibles tout en étant compatible avec la génération de données, et ainsi répondre à notre problématique, 3 défis majeurs doivent être adressés:

Défi 1: Produire automatiquement des comportements crédible: D'après les travaux d'Avridinis et al. [11], la crédibilité peut être atteinte en permettant à l'agent d'être autonome dans ses choix et ses objectifs et en respectant deux critères: (1) **La cohérence** des réactions de l'agent et de ses motivations, (2) **La consistance** des comportements face à une situation similaire. En plus de ces deux points, nous ajoutons aussi **la diversité** des comportements générées afin que l'agent ne produise pas des routines identiques et n'exécute pas toujours les activités de la même manière.

Défi 2: Offrir un compromis entre contrôle et autonomie: Afin que nos agents virtuels puissent être utilisables pour la génération de données, il est important que d'avoir un modèle offrant un compromis entre contrôle et autonomie. Le respect de contraintes fortes peuvent alors rivaliser avec les objectifs propres à l'agent (comme les besoins physiologiques). C'est pourquoi, il sera important d'avoir un modèle d'agent capable d'anticiper les contraintes futures pour éviter les conflits. Ces contraintes fortes proviendront des utilisateurs qui pourraient vouloir configurer notre modèle afin de l'adapter à leurs exigences

ou aux protocoles qui peuvent régir la création de certaines base de données. Ces protocoles pourraient par exemple contenir un calendrier d'activité à faire à des heures précises.

Défi 3: Valider la crédibilité des comportements et des données générées: Afin de s'assurer que notre modèle d'agent produise des comportements suffisamment crédible afin que les données synthétiques puissent être exploitables, il est important de proposer plusieurs méthodes de validation. Ces validations peuvent évaluer la crédibilité des comportements et des données générées.

Contribution

Bien qu'il existe des approches de la littérature pouvant atteindre un de nos 3 défis, aucune d'entre elles peut gérer les trois en même temps. C'est pourquoi, nous proposons dans cette thèse un modèle d'agent spécifiquement conçu pour gérer simultanément ces défis. Pour cela, nous proposons un modèles d'agent capable d'offrir un compromis entre contrôle et autonomie tout en gardant des comportements crédibles (Défis 1 et 2). De plus, plusieurs méthodes de validation seront proposées pour vérifier la crédibilité des comportements et des données (Défi 3).

Chapitre 1: Etat de l'Art

Au sein de l'état de l'art, il existe tout d'abord des approches qui utilisent des simulateurs avec des humains virtuels afin de créer des données synthétiques. En étudiant tout d'abord les environnement simulés proposées par les approches existantes, nous constatons que les environnements 3D sont les plus précis pour illustrer un environnement connecté car certains capteurs peuvent être sensibles aux environnements (comme les caméras, les capteurs de présence, etc.). Du côté des humain virtuels utilisés pour la génération de données, trois catégories peuvent être relevées. La première approche est basées sur les avatars où un véritable humain contrôle l'agent. La seconde est basées sur les données, où un algorithme génère directement des données à partir d'une base de données d'entrée sans avoir besoin d'une simulation d'environnement. La dernière possibilité sont les approches basées agents où un agent (c'est à dire une entité qui peut agir et raisonner sur son environnement) est utilisé pour déclencher des capteurs dans l'environnement afin de produire les données. En étudiant les apports de chacun par rapport à nos défis, il est

constaté que les agents sont plus précis. Enfin, certaines approches utilisant des simulateurs pour générer des données proposent aussi des méthodes pour valider la crédibilité des données synthétiques. Parmi elles, nous avons les approches *a priori*, où l'on utilise des données réelles en entrée pour concevoir le modèle et ainsi obtenir des résultats crédibles. Nous avons aussi les approches *a posteriori*, qui utilisent des métriques (comme des comparaisons statistiques) pour évaluer la crédibilité des données produites. Nous observons alors que les approches *a posteriori* sont les plus adaptées pour s'assurer de la crédibilité mais ne doivent pas se limiter à la comparaison statistique qui peut induire une imprécision dans les résultats.

Table 5.1 – Contribution des différentes approches existantes utilisant des modèles d'agent

Approches Existantes	Défi 1 (Générer des Comportements Crédibles)	Défi 2 (Compromis entre Contrôle et Autonomie)	Défi 3 (Validations Existantes de la Crédibilité)
Automatons	- - -	+	-
Approches Réactives	+ + +	- -	+
Approches basées Planification	- -	+ +	- - -
Approches basées Ordonnancement	+	+ +	+ +
Approches Probabilistes	+ +	- - -	+ + +
Approches basées Apprentissage	+ +	- - -	+
Approaches Mixtes	+ +	+ + +	- - -

Puisque les modèles d'agents sont intéressants pour à la fois permettre la compatibilité avec la génération de données mais aussi pour respecter nos trois défis, nous étudions plus précisément les modèles d'agent existant pour savoir si l'un d'eux pourrait répondre à nos trois défis simultanément. Contrairement à précédemment, cette étude sera étendue à des approches qui ne sont pas toujours utilisées pour la génération de données. Parmi les approches existantes, nous pouvons distinguer 6 catégories dont les apports par rapport à nos trois challenges sont rassemblées dans le Tableau 5.1. On remarque qu'aucune approche existante ne permet de répondre à nos 3 défis simultanément.

Chapitre 2: Modèle d'agent permettant à la fois le contrôle et l'autonomie sur les comportements

Comme les approches existantes ne répondent pas simultanément à nos défis, nous proposons dans ce chapitre un modèle d'agent compatible avec la génération de données, capable de générer des comportements crédibles (Défi 1) tout en offrant un compromis entre contrôle et autonomie (Défi 2).

Concrètement, nous proposons un modèle d'agent basé sur BDI [162] enrichi d'un ordonnanceur réactif capable d'adapter le niveau d'autonomie en fonction des contraintes des utilisateurs et des motivations propres de l'agent comme ses besoins (faim, fatigue, etc.). Dans ce chapitre, nous nous concentrons sur la structure globale de notre modèle d'agent ainsi que sur la gestion des contraintes de temps et des besoins (tels que la soif, la fatigue, etc.). Cette structure est inspirée des architectures BDI [162] qui est constitué de trois composantes principales: *Les croyances* récupérant les informations pertinentes de l'environnement, *les désirs* produisant les objectifs de l'agent en fonctions de ses croyances, et *les intentions* sélectionnant les actions qui permettront de satisfaire un désir. Nous avons décidé de baser notre modèle sur cette architecture car elle est réputée dans la communauté pour sa flexibilité et son approche intuitive pour simuler la prise de décision. Nous l'avons cependant enrichie d'un ordonnanceur car l'architecture de base ne permet pas d'anticiper les contraintes futures, et donc d'adresser notre Défi 2. En plus des trois composantes principales de BDI, nous avons ajouté un quatrième composant pour gérer plus finement les activités au niveau de l'exécution. Notre modèle d'agent illustrée en Figure 5.3 est donc composé des modèles suivants:

Agent Parameter: Ce processus rassemble tous les paramètres initiaux et les contraintes de l'utilisateur qui doivent être pris en compte lors de la simulation. L'utilisateur peut donner un calendrier d'activités en entrée pour fournir les activités qui doivent être effectuées à une période spécifique. **Ce calendrier rassemble toutes les activités obligatoires données par l'utilisateur. Ces activités sont principalement utilisées dans le modèle de prise de décision pour être intégrées dans le plan avec les heures et les durées correctes**. En modifiant les paramètres de besoins, des profils d'agents spécifiques peuvent être créés (l'agent mange plus souvent, se couche plus tard, etc.), augmentant ainsi la diversité des comportements. Toutes ces contraintes sont utilisées pour le contrôle du comportement de l'agent.

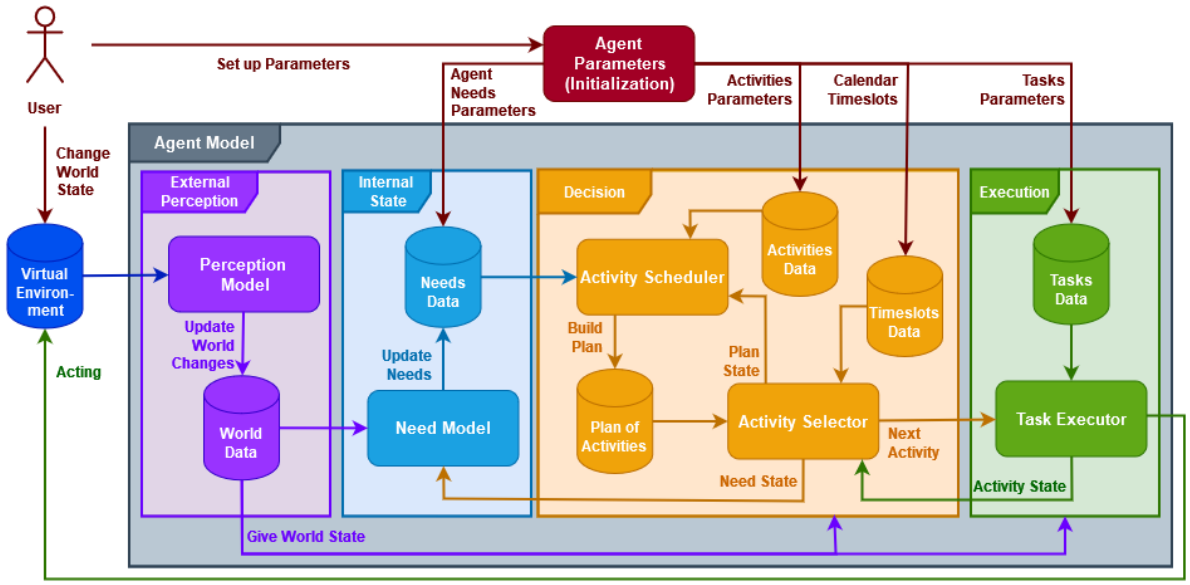

Figure 5.3 – Diagramme de notre modèle d'Agent

External Perception Model: Semblable aux croyances dans BDI, Ce modèle stocke toutes les données pertinentes de l'Environnement Virtuel (EV) dans sa base de données gérée par le modèle #FIVE [31] mis en œuvre dans *Xareus*[4]. Ce modèle est principalement utilisé pour filtrer les activités en fonction de leurs contraintes et fournir les objets et interactions nécessaires pour rendre l'exécution possible dans l'EV.

Internal State Model: Semblable aux désirs dans BDI, il modélise la motivation de l'agent à réaliser une activité. Pour l'instant, il est principalement utilisé pour mettre à jour l'urgence des besoins et appliquer les préférences de l'agent. Les besoins s'inspirent des besoins fondamentaux de l'être humain définis dans la théorie des besoins de Maslow [124]. Ils peuvent être physiologiques, comme la faim, ou plus sophistiqués, comme l'estime de soi. Ce modèle est utilisé pour l'autonomie de l'agent puisqu'il définit des objectifs à atteindre pendant les périodes de temps libre. Les besoins sont configurés à l'aide d'une fonction temporelle, comme dans les travaux de De Sevin et Thalmann [159]. Concrètement, pour calculer l'urgence des besoins, une valeur de priorité est évaluée en fonction d'un seuil de tolérance, utilisé pour savoir quand un besoin devient urgent, ainsi que l'intensité du besoin évoluant dans le temps. L'utilisateur peut également indiquer si un

4. *Xareus Software*: `https://team.inria.fr/hybrid/xareus/`

besoin peut interrompre des activités en cas d'urgence.

Decision-making Model: Ce modèle gère la prise de décision de l'agent et peut être lié aux intentions des modèles BDI. Mais contrairement à BDI, notre modèle établit des plans pendant la simulation. Ce plan est construit de manière à respecter les contraintes de l'utilisateur tout en produisant des choix autonomes pour satisfaire les besoins pendant les périodes de temps libre. Ce modèle contient deux processus principaux: l'Activity Scheduler et l'Activity Selector. L'Activity Scheduler est un ordonnanceur réactif qui élabore un plan d'activités dans une fenêtre de temps donnée par l'Activity Selector. Il tient compte à la fois des contraintes initiales et de l'urgence future des besoins. Notre ordonnanceur est conçu pour replanifier à tout moment, ce qui le rend compatible avec les Environnements Dynamiques. Les interruptions sont également gérées lorsqu'un besoin est trop urgent et qu'une activité peut être interrompue. Lors de la création du plan, les activités satisfaisant les besoins de l'agent sont placées au moment où ces besoins sont urgents. Si l'agent dispose encore de temps libre après avoir programmé des activités satisfaisant ses besoins, des activités par défaut, sont ajoutées pour occuper l'agent. Ces dernières sont choisies au hasard en tenant comptre des préférences de l'agent. L'approximation de l'urgence des besoins ainsi que le choix aléatoire des activités par défaut nous permettent d'introduire de la diversité: les mêmes contraintes d'entrée n'entraînent pas la même séquence d'activités en sortie. Concernant l'Activity Selector, il récupère progressivement l'activité à réaliser à partir du plan généré. Si aucune activité ne peut être exécutée ou a échoué, il envoie un message à l'ordonnanceur pour construire un nouveau plan jusqu'à la prochaine activité obligatoire. L'Activity Selector transmet également l'activité à réaliser au Task Executor qui renvoie ensuite l'état de l'activité.

Task Execution Model: Ce modèle exécute l'activité choisie dans l'EV en lançant la séquence de tâches correspondante. Ces tâches sont constituées d'actions et d'animations qui peuvent être directement exécutées. Ce processus, nouveau par rapport à BDI, permet une meilleur gestion de l'execution des activités. Ce modèle reçoit l'activité à exécuter de l'Activity Selector et renvoie en échange l'état de celle-ci. Dans notre implémentation, la séquence de tâches est réalisée avec un modèle basé sur les réseaux de Petri [140] appelé #SEVEN [49] mis en œuvre dans *Xareus*[4], où un jeton, se déplaçant d'un endroit à un autre, déclenche une action. La progression des jetons peut être gérée par des points de contrôle vérifiant si les conditions sont remplies.

Dans ce Chapitre, nous avons aussi appliqué une méthode de comparaison Entrée-Sortie pour vérifier la crédibilité des comportements générés. Les résultats obtenus dans cette section sont prometteurs et nous fournissent une première validation de la robustesse de notre modèle face à diverses situations où les contraintes de temps et de besoin sont plus ou moins importantes. Nos résultats montrent également que notre agent peut être entièrement contrôlable, partiellement contrôlable ou entièrement autonome sans modifier sa structure. Avec ceci, notre modèle répond à une partie de notre cas d'usage. Concernant l'évaluation de la crédibilité, les résultats sont aussi prometteurs puisque les comportements sont cohérents avec les paramètres d'entrée, diversifiés dans la plupart des cas, et consistants pour un même besoin satisfait. Cependant, cette méthode bien qu'essentielle n'est pas suffisante pour évaluer pleinement la crédibilité des comportements: d'autres méthodes de validation devront être réalisées.

Chapitre 3: Extension du modèle pour améliorer la gestion d'Environnements Dynamiques

Au sein de ce Chapitre, nous proposons un nouveau modèle permettant d'améliorer la gestion des Environnements Dynamiques pour mieux adresser notre Défi 1. En effet, bien que notre ordonnanceur soit compatible avec les Environnements Dynamiques, il reste cependant limité face à la gestion d'événements imprévus provenant de l'environnement (coups de téléphone, objet en panne etc.) et la gestion de ressources limitées (exemple: il n'y a plus de nourriture). Les interruptions des activités en cours quand une ressource est manquante ou quand un événement imprévu se produit n'est pas non plus géré avec la version précédente de notre modèle. Bien que notre cas d'usage puisse être déjà en partie atteint avec la version proposées dans le Chapitre 2, nous souhaitons aller plus loin afin de permettre aux utilisateurs d'utiliser notre modèle pour simuler plus de situations et générer des données plus diversifiés. Cela permettrait aussi de pouvoir faire coexister plusieurs agents en même temps et ainsi pouvoir créer des bases de données multi-agent. Au sein de l'état de l'art, nous avons aussi pu remarquer un manque de solutions pour gérer les ressources et les événements imprévus à la fois au niveau de la planification (anticipation des futures ressources manquantes, adaptation du plan après une interruption, etc.) et de l'exécution (interruption et reprise des activités en cours, etc.). C'est pourquoi, nous proposons dans ce Chapitre un modèle composé de 3 sous-modèles permettant de

gérer les ressources et les événements. Ces modèles décrits ci-dessous sont ajoutés dans notre modèle d'agent comme illustrés dans la Figure 5.4:

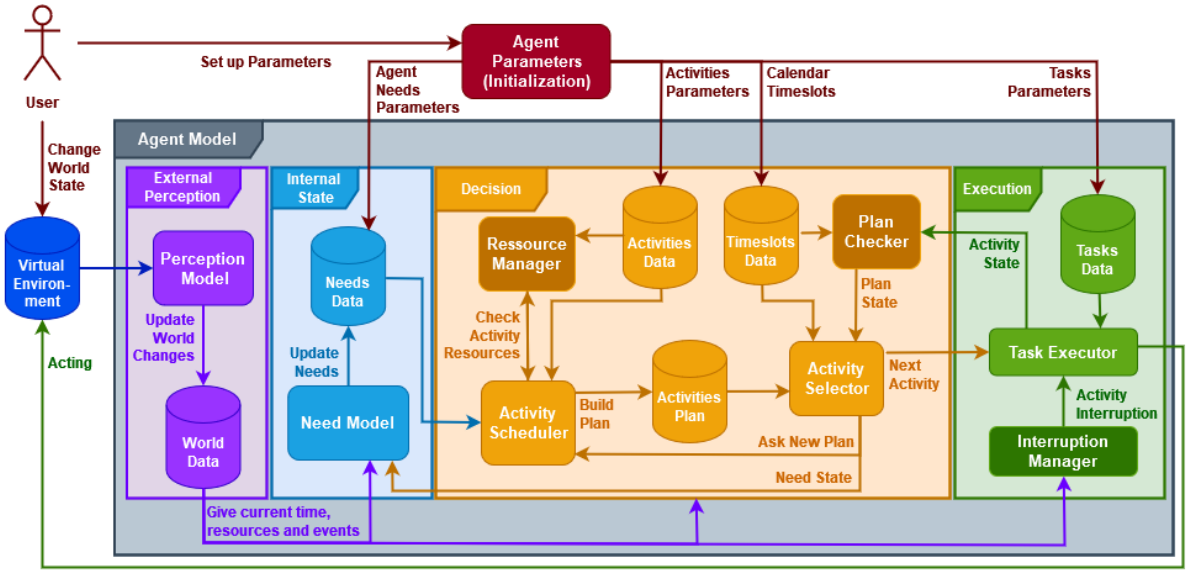

Figure 5.4 – Diagramme de notre Modèle d'Agent pour les Environnements Dynamiques

Le Ressource Manager: Il est utilisé pour trouver et restorer les ressources manquantes liées à chaque activité que l'agent souhaite programmer et ainsi anticiper les futures ressources manquantes. Pour cela, un processus est lancé pour trouver les activités restorant ces ressources. Par conséquent, il travaille en étroite collaboration avec l'Activity Scheduler chargé de construire le plan d'activités. Ce dernier gère aussi les contraintes de temps données par les utilisateurs et les besoins de l'agent. En parallèle, nous avons le Ressource Manager qui s'assure que chaque activité programmée soit compatible avec les ressources actuellement disponibles. Pour connaître l'état actuel, le Ressource Manager récupère ces informations dans l'External Perception Model. En parallèle, il récupère également des informations de la base de données des activités pour savoir quelles ressources sont consommées et produites par chacune d'entre elles. Lors de la création du plan, l'ordonnanceur envoie au Resource Manager chaque activité qui doit être planifiée ainsi que le temps restant. Ce dernier vérifie ensuite la disponibilité des ressources consommées par cette activité. Si certaines ressources sont indisponibles, un processus est lancé pour trouver les activités restorant les ressources manquantes tout en respectant le temps restant. Le plan final est ensuite envoyé à l'Activity Selector.

L'Interruption Manager: Ce modèle travaille en collaboration avec le Task Executor pour gérer les interruptions des activités exécutées. Au cours de l'exécution de l'activité, certaines ressources peuvent soudainement être épuisées ou un événement imprévu peut se produire. Ces changements soudains sont communiqués à l'Interruption Manager par l'External Perception Model. Par exemple, si pendant *Manger*, un autre agent mange la nourriture avant qu'elle ne soit utilisée, alors *Manger* ne peut pas continuer. Pour gérer ces changements inattendus, l'Interruption Manager peut arrêter l'activité en cours en toute sécurité. Un processus *Undo* est alors lancé par l'Interruption Manager pour permettre à l'agent de revenir à un état dans lequel il peut effectuer une autre activité. Si nécessaire, il peut également lancer un processus pour ramener l'agent à l'état où il peut reprendre l'activité interrompue. Dans le cas d'événements imprévus, l'Interruption Manager lance également l'activité traitant cet événement. Lorsque le processus d'interruption est terminé, le Task Executor est informé pour soit poursuivre l'activité interrompue, soit la terminer prématurément. Le Task Executor informe ensuite le Plan Checker.

Le Plan Checker: Le Plan Checker vérifie si le reste du plan est toujours réalisable après l'exécution d'une activité. Pour cela, il vérifie la disponibilité des ressources consommées par l'activité suivante. Si une notification d'interruption est également reçue, l'heure actuelle est également vérifiée pour savoir si l'agent est en avance ou en retard. En fonction de l'impact de ces interruptions, le Plan Checker peut soit apporter quelques ajustements au plan, soit demander une replanification en vidant le reste du plan. Le plan modifié est ensuite envoyé à l'Activity Selector qui sélectionne l'activité suivante ou demande un nouveau plan à l'Activity Scheduler si le plan modifié est vide.

Dans ce Chapitre, nous proposons aussi une version préliminaire d'un protocole d'expérience utilisateur afin d'évaluer notre modèle d'agent dans les Environnements Dynamiques. Le protocole se déroule en deux phases. La première permet de vérifier l'hypothèse suivante: *Les participants ont le sentiment que l'agent réagit comme le ferait un humain face à un événement imprévu.* Cette étape permettrait d'évaluer si les comportements générées au niveau de l'execution sont crédibles aux yeux des utilisateurs dans le cas d'interruptions. La seconde phase permettrait de vérifier l'hypothèse suivante: *Il y a peu de différence entre un calendrier d'activité établi par un humain et celui établi par un agent après une interruption.* Cette étape permettrait d'évaluer si les choix générées par l'agent au niveau de la planification sont proches de ce qu'un humain ferait après avoir subit une

interruption.

Chapitre 4: Méthodes pour valider la crédibilité des données synthétiques

Les Chapitres précédents ont permis de montrer comment nous pouvions addresser nos Défis 1 et 2 ainsi qu'une partie du Défi 3. Nous proposons maintenant dans ce Chapitre une méthode de validation afin de mieux adresser notre Défi 3. Pour relever ce défi, nous proposons de reproduire une expérience de collecte de données réelles dans une maison virtuelle, en utilisant notre modèle d'agent à la place d'un sujet humain. Pour savoir si nos données simulées peuvent être utilisées de la même manière que les données réelles, nous proposons d'évaluer la crédibilité des données simulées en les utilisant pour des modèles d'apprentissage automatique spécialisés dans la compréhension du contexte humain. Pour cela, nous avons choisis deux tâches de compréhension humaine: La prédiction d'activité future et la reconnaissance de l'activité actuelle. La reconnaissance des activités nous permet d'évaluer la crédibilité des interactions effectuées par l'agent dans l'environnement ainsi que les comportements des capteurs. En parallèle, la prédiction d'activité va permettre d'évaluer la crédibilité du modèle de prise de décision mais aussi des routines générées.

Figure 5.5 – Sections du bureau, de la cuisine et du salon, tels qu'ils sont vus dans l'appartement réel d'Orange4Home et dans notre environnement virtuel.

La base de données réelle que nous avons souhaités répliquer est celle d'Orange4Home

[51]. Pour cela, nous avons reproduit en virtuel la maison intelligente dans laquelle les données réelles ont été générées comme illustré en Figures 5.5. Nous avons aussi répliquer les capteurs qui étaient les plus importants pour permettre la reconnaissance des activités. Ces capteurs concernent tous ceux dont l'état change après l'action d'un agent (interrupteurs, bouton ON/OFF des appareils, ouverture et fermeture des portes, etc.) mais aussi ceux qui mesurent des débits d'eau et ceux qui détectent la présence de l'agent dans une pièce (comme les détecteurs de présence). Enfin, nous avons aussi respecté le même protocole que la personne réelle qui a généré les données d'Orange4Home en respectant notamment le calendrier d'activité à réaliser à des heures précise. Ce calendrier a été placé en entrée de notre modèle d'agent afin qu'il soit considérer comme une contrainte forte par notre agent.

Pour vérifier si nos données synthétiques peuvent se substituer au données réelles, nous avons entraînés les deux algorithmes de prédiction et de détection d'activité avec d'une part des données synthétiques et d'autres part des données réelles. Nous avons ensuite comparé l'écart de performance entre les deux. Si les écarts sont faibles, alors cela signifie que les données synthétiques offre le même niveau d'efficacité que les données réelles dans ce contexte précis. Les résultats obtenus sont prometteur car l'écart entre les deux est relativement faible même si quelques imperfections ont été constatées dans certains cas. Cela montre donc qu'utiliser des données synthétiques à la place de données réelles pour entraîner des algorithmes de machine learning est prometteur. Cette expérience n'est cependant pas suffisante pour valider le fait que notre modèle produise des données synthétiques suffisamment crédibles pour être substituable aux données réelles. Elles permettent cependant de confirmer que la substitution est possible dans un cas particulier. D'autres tests avec d'autres bases de données seront nécessaire dans le futur pour valider notre modèle et ainsi pleinement adresser notre Challenge 3.

Conclusion

Dans cette thèse, nous avons tenter de proposer une solution pour simuler d'humains virtuels générant des comportements crédibles tout en étant compatibles avec la génération de données d'activité quotidienne. Pour y parvenir, nous avons expliqué que trois défis devaient être relevés. Le premier consiste à proposer un modèle de comportement humain capable de générer des comportements crédibles. Le second consiste à proposer un modèle offrant un compromis entre contrôle et autonomie afin d'être compatible avec

la génération de données. Enfin, la dernière consiste à mettre en place différentes méthodes de validation pour vérifier la crédibilité des comportements générés et des données synthétiques.

A une époque où la collecte de données est devenue un défi majeur dans de nombreux domaines, nous espérons que proposer de nouvelles approches capables de générer des données synthétiques exploitables permettrait d'enrichir les bases de données réelles existantes, avoir des informations sur les situations difficiles à réaliser dans la vie réelle, et limiter la collecte de données provenant d'utilisateurs réels. En parallèle, relever le défi 2 nous a amené à développer une approche ayant l'avantage d'être contrôlable, configurable et transparente dans son fonctionnement. Ceci va à l'encontre de la nouvelle génération d'algorithmes basés sur l'apprentissage, où il devient difficile de contrôler les comportements de sortie et de savoir comment ces comportements ont été obtenus. De plus, notre approche a l'avantage de ne dépendre d'aucune donnée pour fonctionner.

Dans nos travaux futurs à court terme, nous souhaitons améliorer les résultats obtenus dans le Chapitre 4 et réaliser l'expérience utilisateur proposée dans le Chapitre 3. Nous souhaiterions ensuite améliorer les relations entre l'agent et son environnements en proposant de nouveaux besoins qui dépendraient de facteurs externes tels que la température ou l'humidité mais aussi de facteurs sociétaux tels que l'envie ou non de partager des activités avec d'autres agents. Nous souhaiterions ensuite rendre interdépendants les besoins afin que la satisfaction de l'un impacte l'urgence de l'autre (par exemple *Faire du Sport* augmenterait la soif). Nous aimerions ensuite étudier plus finement l'impact des préférences sur les comportements. Enfin, nous souhaiterions aussi proposer de nouveaux outils pour créer plus facilement de nouvelles activités et de nouvelles animations, mais aussi pour offrir de nouveaux moyens de configurer l'agent.

Concernant nos travaux sur le long terme, nous souhaiterions développer la collaboration entre plusieurs agents afin qu'il puisse se synchroniser et réaliser des activités quotidiennes ensembles. Pour aller plus le loin dans le test de la robustesse de notre modèle, nous souhaiterions faire collaborer un agent avec un humain. Nous aimerions ensuite étendre notre modèle sur d'autres types de données comme la capture vidéo afin d'avoir d'autres méthodes de validation. Enfin, nous souhaiterions proposer des méthodes pour détecter les potentiels biais qui seraient produits par les algorithmes d'apprentissage servant à vérifier la crédibilité de nos données, en étudiant notamment l'impact des données synthétiques sur les biais d'apprentissage.

BIBLIOGRAPHY

[1] C. Adam and B. Gaudou, « Bdi agents in social simulations: a survey », *The Knowledge Engineering Review*, vol. 31, *3*, pp. 207–238, 2016.

[2] L. Airale, D. Vaufreydaz, and X. Alameda-Pineda, « Socialinteractiongan: multi-person interaction sequence generation », *IEEE Transactions on Affective Computing*, 2022.

[3] H. Alemdar, H. Ertan, O. D. Incel, and C. Ersoy, « Aras human activity datasets in multiple homes with multiple residents », *in 2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*, IEEE, 2013, pp. 232–235.

[4] N. Alshammari, T. Alshammari, M. Sedky, J. Champion, and C. Bauer, « Openshs: open smart home simulator », *Sensors*, vol. 17, *5*, p. 1003, 2017.

[5] E. Amouroux, T. Huraux, F. Sempé, N. Sabouret, and Y. Haradji, « Simulating human activities to investigate household energy consumption », *in 5th International Conference on Agents and ARTificial intelligence (ICAART 2013)*, 2013.

[6] É. Amouroux, T. Huraux, F. Sempé, N. Sabouret, and Y. Haradji, « Smach: agent-based simulation investigation on human activities and household electrical consumption », *in Agents and Artificial Intelligence: 5th International Conference, ICAART 2013, Barcelona, Spain, February 15-18, 2013. Revised Selected Papers 5*, Springer, 2014, pp. 194–210.

[7] G. Anastassakis and T. Panayiotopoulos, « A tool for programming the behaviour of intelligent virtual agents in Prolog », en, *in 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, Corfu, Greece: IEEE, Jul. 2015, pp. 1–6.

[8] B. Archimede and T. Coudert, « Reactive scheduling using a multi-agent model: the scep framework », *Engineering Applications of Artificial Intelligence*, vol. 14, *5*, pp. 667–683, 2001.

[9] T.-C. Au, U. Kuter, and D. Nau, « Planning for interactions among autonomous agents », *in Programming Multi-Agent Systems: 6th International Workshop, ProMAS 2008, Estoril, Portugal, May 13, 2008. Revised Invited and Selected Papers 6*, Springer, 2009, pp. 1–23.

[10] J. Auld and A. K. Mohammadian, « Activity planning processes in the agent-based dynamic activity planning and travel scheduling (adapts) model », *Transportation Research Part A: Policy and Practice*, vol. 46, *8*, pp. 1386–1403, 2012.

[11] N. Avradinis, T. Panayiotopoulos, and G. Anastassakis, « Behavior believability in virtual worlds: agents acting when they need to », en, *SpringerPlus*, Dec. 2013.

[12] N. Avradinis, T. Panayiotopoulos, and G. Anastassakis, « Modelling basic needs as agent motivations », *International Journal of Computational Intelligence Studies 10*, vol. 2, *1*, pp. 52–75, 2013.

[13] R. Aylett and M. Cavazza, « Intelligent virtual environments-a state-of-the-art report. », *in Eurographics (State of the Art Reports)*, 2001.

[14] B. Azvine, D. Djian, K. C. Tsui, and W. Wobcke, « The intelligent assistant: an overview », *Intelligent Systems and Soft Computing*, pp. 215–238, 2000.

[15] N. Banovic, T. Buzali, F. Chevalier, J. Mankoff, and A. K. Dey, « Modeling and understanding human routine behavior », *in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 248–260.

[16] A. L. Barriuso, F. De la Prieta, G. Villarrubia González, D. H. De La Iglesia, and Á. Lozano, « Movicloud: agent-based 3d platform for the labor integration of disabled people », *Applied Sciences*, vol. 8, *3*, p. 337, 2018.

[17] M. M. Bejani and M. Ghatee, « A systematic review on overfitting control in shallow and deep neural networks », *Artificial Intelligence Review*, pp. 1–48, 2021.

[18] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, « Jade: a software framework for developing multi-agent applications. lessons learned », *Information and Software technology*, vol. 50, *1-2*, pp. 10–21, 2008.

[19] P. Bercher, S. Keen, and S. Biundo, « Hybrid planning heuristics based on task decomposition graphs », *in Proceedings of the International Symposium on Combinatorial Search*, vol. 5, 2014, pp. 35–43.

[20] C. R. Bhat, K. G. Goulias, R. M. Pendyala, R. Paleti, R. Sidharthan, L. Schmitt, and H.-H. Hu, « A household-level activity pattern generation model with an application for southern california », *Transportation*, vol. 40, pp. 1063–1086, 2013.

[21] C. R. Bhat, K. G. Goulias, R. M. Pendyala, R. Paleti, R. Sidharthan, L. Schmitt, and H.-h. Hu, « A household-level activity pattern generation model for the simulator of activities, greenhouse emissions, networks, and travel (simagent) system in southern california », *in 91st Annual Meeting of the Transportation Research Board, Washington, DC*, 2012.

[22] A. L. Blum and M. L. Furst, « Fast planning through planning graph analysis », *Artificial intelligence*, vol. 90, *1-2*, pp. 281–300, 1997.

[23] A. Bogdanovych and T. Trescak, « Generating Needs, Goals and Plans for Virtual Agents in Social Simulations », en, *in Intelligent Virtual Agents*, D. Traum, W. Swartout, P. Khooshabeh, S. Kopp, S. Scherer, and A. Leuski, Eds., vol. 10011, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2016, pp. 397–401.

[24] A. Bogdanovych and T. Trescak, « To plan or not to plan: lessons learned from building large scale social simulations », *in Intelligent Virtual Agents: 17th International Conference, IVA 2017, Stockholm, Sweden, August 27-30, 2017, Proceedings 17*, Springer, 2017, pp. 53–62.

[25] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi, « Multi-agent oriented programming with jacamo », *Science of Computer Programming*, vol. 78, *6*, pp. 747–761, 2013.

[26] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

[27] B. Bouchard, S. Gaboury, K. Bouchard, and Y. Francillette, « Modeling human activities using behaviour trees in smart homes », *in Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference*, 2018, pp. 67–74.

[28] K. Bouchard, A. Ajroud, B. Bouchard, and A. Bouzouane, « Simact: a 3d open source smart home simulator for activity recognition », *in Advances in Computer Science and Information Technology: AST/UCMA/ISA/ACN 2010 Conferences,*

Miyazaki, Japan, June 23-25, 2010. Joint Proceedings, Springer, 2010, pp. 524–533.

[29] M. Bourgais, P. Taillandier, and L. Vercouter, « Ben: an architecture for the behavior of social agents », *Journal of Artificial Societies and Social Simulation*, vol. 23, *4*, 2020.

[30] M. Bourgais, P. Taillandier, L. Vercouter, and C. Adam, « Emotion modeling in social simulation: a survey », *Journal of Artificial Societies and Social Simulation*, vol. 21, *2*, 2018.

[31] R. Bouville, V. Gouranton, T. Boggini, F. Nouviale, and B. Arnaldi, « #FIVE : High-level components for developing collaborative and interactive virtual environments », en, *in 2015 IEEE 8th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, Arles, France: IEEE, Mar. 2015, pp. 33–40.

[32] D. A. Bowman, C. North, J. Chen, N. F. Polys, and P. S. Pyla, « Information-Rich Virtual Environments: Theory, Tools, and Research Agenda », en, p. 10, 2003.

[33] M. Bratman, « Intention, plans, and practical reason », 1987.

[34] J. C. Brustoloni, *Autonomous agents: Characterization and requirements*. Citeseer, 1991.

[35] R. C. Cardoso and A. Ferrando, « A review of agent-based programming for multi-agent systems », *Computers*, vol. 10, *2*, p. 16, 2021.

[36] A. Casali, L. Godo, and C. Sierra, « A graded BDI agent model to represent and reason about preferences », en, *Artificial Intelligence*, vol. 175, *7-8*, pp. 1468–1478, May 2011.

[37] C. Castelfranchi, « Guarantees for autonomy in cognitive agent architecture », *in Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages Amsterdam, The Netherlands August 8–9, 1994 Proceedings 1*, Springer, 1995, pp. 56–70.

[38] C. Castelfranchi and R. Falcone, « From automaticity to autonomy: the frontier of artificial agents », *Agent autonomy*, pp. 103–136, 2003.

[39] M. Cavazza, « Al in computer games: Survey and perspectives », en, *Virtual Reality*, vol. 5, *4*, pp. 223–235, Dec. 2000.

[40] M. Cavazza, F. Charles, and S. J. Mead, « Interacting with virtual characters in interactive storytelling », *in Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, 2002, pp. 318–325.

[41] P. H.-M. Chang, Y.-H. Chien, E. C.-C. Kao, and V.-W. Soo, « A knowledge-based scenario framework to support intelligent planning characters », *in IVA*, Springer, 2005, pp. 134–145.

[42] J. M. Chaquet, E. J. Carmona, and A. Fernández-Caballero, « A survey of video datasets for human action and activity recognition », *Computer Vision and Image Understanding*, vol. 117, *6*, pp. 633–659, 2013.

[43] D. Charypar and K. Nagel, « Generating complete all-day activity plans with genetic algorithms », *Transportation*, vol. 32, *4*, pp. 369–397, 2005.

[44] D. Charypar and K. Nagel, « Q-learning for flexible learning of daily activity plans », *Transportation Research Record*, vol. 1935, *1*, pp. 163–169, 2005.

[45] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, « Deep learning for sensor-based human activity recognition: overview, challenges, and opportunities », *ACM Computing Surveys (CSUR)*, vol. 54, *4*, pp. 1–40, 2021.

[46] P. Chevaillier, T.-H. Trinh, M. Barange, P. De Loor, F. Devillers, J. Soler, and R. Querrec, « Semantic modeling of virtual environments using mascaret », *in 2012 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, IEEE, 2012, pp. 1–8.

[47] A. Chowanda, P. Blanchfield, M. Flintham, and M. Valstar, « Erisa: building emotionally realistic social game-agents companions », *in Intelligent Virtual Agents: 14th International Conference, IVA 2014, Boston, MA, USA, August 27-29, 2014. Proceedings 14*, Springer, 2014, pp. 134–143.

[48] G. Claude, V. Gouranton, and B. Arnaldi, « Roles in collaborative virtual environments for training », *in Proceedings of International Conference on Artificial Reality and Telexistence Eurographics Symposium on Virtual Environments*, 2015, pp. 1–8.

[49] G. Claude, V. Gouranton, R. B. Berthelot, and B. Arnaldi, « Short paper: #SEVEN, a sensor effector based scenarios model for driving collaborative virtual environment », *in ICAT-EGVE, International Conference on Artificial Reality and Telexistence, Eurographics Symposium on Virtual Environments*, 2014, pp. 1–4.

[50] J. Cumin, G. Lefebvre, F. Ramparany, and J. L. Crowley, « Human activity recognition using place-based decision fusion in smart homes », *in International and Interdisciplinary Conference on Modeling and Using Context*, Cham: Springer International Publishing, 2017, pp. 137–150.

[51] J. Cumin, G. Lefebvre, F. Ramparany, and J. L. Crowley, « A dataset of routine daily activities in an instrumented home », *in Ubiquitous Computing and Ambient Intelligence - 11th International Conference, UCAmI 2017, Philadelphia, PA, USA, November 7-10, 2017, Proceedings*, S. F. Ochoa, P. Singh, and J. Bravo, Eds., ser. Lecture Notes in Computer Science, vol. 10586, Springer, 2017, pp. 413–425.

[52] J. Cumin, G. Lefebvre, F. Ramparany, and J. L. Crowley, « PSINES: activity and availability prediction for adaptive ambient intelligence », *ACM Trans. Auton. Adapt. Syst.*, vol. 15, *1*, 1:1–1:12, 2021.

[53] K. Darty, J. Saunier, and N. Sabouret, « Agents behavior semi-automatic analysis through their comparison to human behavior clustering », *in Intelligent Virtual Agents: 14th International Conference, IVA 2014, Boston, MA, USA, August 27-29, 2014. Proceedings 14*, Springer, 2014, pp. 154–163.

[54] S. Das, R. Dai, M. Koperski, L. Minciullo, L. Garattoni, F. Bremond, and G. Francesca, « Toyota smarthome: real-world activities of daily living », *in Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 833–842.

[55] A. De Almeida, P. Fonseca, B. Schlomann, and N. Feilberg, « Characterization of the household electricity consumption in the eu, potential energy savings and specific policy recommendations », *Energy and buildings*, vol. 43, *8*, pp. 1884–1894, 2011.

[56] F. De la Torre, J. Hodgins, A. Bargteil, X. Martin, J. Macey, A. Collado, and P. Beltran, « Guide to the carnegie mellon university multimodal activity (cmu-mmac) database », 2009.

[57] L. De Silva, S. Sardina, and L. Padgham, « First principles planning in bdi systems », *in Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2009, pp. 1105–1112.

[58] E. Delzendeh, S. Wu, A. Lee, and Y. Zhou, « The impact of occupants' behaviours on building energy analysis: a research review », *Renewable and sustainable energy reviews*, vol. 80, pp. 1061–1071, 2017.

[59] L. Ding, P. Kolari, Z. Ding, and S. Avancha, « Using ontologies in the semantic web: a survey », *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, pp. 79–113, 2007.

[60] D. Djaouti, J. Alvarez, and J.-P. Jessel, « Classifying serious games: the g/p/s model », *in Handbook of research on improving learning and motivation through educational games: Multidisciplinary approaches*, IGI global, 2011, pp. 118–136.

[61] D. Dörner and C. D. Güss, « PSI: A Computational Architecture of Cognition, Motivation, and Emotion », en, *Review of General Psychology*, vol. 17, *3*, pp. 297–317, Sep. 2013.

[62] A. Dorri, S. S. Kanhere, and R. Jurdak, « Multi-agent systems: a survey », *Ieee Access*, vol. 6, pp. 28 573–28 593, 2018.

[63] M. Dragoni, C. Ghidini, P. Busetta, M. Fruet, and M. Pedrotti, « Using ontologies for modeling virtual reality scenarios », *in The Semantic Web. Latest Advances and New Domains: 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31–June 4, 2015. Proceedings 12*, Springer, 2015, pp. 575–590.

[64] F. Dvorák and R. Barták, « Ai planning with time and resource constraints », *in Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS 2010)*, 2010, pp. 5–13.

[65] F. Dvořák, « Ai planning with time and resource constraints », 2009.

[66] H. T. Ebuy, H. Bril El Haouzi, R. Benelmir, and R. Pannequin, « Occupant behavior impact on building sustainability performance: a literature review », *Sustainability*, vol. 15, *3*, p. 2440, 2023.

[67] S. Egami, S. Nishimura, and K. Fukuda, « Virtualhome2kg: constructing and augmenting knowledge graphs of daily activities using virtual space. », *in ISWC (Posters/Demos/Industry)*, 2021.

[68] A. Elbayoudi, A. Lotfi, C. Langensiepen, and K. Appiah, « Modelling and simulation of activities of daily living representing an older adult's behaviour », *in Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2015, pp. 1–8.

[69] K. Erol, J. A. Hendler, and D. S. Nau, « Umcp: a sound and complete procedure for hierarchical task-network planning. », *in Aips*, vol. 94, 1994, pp. 249–254.

[70] R. E. Fikes and N. J. Nilsson, « Strips: A new approach to the application of theorem proving to problem solving », en, *Artificial Intelligence*, vol. 2, *3-4*, pp. 189–208, Dec. 1971.

[71] J. Flotyński and K. Walczak, « Ontology-Based Representation and Modelling of Synthetic 3D Content: A State-of-the-Art Review: Ontology-Based Representation and Modelling of Synthetic 3D Content », en, *Computer Graphics Forum*, vol. 36, *8*, pp. 329–353, Dec. 2017.

[72] M. Fox and D. Long, « Pddl2. 1: an extension to pddl for expressing temporal planning domains », *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[73] S. Franklin and A. Graesser, « Is it an agent, or just a program?: a taxonomy for autonomous agents », *in International workshop on agent theories, architectures, and languages*, Springer, 1996, pp. 21–35.

[74] P. Gebhard, « ALMA: a layered model of affect », en, *in Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems - AAMAS '05*, The Netherlands: ACM Press, 2005, p. 29.

[75] I. Georgievski and M. Aiello, « An overview of hierarchical task network planning », *arXiv preprint arXiv:1403.7426*, 2014.

[76] N. Ghouaiel, S. Garbaya, J.-M. Cieutat, and J.-P. Jessel, « Mobile augmented reality in museums: towards enhancing visitor's learning experience », *International Journal of Virtual Reality*, vol. 17, *1*, pp. 21–31, 2017.

[77] L. R. Goldberg, « An alternative" description of personality": the big-five factor structure. », *Journal of personality and social psychology*, vol. 59, *6*, p. 1216, 1990.

[78] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, « Generative adversarial networks », *Communications of the ACM*, vol. 63, *11*, pp. 139–144, 2020.

[79] K. G. Goulias, C. R. Bhat, R. M. Pendyala, Y. Chen, R. Paleti, K. C. Konduri, G. Huang, and H.-H. Hu, « Simulator of activities, greenhouse emissions, networks, and travel (simagent) in southern california: design, implementation, preliminary findings, and integration plans », *in 2011 IEEE Forum on Integrated and Sustainable Transportation Systems*, IEEE, 2011, pp. 164–169.

[80] L. Gramoli, J. Lacoche, A. Foulonneau, V. Gouranton, and B. Arnaldi, « Control your virtual agent in its daily-activities for long periods », *in International Conference on Practical Applications of Agents and Multi-Agent Systems*, Springer, 2022, pp. 203–216.

[81] L. Gramoli, J. Lacoche, A. Foulonneau, V. Gouranton, and B. Arnaldi, « Needs model for an autonomous agent during long-term simulations », *in Modeling and Animating Realistic Crowds and Humans (MARCH) Workshop, IEEE International Conference on Artificial Intelligence and Virtual Reality, AIVR 2021, Taichung, Taiwan, November 15-17, 2021*, IEEE, 2021, pp. 134–138.

[82] G. Gröger and L. Plümer, « Citygml–interoperable semantic 3d city models », *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 71, pp. 12–33, 2012.

[83] R. Guizzardi, B. G. Carneiro, D. Porello, and G. Guizzardi, « A core ontology on decision making », *in Proceedings of the XIII Seminar on Ontology Research in Brazil and IV Doctoral and Masters Consortium on Ontologies (ONTOBRAS 2020)*, CEUR-WS, vol. 2728, 2020, pp. 9–21.

[84] M. Gutierrez, F. Vexo, and D. Thalmann, « Semantics-based representation of virtual environments », *International journal of computer applications in technology*, vol. 23, *2-4*, pp. 229–238, 2005.

[85] K. M. N. Habib, « A random utility maximization (rum) based dynamic activity scheduling model: application in weekend activity scheduling », *Transportation*, vol. 38, *1*, pp. 123–151, 2011.

[86] O. Handel, « Modeling dynamic decision-making of virtual humans », *Systems*, vol. 4, *1*, p. 4, 2016.

[87] O. Handel and A. Borrmann, « Analyzing a multi-agent-system decision architecture aiming to model the behavior of virtual humans », *in Proc. of the 16th International Conference on Computing in Civil and Building Engineering*, 2016.

[88] A. Helal, K. Cho, W. Lee, Y. Sung, J. Lee, and E. Kim, « 3d modeling and simulation of human activities in smart spaces », *in 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, IEEE, 2012, pp. 112–119.

[89] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, « Gans trained by a two time-scale update rule converge to a local nash equilibrium », *in Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6629–6640.

[90] B. Ho, D. Vogts, and J. Wesson, « A smart home simulation tool to support the recognition of activities of daily living », *in Proceedings of the South African Institute of Computer Scientists and Information Technologists 2019*, 2019, pp. 1–10.

[91] N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas, « Jack intelligent agents-summary of an agent infrastructure », *in 5th International conference on autonomous agents*, vol. 6, 2001.

[92] S. N. N. Htun, S. Egami, and K. Fukuda, « A survey and comparison of activities of daily living datasets in real-life and virtual spaces », *in 2023 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, 2023, pp. 1–7.

[93] H. Jang, S. Hao, P. M. Chu, P. K. Sharma, Y. Sung, and K. Cho, « Deep Q-network-based multi-criteria decision-making framework for virtual simulation environment », en, *Neural Computing and Applications*, p. 15, Apr. 2020.

[94] C.-J. Jorgensen and F. Lamarche, « Space and Time Constrained Task Scheduling for Crowd Simulation », Research Report PI 2013, Jan. 2014.

[95] C.-J. Jørgensen and F. Lamarche, « Combining activity scheduling and path planning to populate virtual cities », *in Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1129–1130.

[96] D. B. Jørgensen, K. Hallenborg, and Y. Demazeau, « Extending agent based telehealth platform with activities of daily living reasoning capabilities », *in 2016 IEEE International Conference on Healthcare Informatics (ICHI)*, IEEE, 2016, pp. 168–176.

[97] O. Kamara-Esteban, G. Azkune, A. Pijoan, C. E. Borges, A. Alonso-Vicario, and D. López-de-Ipiña, « Massha: an agent-based approach for human activity simulation in intelligent environments », *Pervasive and Mobile Computing*, vol. 40, pp. 279–300, 2017.

[98] Y. Kang and A.-H. Tan, « Self-organizing cognitive models for virtual agents », *in Intelligent Virtual Agents: 13th International Conference, IVA 2013, Edinburgh, UK, August 29-31, 2013. Proceedings 13*, Springer, 2013, pp. 29–43.

[99] A. Kashif, S. Ploix, and J. Dugdale, « A modern approach to include representative behaviour models in energy simulations », *Towards Energy Smart Homes: Algorithms, Technologies, and Applications*, pp. 489–542, 2021.

[100] A. Kashif, S. Ploix, J. Dugdale, P. Reignier, and M. K. Shahzad, « Virtual simulation with real occupants using serious games », *in Proceedings of the 14th International Conference of the International Building Performance Simulation Association*, 2015, pp. 2712–2719.

[101] A. Kashif, S. Ploix, J. Dugdale, and X. H. B. Le, « Simulating the dynamics of occupant behaviour for power management in residential buildings », *Energy and Buildings*, vol. 56, pp. 85–93, 2013.

[102] J. Kessing, T. Tutenel, and R. Bidarra, « Designing semantic game worlds », *in Proceedings of the The third workshop on Procedural Content Generation in Games*, 2012, pp. 1–9.

[103] J. Kim, T. Hong, J. Jeong, M. Lee, M. Lee, K. Jeong, C. Koo, and J. Jeong, « Establishment of an optimal occupant behavior considering the energy consumption and indoor environmental quality by region », *Applied Energy*, vol. 204, pp. 1431–1443, 2017.

[104] L. Klein, J.-y. Kwak, G. Kavulya, F. Jazizadeh, B. Becerik-Gerber, P. Varakantham, and M. Tambe, « Coordinating occupant behavior for building energy and comfort management using multi-agent systems », *Automation in construction*, vol. 22, pp. 525–536, 2012.

[105] U. Köckemann, « Constraint-based methods for human-aware planning », Ph.D. dissertation, Örebro university, 2016.

[106] U. Köeckemann, F. Pecora, and L. Karlsson, « Inferring context and goals for online human-aware planning », *in 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2015, pp. 550–557.

[107] D. Kudryavtsev, T. Gavrilova, M. Smirnova, and K. Golovacheva, « Modelling consumer knowledge: the role of ontology », *Procedia Computer Science*, vol. 176, pp. 500–507, 2020.

[108] N. Kushmerick, S. Hanks, and D. S. Weld, « An algorithm for probabilistic planning », *Artificial Intelligence*, vol. 76, *1-2*, pp. 239–286, 1995.

[109] P. Labone and M. Ghallab, « Planning with sharable resource constraints », *in International Joint Conference on Artificial Intelligence*, 1995.

[110] P. Laborie, « Algorithms for propagating resource constraints in ai planning and scheduling: existing approaches and new results », *Artificial Intelligence*, vol. 143, *2*, pp. 151–188, 2003.

[111] J. Lacoche, M. Le Chénéchal, E. Villain, and A. Foulonneau, « Model and tools for integrating iot into mixed reality environments: towards a virtual-real seamless continuum », *in ICAT-EGVE 2019-International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments*, 2019.

[112] J. E. Laird, « An analysis and comparison of act-r and soar », *arXiv preprint arXiv:2201.09305*, 2022.

[113] V. Lanquepin, K. Carpentier, D. Lourdeaux, M. Lhommet, C. Barot, and K. Amokrane, « Humans: a human models based artificial environments software platform », *in Proceedings of the Virtual Reality International Conference: Laval Virtual*, 2013, pp. 1–8.

[114] F. Lécuyer, V. Gouranton, A. Lamercerie, A. Reuzeau, B. Caillaud, and B. Arnaldi, « Unveiling the implicit knowledge, one scenario at a time », en, *The Visual Computer*, vol. 36, *10-12*, pp. 1951–1963, Oct. 2020.

[115] F. Lécuyer, V. Gouranton, A. Reuzeau, R. Gaugne, and B. Arnaldi, « Action sequencing in vr, a no-code approach », *Transactions on Computational Science XXXVII: Special Issue on Computer Graphics*, pp. 57–76, 2020.

[116] D. Lee and M. Yannakakis, « Principles and methods of testing finite state machines-a survey », *Proceedings of the IEEE*, vol. 84, *8*, pp. 1090–1123, 1996.

[117] W. Lee, S. Cho, P. Chu, H. Vu, S. Helal, W. Song, Y.-S. Jeong, and K. Cho, « Automatic agent generation for IoT-based smart house simulator », en, *Neurocomputing*, vol. 209, pp. 14–24, Oct. 2016.

[118] C. Li, F. Xia, R. Martin-Martin, M. Lingelbach, S. Srivastava, B. Shen, K. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, K. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese, « Igibson 2.0: object-centric simulation for robot learning of everyday household tasks », *arXiv preprint arXiv:2108.03272*, 2021.

[119] J. Li, Z. J. Yu, F. Haghighat, and G. Zhang, « Development and improvement of occupant behavior models towards realistic building performance simulation: a review », *Sustainable Cities and Society*, vol. 50, p. 101 685, 2019.

[120] D. Long, M. Fox, L. Sebastia, and A. Coddington, « An examination of resources in planning », *in Proc. of 19th UK Planning and Scheduling Workshop, Milton Keynes*, 2000.

[121] J. Lundström, J. Synnott, E. Järpe, and C. D. Nugent, « Smart home simulation using avatar control and probabilistic sampling », *in 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, IEEE, 2015, pp. 336–341.

[122] S. Marsella and J. Gratch, « Modeling coping behavior in virtual humans: don't worry, be happy », *in Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003, pp. 313–320.

[123] S. Mascarenhas, M. Guimarães, P. A. Santos, J. Dias, R. Prada, and A. Paiva, « Fatima toolkit–toward an effective and accessible tool for the development of intelligent virtual agents and social robots », *arXiv preprint arXiv:2103.03020*, 2021.

[124] A. H. Maslow, « A theory of human motivation. », *Psychological review*, vol. 50, *4*, p. 370, 1943.

[125] R. J. McCall, S. Franklin, U. Faghihi, J. Snaider, and S. Kugele, « Artificial Motivation for Cognitive Software Agents », en, *Journal of Artificial General Intelligence*, vol. 11, *1*, pp. 38–69, Jan. 2020.

[126] D. M. McDermott, « The 1998 ai planning systems competition », *AI magazine*, vol. 21, *2*, pp. 35–35, 2000.

[127] K. Meister, M. Frick, and K. W. Axhausen, « Generating daily activity schedules for households using genetic algorithms », *in 5th Swiss Transport Research Conference (STRC 2005)*, STRC, 2005.

[128] J.-A. Meyer, « Artificial life and the animat approach to artificial intelligence », *in Artificial intelligence*, Elsevier, 1996, pp. 325–354.

[129] E. J. Miller and M. J. Roorda, « Prototype model of household activity-travel scheduling », *Transportation Research Record*, vol. 1831, *1*, pp. 114–121, 2003.

[130] K. L. Myers, « User guide for the procedural reasoning system », *SRI International AI Center Technical Report. SRI International, Menlo Park, CA*, 1997.

[131] D. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila, « Shop: simple hierarchical ordered planner », *in Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, 1999, pp. 968–973.

[132] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, « Shop2: an htn planning system », *Journal of artificial intelligence research*, vol. 20, pp. 379–404, 2003.

[133] D. Norman, *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[134] I. Nunes, C. J. De Lucena, and M. Luck, « Bdi4jade: a bdi layer on top of jade », *in Ninth International Workshop on Programming Multi-Agent Systems:(ProMAS 2011), Taipei, Taiwan*, 2011, pp. 88–103.

[135] S. A. Ordóñez Medina, « Personalized multi-activity scheduling of flexible activities », *Arbeitsberichte Verkehrs-und Raumplanung*, vol. 1099, 2015.

[136] J. Orkin, « Applying goal-oriented action planning to games », *AI game programming wisdom*, vol. 2, pp. 217–228, 2003.

[137] A. Ortony, G. L. Clore, and A. Collins, *The cognitive structure of emotions*. Cambridge university press, 1990.

[138] A. Ospina-Bohórquez, S. Rodríguez-González, and D. Vergara-Rodríguez, « On the synergy between virtual reality and multi-agent systems », *Sustainability*, vol. 13, *8*, p. 4326, 2021.

[139] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, « Generative agents: interactive simulacra of human behavior », *arXiv preprint arXiv:2304.03442*, 2023.

[140] J. L. Peterson, « Petri nets », *ACM Computing Surveys (CSUR)*, vol. 9, *3*, pp. 223–252, 1977.

[141] A. Pokahr, L. Braubach, and W. Lamersdorf, « Jadex: a bdi reasoning engine », *Multi-agent programming: Languages, platforms and applications*, pp. 149–174, 2005.

[142] J. Pougala, T. Hillel, and M. Bierlaire, « Capturing trade-offs between daily scheduling choices », *Journal of choice modelling*, vol. 43, p. 100 354, 2022.

[143] J. Pougala, T. Hillel, and M. Bierlaire, « Choice set generation for activity-based models », *in Swiss Transport Research Conference*, 2021.

[144] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, « Virtualhome: simulating household activities via programs », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[145] R. Querrec, C. Buche, E. Maffre, and P. Chevaillier, « Sécurévi: virtual environments for fire-fighting training », *in 5th virtual reality international conference (VRIC'03)*, 2003, pp. 169–175.

[146] M. A. Ramos, V. Munoz-Jimenez, F. F. Ramos, J. R. M. Romero, A. L. Lopez, and B. E. Ordoñez G, « Evolutive autonomous behaviors for agents system in serious games », *in 2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 2015, pp. 226–231.

[147] A. S. Rao and M. P. Georgeff, « Bdi agents: from theory to practice. », *in Icmas*, vol. 95, 1995, pp. 312–319.

[148] J. Renoux and F. Klugl, « Simulating daily activities in a smart home for data generation », *in 2018 Winter Simulation Conference (WSC)*, IEEE, 2018, pp. 798–809.

[149] Q. Reynaud, Y. Haradji, F. Sempé, and N. Sabouret, « Using time use surveys in multi agent based simulations of human activity. », *in ICAART (1)*, 2017, pp. 67–77.

[150] Y. Rizk, M. Awad, and E. W. Tunstel, « Decision making in multiagent systems: a survey », *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, *3*, pp. 514–529, 2018.

[151] A. Roitberg, D. Schneider, A. Djamal, C. Seibold, S. Reiß, and R. Stiefelhagen, « Let's play for action: recognizing activities of daily living by learning from life simulation video games », *in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 8563–8569.

[152] M. J. Roorda, E. J. Miller, and K. M. Habib, « Validation of tasha: a 24-h activity scheduling microsimulation model », *Transportation Research Part A: Policy and Practice*, vol. 42, *2*, pp. 360–375, 2008.

[153] A. Saadi, R. Maamri, and Z. Sahnoun, « A natural inclusion of motives inside bdi agents », *in 2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS)*, IEEE, vol. 1, 2019, pp. 1–5.

[154] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, « Improved techniques for training gans », *in Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 2226–2234.

[155] S. Sardina and L. Padgham, « A BDI agent programming language with failure handling, declarative goals, and planning », en, *Autonomous Agents and Multi-Agent Systems*, vol. 23, *1*, pp. 18–70, Jul. 2011.

[156] M. Schumann, Q. Reynaud, N. Sabouret, F. Sempé, Y. Haradji, B. Charrier, J. Albouys, and C. Inard, « Multi-agent based simulation of human activity for building and urban scale assessment of residential load curves and energy use », *in Building Simulation 2021 Conference*, 2021.

[157] A. Schwartz, « A reinforcement learning method for maximizing undiscounted rewards », *in Proceedings of the tenth international conference on machine learning*, vol. 298, 1993, pp. 298–305.

[158] M. Schweiker, E. Ampatzi, M. S. Andargie, R. K. Andersen, E. Azar, V. M. Barthelmes, C. Berger, L. Bourikas, S. Carlucci, G. Chinazzo, L. P. Edappilly, M. Favero, S. Gauthier, A. Jamrozik, M. Kane, A. Mahdavi, C. Piselli, A. L. Pisello, A. Roetzel, A. Rysanek, K. Sharma, and S. Zhang, « Review of multi-

domain approaches to indoor environmental perception and behaviour », *Building and Environment*, vol. 176, p. 106 804, 2020.

[159] E. de Sevin and D. Thalmann, « A motivational model of action selection for virtual humans », en, *in International 2005 Computer Graphics*, Stony Brook, NY, USA: IEEE, 2005, pp. 213–220.

[160] A. Shirvani and S. Ware, « A formalization of emotional planning for strong-story systems », *in Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, 2020, pp. 116–122.

[161] L. de Silva, « BDI Agent Reasoning with Guidance from HTN Recipes », en, *in Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, São Paulo, Brazil: International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2017, pp. 759–767.

[162] L. d. Silva, F. Meneguzzi, and B. Logan, « BDI Agent Architectures: A Survey », en, *in Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, Yokohama, Japan, Jul. 2020, pp. 4914–4921.

[163] D. E. Smith, J. Frank, and A. K. Jónsson, « Bridging the gap between planning and scheduling », *The Knowledge Engineering Review*, vol. 15, *1*, pp. 47–83, 2000.

[164] S. F. Smith, « Reactive scheduling systems », *in Intelligent scheduling systems*, Springer, 1995, pp. 155–192.

[165] S. F. Smith and M. A. Becker, « An ontology for constructing scheduling systems », *in Working Notes of 1997 AAAI Symposium on Ontological Engineering*, AAAI Press Stanford, CA, 1997, pp. 120–127.

[166] S. Srivastava, C. Li, M. Lingelbach, R. Martin-Martin, F. Xia, K. Vainio, Z. Lian, C. Gokmen, S. Buch, C. K. Liu, S. Savarese, H. Gweon, J. Wu, and L. Fei-Fei, « Behavior: benchmark for everyday household activities in virtual, interactive, and ecological environments », *arXiv preprint arXiv:2108.03332*, 2021.

[167] D. A. Suyikno and A. Setiawan, « Feasible npc hiding behaviour using goal oriented action planning in case of hide-and-seek 3d game simulation », *in 2019 Fourth International Conference on Informatics and Computing (ICIC)*, IEEE, 2019, pp. 1–6.

[168] W. R. Swartout, J. Gratch, R. W. Hill Jr, E. Hovy, S. Marsella, J. Rickel, and D. Traum, « Toward virtual humans », *AI Magazine*, vol. 27, *2*, pp. 96–96, 2006.

[169] I. Trentin, O. Boissier, and F. Ramparany, « Insights about user-centric contextual online adaptation of coordinated multi-agent systems in smart homes », *in Rencontres des Jeunes Chercheurs en Intelligence Artificielle 2019*, 2019, pp. 35–42.

[170] T. Trescak and A. Bogdanovych, « Simulating complex social behaviours of virtual agents through case-based planning », *Computers & Graphics*, vol. 77, pp. 122–139, 2018.

[171] T.-H. Trinh, R. Querrec, P. De Loor, and P. Chevaillier, « Ensuring semantic spatial constraints in virtual environments using uml/ocl », *in Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, 2010, pp. 219–226.

[172] V. Vidal and H. Geffner, « Branching and pruning: an optimal temporal pocl planner based on constraint programming », *Artificial Intelligence*, vol. 170, *3*, pp. 298–335, 2006.

[173] S. Vosinakis and N. Avradinis, « Virtual agora: representation of an ancient greek agora in virtual worlds using biologically-inspired motivational agents. », *Mediterranean Archaeology & Archaeometry*, vol. 16, *5*, 2016.

[174] S. Vosinakis and T. Panayiotopoulos, « Programmable agent perception in intelligent virtual environments », *in Intelligent Virtual Agents: 4th International Workshop, IVA 2003, Kloster Irsee, Germany, September 15-17, 2003. Proceedings 4*, Springer, 2003, pp. 202–206.

[175] K. Walczak, D. Rumiński, and J. Flotyński, « Building contextual augmented reality environments with semantics », *in 2014 International Conference on Virtual Systems & Multimedia (VSMM)*, IEEE, 2014, pp. 353–361.

[176] C. J. Watkins and P. Dayan, « Q-learning », *Machine learning*, vol. 8, pp. 279–292, 1992.

[177] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah, « Declarative & procedural goals in intelligent agent systems », *KR*, vol. 2002, pp. 470–481, 2002.

[178] W. Wobcke and A. Sichanie, « A reactive scheduling agent architecture for coordinating autonomous assistants », *in Intelligent agent technology: Systems, methodologies, and tools*, Citeseer, 1999.

[179] M. Xu, K. Bauters, K. McAreavey, and W. Liu, « A formal approach to embedding first-principles planning in bdi agent systems », *in International Conference on Scalable Uncertainty Management*, Springer, 2018, pp. 333–347.

[180] D. Yan, W. O'Brien, T. Hong, X. Feng, H. B. Gunay, F. Tahmasebi, and A. Mahdavi, « Occupant behavior modeling for building performance simulation: current state and future challenges », *Energy and buildings*, vol. 107, pp. 264–278, 2015.

[181] Y. Zhao, F. F. Pour, S. Golestan, and E. Stroulia, « Bim sim/3d: multi-agent human activity simulation in indoor spaces », *in 2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)*, IEEE, 2019, pp. 18–24.

[182] V. A. Ziparo, L. Iocchi, P. U. Lima, D. Nardi, and P. F. Palamara, « Petri net plans: a framework for collaboration and coordination in multi-robot systems », *Autonomous Agents and Multi-Agent Systems*, vol. 23, pp. 344–383, 2011.

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

INSA RENNES

Titre : Simulation d'Agents Autonomes dans des Environnements Virtuels Connectés

Mot clés : Agents Autonomes, Génération de données synthétiques, Activités quotidiennes, Environnements Virtuels Connectés, Intelligence Artificielle, Ordonnanceur, Modèle BDI

Résumé : Comprendre le comportement humain est essentiel pour développer des systèmes adaptatifs tels que des environnements connectés. Des données réelles sont alors nécessaires, mais elles sont rares et coûteuses à produire. Une solution consisterait à générer des données synthétiques en simulant les environnements et leurs occupants. Cependant, nous devons nous assurer qu'elles sont suffisamment crédibles pour être utilisables. Cette crédibilité va notamment dépendre de celle des humains simulés et de leur capacité à interagir avec leurs environnements virtuels pouvant produire de telles données. Cette approche soulève alors la question suivante : Comment simuler des humains virtuels capables de produire des comportements crédibles tout en étant compatible avec la génération de données ? Pour répondre à cela, 3 défis majeurs doivent être adressés : (1) Produire automatiquement des comportements humains crédibles, (2) Offrir un compromis entre contrôle et autonomie sur les comportements, et (3) Valider la crédibilité des données et des comportements. Pour relever ces défis, nous proposons un modèle d'agent autonome capable de fournir des comportements à la fois autonomes et contrôlables tout en permettant l'exécution d'activités dans des environnements 3D. Nous proposerons différentes méthodes validant la crédibilité des données et des comportements.

Title: Simulating Autonomous Agents in Connected Virtual Environments

Keywords: Autonomous Agents, Synthetic data generation, Daily-activities, Connected Virtual Environments, Artificial Intelligence, Scheduling, BDI Model

Abstract: Understanding human behavior is essential to develop adaptive systems such as connected environments. To do this, real data are required but they are scarce and costly to produce. One solution would be to generate synthetic data by simulating the environments and their occupants. However, we need to ensure that these data are sufficiently credible. This credibility particularly depends on the simulated virtual humans' credibility and their ability to interact with virtual environments producing such data. This approach raises the following question: How can we simulate virtual humans able to produce credible behaviors while being compatible with data generation? To address this, 3 major challenges must be addressed: (1) Automatically producing credible human behaviors, (2) Offering a compromise between control and autonomy over behaviors, and (3) Validating the credibility of data and behaviors. To address these challenges, we propose an autonomous agent model providing both autonomous and controllable behaviors, while enabling activity execution in 3D environments. Various validation methods are also proposed to assess the credibility of the behaviors and synthetic data.