International Conference on Artificial Reality and Telexistence
Eurographics Symposium on Virtual Environments (2019)
Y. Kakehi and A. Hiyama (Editors)

Model and Tools for Integrating IoT into Mixed Reality Environments: Towards a Virtual-Real Seamless Continuum

J. Lacoche[1], M. Le Chenechal[2], E. Villain[1] and A. Foulonneau[1]

[1]Orange Labs Rennes [2]OpenMind Innovation

Abstract

This paper introduces a new software model and new tools for managing indoor smart environments (smart home, smart building, smart factories, etc.) thanks to MR technologies. Our fully-integrated solution is mainly based on a software modelization of connected objects used to manage them independently from their actual nature: these objects can be simulated or real. Based on this model our goal is to create a continuum between a real smart environment and its 3D digital twin in order to simulate and manipulate it. Therefore, two kinds of tools are introduced to leverage this model. First, we introduce two complementary tools, an AR and a VR one, for the creation of the digital twin of a given smart environment. Secondly, we propose 3D interactions and dedicated metaphors for the creation of automation scenarios in the same VR application. These scenarios are then converted to a Petri-net based model that can be edited later by expert users. Adjusting the parameters of our model allows to navigate on the continuum in order to use the digital twin for simulation, deployment and real/virtual synchronization purposes. These different contributions and their benefits are illustrated thanks to the automation configuration of a room in our lab.

CCS Concepts

• ***Human-centered computing*** → ***Interactive systems and tools;*** *HCI theory, concepts and models; Virtual reality;*

1. Introduction

The emergence of Mixed Reality (MR) technologies, which include Augmented Reality (AR) and Virtual Reality (VR), provides more affordable hardware and improves the acceptability of such devices to the consumer market. This growing market leads to a dissemination of these technologies into companies of any size as well as in the users' own homes. Meanwhile, the Internet of Things (IoT) is a pretty dense network of physical objects connected over the Internet that are able to sense the real world and to act on it [Kop11]. As these objects and their features are ramping up while becoming cheaper [JJH*15], they tend to transform our world into a composition of smart environments: smart-homes, smart-buildings, smart cities. A smart environment is defined as an environment able to acquire and apply knowledge about the environment and its inhabitants in order to improve their experience in it [CD07]. Particularly, connected objects can offer services to automatize some users' recurrent behaviors in order to save them time and efforts, and assist them in different tasks in such smart environments.

MR and IoT fields share a common aspect that is to propose a way to build a connection between the real world and a virtual version of it: its digital twin [GS12]. While MR focuses more on interactions and rendering, IoT allows the automation of routines that act seamlessly from the user's point of view. However, the IoT setup process in smart environments remains pretty complex to achieve as it requests a deep understanding of the system in order to create desired behaviors. Nowadays, only desktop applications are available to achieve this goal, which increases the cognitive load of users. Indeed, they have to map identifiers, without any spatial references, to real connected objects located in their smart environment. We state that MR is able to solve this issue by providing an immersive way to manage IoTs behaviors in smart environments.

Our goal is to create a continuum between a real smart environment and its digital twin using MR technologies. The digital twin aims to act as a mediation tool between a user and the real smart environment he/she wants to manage. We propose to use the digital twin of a real environment for the configuration of its behaviors and for their validation and deployment. Our contributions focus on automation behaviors but they could also target the creation of interaction behaviors. Automation behaviors differ from interaction behaviors as their progress does not require an explicit user command such as a gesture or a voice command. To do so, we introduce a software model based on the multi-agent model Presentation-Abstraction-Control (PAC) [Cou87] that allows a seamless management of simulated and real connected object. This model is leveraged by two kinds of tools in order to provide a full pipeline for managing a smart environment. For now, these tools focus on indoor smart environments but they could be extended later to support the outdoor ones as well. These tools address application de-

DOI: 10.2312/egve.20191286

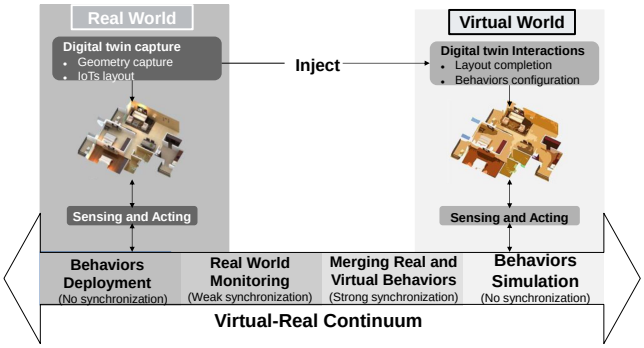


Figure 1: *A digital twin of a smart environment is captured in AR and configured in VR. Our software model described in this paper allows to navigate on the virtual-real continuum to exploit this digital twin for multiple purposes.*

velopers and designers as well as architects and building managers. First, we propose a Google Tango based application to capture the geometry and the positions of the connected objects of a given environment. Then, a VR editor allows to load the result of this capture (or another geometry file) in order to complete its configuration with additional connected objects and furniture. Then, within the same VR application, automation scenarios can be created thanks to 3D interactions and to a dedicated visual metaphor based on 3D jacks connectors and sockets. This VR editor generates a Petri-net based scenario that can be edited later by expert users. Once the capture and configuration steps are performed, our software model allows to navigate on the real virtual continuum in order to define how much the real world and its digital twin are linked. Indeed, this property of our model can be used to move from purely simulations to real deployments, through real and virtual worlds synchronization. This continuum and our pipeline for the creation of a digital twin are detailed in Figure 1. To illustrate these different contributions, the different examples that we give in this paper are based on the management of a room in our lab equipped with traditional automation objects: a plug connected to a lamp, a siren, a smoke sensor, a motion sensor, a temperature sensor, and a door sensor.

Our paper is structured as follows, Section 2 presents some related work. Then, Section 3 introduces our software model dedicated to the implementation of connected objects proxies. Section 4 presents our tool to capture and configure a 3D smart environment while Section 5 describes our approach for the creation of automation behaviors and how they can be simulated and deployed. Finally, we conclude and give some opportunities for future work.

2. Related Work

The idea to create a continuum between the real and the digital world has already been explored. Lifton and Paradiso [LP09] introduced the concept of dual reality that aims to merge the real and the virtual world by means of networked sensors and actuators. It allows reflecting changes made into the virtual world in the real one and vice versa. These solutions can be classified into two categories. First, some solutions propose to use virtual environments (VEs) for prototyping connected objects behaviors in the real world. Secondly, other solutions focus on using VEs for simulating connected objects behaviors. The goal of such a simulation is to validate these behaviors before deployment and to generate data.

Prototyping Smart-Environments

AVIot [JJH*15] is a web-based interactive authoring and visualization tool of indoor smart environments. It proposes the user to import the model of his/her environment or to select in a database a 3D model close to it. Automation behaviors can be configured with a scripting language or with visual programming thanks to a combination of boolean events and actions. For now, it does not propose any solution for linking the prototyped smart environment with the real world. Similarly, Belluci et al. [BZDA17] propose X-Reality toolkit for the rapid prototyping of smart environments dedicated to novice users based on the configuration of a 3D digital twin of the real environment. It includes a virtual world 2D editor, a state chart editor to program behaviors of virtual and physical elements and a runtime execution server for deployment. X-Reality toolkit allows the communication between virtual and physical objects opening perspectives for real-time monitoring and simulation. To continue, Pfeiffer et al. [PPL18] introduce a pipeline for the creation of MR interactions with IoTs. An application designer has to capture a 360 panorama of the real environment first and then configure it in VR thanks to pre-developed visual augmentations and behaviors. Automation behaviors are programmed thanks to Node-RED (`https://nodered.org/`) and interaction events are published using MQTT [MQT14]. With this process, they can target both simulated and real connected objects. It allows a user to prototype its MR application in a 360 panorama then deploy it in the real world. Finally, Reality Editor [HHM13] is a a handheld AR interface that allows users to create automation behaviors by linking objects between them directly in the real world.

Simulating Smart-Environments

Simulation of smart environments can be used to validate predefined services and behaviors before deploying them in the real world. UbiReal [NYT*06] is a solution for simulating smart environments in order to ease the development of ubiquitous applications. It includes a graphical user interface to design 3D indoor smart environments and the possibility to visualize devices states with 3D animations during the simulation. System testing can be performed to determine if an application works as expected in a variety of contexts thanks to the simulation of wired and wireless communication, the emulation of physical quantities (temperature, humidities, etc) and the simulation of users behaviors. Similarly, V-PlaceSims [LCK08] is a simulation tool that allows an architect to design and configure a smart home from a floor plan. It is developed as an online VE platform for design collaboration between the architects and potential inhabitants. In order to provide feedback to the architects, end users can explore the space thanks to 3D avatars to check how the environment looks like, and also how it reacts to their behaviors thanks to a real-time simulation. Simulation of smart environments can also be used for generating data of connected objects. Indeed, such data can be exploited to perform classification and recognition of activities of daily living. In that category, Ariani et al. [ARCL13] propose a smart home simulation used to generate data of ambient sensors. It is based on a 2D editor to design a floor plan and to configure its sensors. Then, they simulate activities in the smart home thanks to the A* pathfinding

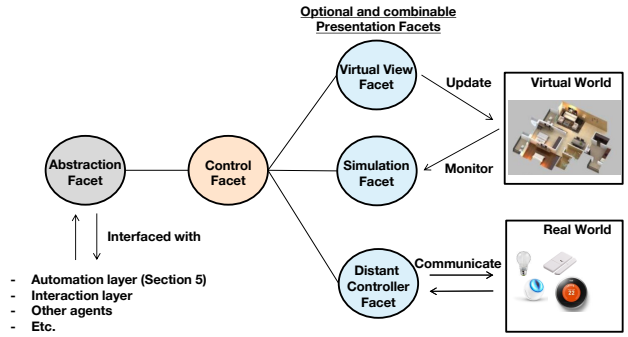


Figure 2: *Our PAC [Cou87] based model dedicated to the manipulation of simulated and real connected objects*

algorithm [HNR68]. Park et al. [PMBK15] developed a 3D smart home simulator with Unity3D (https://unity3d.com/). In this simulator, the user controls a virtual character with the mouse and the keyboard in a VE equipped with virtual ambient sensors in order to generate data. Last, OpenSHS [AAS*17] is an open source 3D simulation tool developed with Blender (https://www.blender.org/) for smart home dataset generation. It includes a library of devices such as doors and motion sensors, appliances switches and light controllers. The design of the VE and the positioning of these sensors is performed in Blender. Data are generated by a user interacting with the VE and with a replication algorithm dedicated to the generation of large datasets.

Summary

For now, none of the different prototyping solutions propose a full workflow to both simulate the created behaviors and monitor their execution in the real world. Also, simulation solutions do not provide any link between the simulation and the real world. Most of the times the geometry of the simulated VE is not based on the real world. Also, in those solutions, the simulated behaviors cannot be replicated automatically in the real world. Overall, we believe that these solutions lack of a common interface for simulated and real connected objects in order to address with the same approach the processes of prototyping, simulating, deploying and monitoring. Moreover, the VEs and behavioral editors provided by the different tools are mainly based on 2D interactions while 3D interactions could ease the different processes by giving more spatial cues to the end users. Regarding the design of the manipulated VEs, the link with the real world could be more developed thanks to 3D capturing tools or simply by supporting the import of common CAD formats. Our contributions aim to address theses current limitations.

3. PAC-based Virtual and Real IoT model

Our goal is to provide developers with a model for the development of proxies used to monitor and manipulate connected objects that can be virtual (simulated), real or both. This is motivated by our need to create a continuum between the real world and its digital twin. Indeed, when a developer implements automation and interaction behaviors involving a given smart environment, an ideal workflow must allow him to validate them in its digital twin, then deploy them in the real world and last monitor their progress by dynamically coupling the real world and its digital twin. These steps correspond to different positions on the real virtual continuum described in Figure 1. Regarding these needs, such a model must:

- **R1:** Ensure that a proxy can be linked to a virtual object, to a real one or to both at the same time. This configuration must be possibly modified in order to allow the developer to navigate on the virtual real continuum (switch from simulation, to deployment, to synchronization).
- **R2:** Allow the definition of high-level behaviors that can be applied seamlessly to real and virtual connected objects.
- **R3:** Allow the communication between connected objects independently of their nature (real, virtual or both).

In the field of Human Computer Interaction (HCI), multi-agent models propose to represent an application as a collection of specialized computing units: the agents. This approach is particularly interesting in the field of smart environments as such an environment can be considered as a collection of connected objects that can communicate. Thus, our model is a specialization of a commonly used multi-agent model: Presentation-Abstraction-Control (PAC) [Cou87]. PAC decomposes an agent into three facets: the *Abstraction* holds the semantics and the main features of the agent. The *Presentation* facet manages both the inputs and the graphical representation of the agent. The *Control* facet ensures the consistency between the Presentation and Abstraction facets thanks to a synchronization mechanism. Comparing to other agent models such as Model-View-Controller (MVC) [KP*88], PAC allows a better decoupling between the functional features and the input and output features of an agent thanks to its Control facet. This is particularly needed in our case and this is why the model that we propose is based on PAC. Moreover, recently, specializations of the PAC model were provided in order to suit the needs of 3D user interfaces. First, PAC-C3D [DF11] ensures a strong decoupling between the core functions of a VE, its representations, and its collaborative features. Then, the model proposed by Lacoche et al. [LDA*15] introduces an additional presentation facet, the *Logical Driver*, dedicated to handle particular input and output devices.

In order to fulfill our different requirements, the model that we propose, described in Figure 2, decomposes the Presentation Facet of PAC into three different facets:

- The **Virtual View** aims at rendering (visual, audio, etc.) a connected object in the virtual world. It handles its 3D geometry and may update it according to the modifications of the agent state. For instance, for a light object, it will create a virtual light in the scene and manage its activation status, its color, its intensity, etc. Regarding the siren that is available in the room that we want to manage, this facet can play a sound in the virtual world.
- The **Simulation Facet** aims at simulating the behavior of a connected object. It can report properties modifications and commands to the agent according to its observation of the virtual world. This simulation focuses on the functional aspects of a given object and not on low level factors such as network communications. The realism of the simulated behavior depends on the quality of the implementation of this facet made by the developer. For instance, for the simulation of a motion sensor, it will use a physics engine to detect a collision between a virtual human and its detection area combined with a ray-cast to check

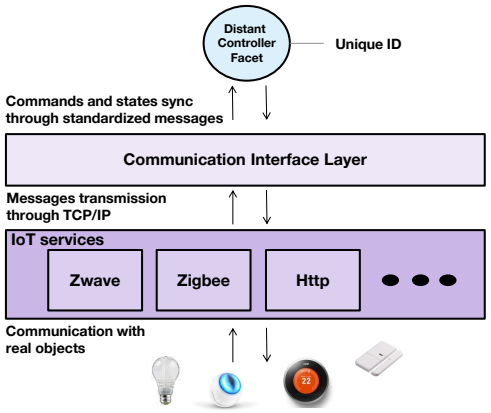


Figure 3: *Distant Controller facets communicate with real objects thanks to a communication layer that is interfaced with IoT hardware services. Each real object is configured with a unique id.*

if this virtual human is in its line of sight. Regarding a temperature sensor, such value cannot be observed directly from the virtual world. In our simulation tool described in Section 5.2, the temperature is set in VR by the end user. Such value could also be retrieved from a web service such as OpenWeatherMap (https://openweathermap.org/).

- The **Distant Controller** aims at synchronizing the state of the agent and the state of a distant real object. It can report properties modifications and send commands from the agent to the real object and vice versa. In our implementation, it does not manipulate directly a physical hardware with a particular SDK. Indeed, as described in Figure 3, it exchanges standardized messages with a communication layer interfaced with IoTs hardware services through TCP/IP. Each of the real objects detected by these services needs to be registered in the communication layer with a unique id. The Distant Controller facet needs to be configured with its id to be able to communicate with a particular object. For now, it is interfaced with a service that can communicate with real objects through the Zwave (https://z-wavealliance.org/) protocol. Other protocols could be integrated later such as HTTP or Zigbee (https://www.zigbee.org/). The implementation details of this communication layer, those IoT services and the discovering of objects by these services are out of the scope of this paper.

For a given agent, these facets are optional, switchable between sessions, and also combinable in order to cover **R1**. Indeed, it means that an object can be either fully virtual or fully real but also virtual and real simultaneously. For instance, if a Virtual View facet and a Distant Controller facet are deployed for a given object, the virtual aspect of the object is automatically synchronized with its aspect in the real world. For a temperature sensor, the value displayed in the real world could be automatically displayed in the virtual world. In addition, if a Distant Controller facet and a Simulation Facet are deployed at the same time, an object could react to both virtual and real worlds. As an example, a motion sensor could detect real persons as well as virtual humans moving in the digital twin of the real world. Additional examples are provided in Section 5.2.

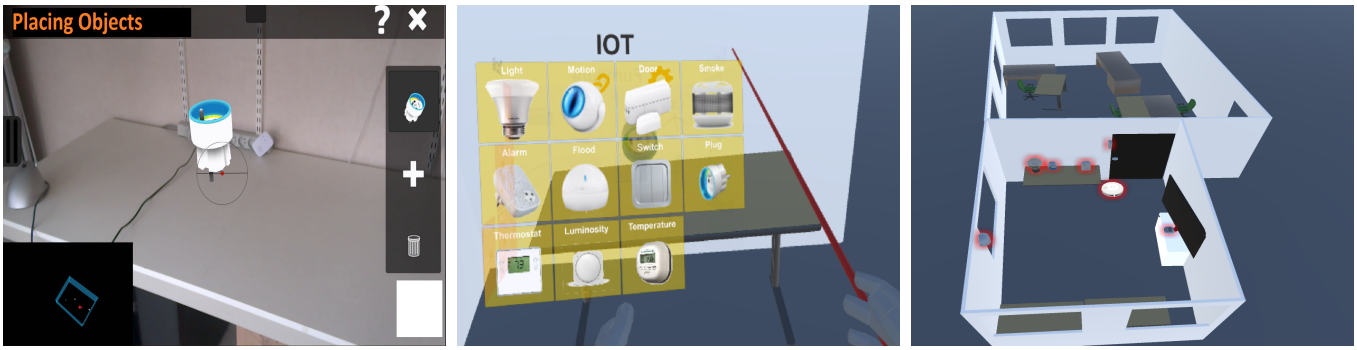
Regarding the other facets, the role of the **Abstraction** facet globally remains the same. It contains and gives access to the properties and the commands of an object. It can be interfaced with the processing layers (interaction, automation, etc.) of the application. Communications between different connected objects are also performed through this facet. In the original version of PAC, these accesses are made through the Control facet. Our implementation allows a direct access to the Abstraction facet for simplicity reasons. Indeed, this facet gives a unique interface for these properties and commands independently of the actual nature of the manipulated object. The automation tool presented in Section 5 directly exploits this capacity of our model. These aspects of the Abstraction facet allows the model to cover **R2** and **R3**. Finally, the **Control** facets keeps the same role as in the original PAC model. It ensures the consistency between the Abstraction and the Presentations facets. It ensures that when a property is modified or when a command is called in the Abstraction facet, the Presentations facets are correctly notified. A property modified by a Presentation facet is also reflected on the Abstraction facet thanks to it. In our implementation it can also handle concurrency between the different Presentation facets. For instance, if both Simulation and Distant controller facets are deployed at the same time, they can provide opposite commands and properties in some particular cases. Therefore, in our implementation the application developer can configure which facet to prioritize (we do not propose merging capabilities yet).

An agent created with this model can encapsulate all features of a connected object or can decompose an object with multiple features into multiple agents. For instance, a Fibaro Motion Sensor Fgms-001 (https://www.fibaro.com/en/products/motion-sensor/), that is available in our lab, can perceive motions as well as the luminosity and the temperature of a room. Our model can support both approaches but in our implementation we choose the second one in order to propose a generic library that does not rely on current particular hardware.

Using this model we have created a library of objects developed in C# for Unity3D. It contains different types of ambient sensors such as motion, door, temperature, humidity, luminosity, smoke and flood sensors. It also contains different kinds of appliances such as light controller, plug, alarm, thermostat. This library includes the different objects contained in the room that we want to manage. Our implementation also integrates an XML configuration file to setup the objects of a given environment. It can be used to edit their position, their id and to determine which facets of each agent need to be instantiated. This library is directly exploited by the tools presented in Section 4 and 5. In the next Section, we describe a tool able to generate the XML configuration file of a given environment.

4. Creating a Digital Twin of a Real Smart Environment

In this Section, we propose an AR tool to create a digital twin of an indoor smart environment. It is defined by the geometry of the real environment and by the connected objects it contains. These objects are taken from the library implemented using our software model. Once this digital twin is captured in AR, it can then be completed in a VR application. This digital twin aims to act as a mediation tool between the user and the smart environment he/she wants to manage. Our goal is to provide the user with an easier and more natural way to manage it compared to classical desktop-based interfaces.



(a) (b) (c)

Figure 4: *Creating a digital twin of an indoor smart environment. (a) We use a Google-Tango to capture the geometry of an environment and the arrangement of its objects. The geometry is visible at the bottom left. (b) Connected objects and furniture are added in VR using a ray-based interaction technique and a menu attached to the non dominant hand of the user. (c) The digital twin of our lab. Connected objects are highlighted in red: a temperature sensor on the right, a smoke sensor on the ceiling, a plug, a siren, a lamp and a door sensor at the top.*

4.1. AR Environment Capture

We propose an AR tool to capture the geometry of an indoor smart environment and the arrangement of its objects. Indeed, most of the times the end user does not have a 3D model of it. This is especially true for smart homes. It is less true for smart buildings that often have an associated Building Information Modeling (BIM) File.

This tool is a Google-Tango based application. The Tango tablet provides an AR display with 6 Degrees of Freedom (DoFs) tracking and a depth camera that is able to reconstruct the real environment and output the 3D mesh of it in real-time [Gül16]. However, from our perspective, the current Tango 3D reconstruction feature generates a fairly accurate mesh. This mesh also lacks of semantic information that could be useful to identify which part of the mesh is a wall or the floor. Thus, rather than using the 3D reconstructed mesh, we choose another approach in order to capture the geometry of the environment. Indeed, the device is able to estimate a 3D plane regarding the current captured point cloud. We leverage this feature to define the geometry of the rooms by asking the user to target the floor, then the walls and click on the touch-screen for each of these parts in order to make the application constructs a clean 3D mesh of the room geometry as shown in Figures 4a and 4c. This process can be performed for a whole floor of a real environment. When a user selects the same wall in two connected rooms, tracking drifting issues can make them have a slightly different estimated position. In that case, we perform a merging process for these walls. Opening such as doors and windows can also be configured. For a given opening, the user has to point its 4 corners (more complex openings are not supported yet). Then, we use raycasting with the previously defined planes to determine the world position of each corner. Finally, the same approach is used to determine the connected objects positions in the environment. In this last step, the user targets each of these objects, chooses which object he currently points to in our library of objects, and touches the screen to define the 3D placement of the pointed real object. This process is illustrated in Figure 4a where the user places a connected plug in our lab. Raycasting is also used to determine their 3D position. The results of this tool are an OBJ file for the geometry of the environment and a first version of the XML file that contains the connected objects parameters.

This approach gives a simple version of the geometry of the real environment without any texture information but with identified structures (floor, walls, openings) and with the appropriate dimensions as shown in Figure 4c. Before using it in the next steps of our pipeline, an environment captured with this tool can be edited in 3D modeling tool or directly in Unity3D. This allows to reach a higher level of realism, to add more complexes structures or to assemble multiple captured floors. For instance, the door that separates the two rooms in Figure 4c has been added in Unity3D.

4.2. VR Environment completion

In that second step, the user edits in VR a smart environment, captured with the previously described tool, or loaded from an existing model in a particular format such as FBX or IFC. Indeed, this VR approach is useful for managing a building that is not available for an AR capture. For instance, an architect could want to configure the automation behaviors of a building before its construction. In that case, for now, this loaded model needs to be manually edited in Unity3D in order to add colliders to walls and floors.

The goal of this second step is to complete the information of the 3D digital twin of the environment by adding additional connected objects and by adding custom furniture from a list of predefined 3D models. This VR tool is also used to select the unique identifier of each object. This is required to allow each object to communicate with an IoT service through our communication layer as described in Section 3. Adding furniture is optional as it is used to create a more accurate digital twin. These furniture can be important if we want to simulate a virtual agent navigating in the digital twin.

This VE is implemented in Unity3D and with SteamVR (`https://steamcommunity.com/steamvr`) in order to be compatible with multiple headsets. It requires two 6DoF controllers to be usable. The same VE is used in the automation tool described in Section 5. This VE uses an asymmetric bimanual interaction paradigm based on the use of a 3D ray in the dominant hand that is able to select and manipulate close and distant objects without the need to move. Moreover, this 3D ray can be used to define a spot on the floor to teleport the user providing him large-scale naviga-

tion capability. Furthermore, a hand-held 3D interface is carried by the non-dominant hand and displays a rotating menu with different panels to add objects in the scene: connected objects and furniture. For instance the panel for adding connected objects is shown in Figure 4b. Furniture 3D models are loaded from a dedicated folder. It allows the end user to add his/her own objects that correspond to their environment. Last, a dedicated 3D interface with a virtual keyboard is also provided to configure the id of a given object.

At the end of this step, the digital twin of a smart environment is completed. For instance, the digital twin of our lab is depicted in Figure 4c. This digital twin is ready to be used for editing its behaviors and to simulate and deploy them as described next.

5. Automation configuration, simulation and deployment

In the field of smart environments, one of the main uses of connected objects is the creation of automation scenarios in order to assist building occupants in their daily routines or for security reasons. Current approaches for creating such automation scenarios are mainly dedicated to expert users and are often based on desktop interfaces that lack of context information. For instance, it can be easier to find some objects placed in an indoor environment using real spatial cues instead of not so-meaningful identifiers.

Regarding the automation of our lab, our need is to define three different automation scenarios

- Everyday life scenario : when the door is opened and the motion sensor detects someone, the plug is activated in order to switch on the lamp. When the motion sensor detects no one, the plug is disabled.
- A security scenario: when the smoke sensor is triggered, the alarm needs to be activated.
- An anti-intrusion scenario: when the door is opened during night hours the alarm needs to be activated.

We propose to edit such behaviors in VR in the digital twin of the real environment. Then, we demonstrate how these behaviors can be simulated and deployed using this VE and the configuration possibilities offered by our software model.

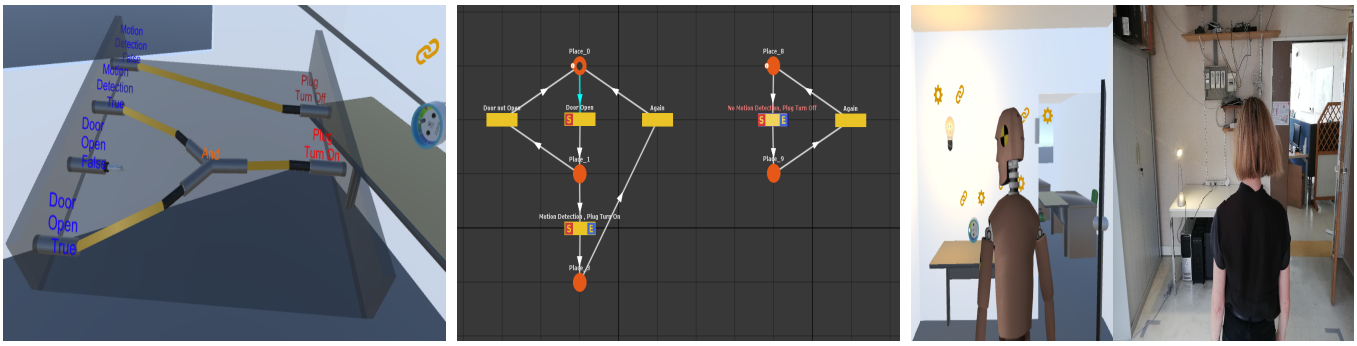
5.1. Creation of automation behaviors

Our approach exploits the VE described in the previous Section. Automation behaviors can be created after the digital twin configuration is complete. The main purpose of this work is to explore novel 3D interaction metaphors that are able to create complex links between connected objects defining behavioral scenarios in an easier way than classical desktop interfaces. We still provide expert users with such a desktop interface in order to add more complexity to the behaviors created in VR. This 2D editor relies on the concept of Petri-nets [Mur89].

The process of creating a behavior in this VE is composed of two steps. The first step is to simply link objects in order to begin the definition of a specific behavior between sensors and actuators. Then, in a second step, these links must be configured to precisely create the mapping that defines which input triggers which output. These mappings are made between properties (inputs) and functions (outputs) that are exposed by the Abstraction Facet of our model introduced in Section 3. This aspect of our model allows to check a connected object state or to trigger one of its actuators independently of its real nature: real or simulated. Therefore, the automation behaviors created with this tool can be deployed seamlessly to a real smart environment or to a simulation.

Regarding the first step, in order to initialize a link between two objects of the scene, the user has to point to one object after the other with the 3D ray while pressing the selection button (in our case: the trigger). A 3D straight line is displayed in the VE to materialize the created link. Then, in order to configure the mapping between the sensors and actuators of multiple linked objects, we propose a world-anchor link box metaphor. The main idea of this metaphor is to represent objects' sensors as jacks connectors and actuators as sockets in which jacks can be plugged. Also, these jacks wires can be grabbed and moved by the user with the 3D ray. For a given object, sensors correspond to properties in the Abstraction facet and actuators to functions. In our implementation, they are extracted from the Abstraction class using C# custom attributes. Compatible properties include boolean and number (float, int, etc.) values. For now, actuators correspond to functions without attributes. While boolean properties can be directly mapped to an actuator, number properties need to be converted. Therefore, we provide multiple 3D operators (one input socket, one output jack) to convert a number property to a boolean one. These operators are LESS, GREATER and EQUALS. The comparison value can be increased and decreased by the end user with two 3D buttons. We also introduce 3D operator components to manage logical links, e.g AND and OR relations, that expose two sockets and one jack in order to combine both inputs into a single output. These components can be recursively plugged in order to possibly make an infinite combination of logical links. The link box can also be manipulated, and the user can place it in the VE to organize his/her space. It can also be hidden to avoid visual overloads. We provide an example of a link box in Figure 5a. It corresponds to the everyday life scenario described in the introduction of this Section. In this link box, the "Turn On" function of the plug is connected, through an AND component, to the "Open" property of the door sensor and to the "Presence detected" of the motion sensor. The "Turn Off" function of the plug is connected to the "No Presence detected" of the motion sensor. This scenario controls the behavior of the lamp as this one is connected to the plug in the real world.

Once completed, a scenario is saved in a Petri-Net based scenario model based on #SEVEN [CGBA14]. We use an implementation of these concepts provided by a third party. As an example, the corresponding Petri-Net of the behavior detailed in Figure 5a is presented in Figure 5b. It is composed of two Petri-nets. The left one checks if the door is open and if the motion sensor detects someone before turning on the plug (and therefore the lamp). It corresponds to the "AND" relation. The right one checks at each processing step if the motion sensor does not detect anyone. In that case, it turns off the plug. In #SEVEN, the context is checked thanks to sensors components in order to progress in the Petri-Net. When such progress is made, an action can be triggered thanks to effectors components. These components exploit the Abstraction facet of our model in order to check the properties of an object and to trigger its commands. This approach has three advantages. First, Petri-Net is a widespread concept in the field of automation. It eases the understanding of a



(a) (b) (c)

Figure 5: *Automation of a smart environment. (a) Objects sensors and actuators are connected with a link box metaphor. Here, the plug is enabled when the door is open and the motion sensor detects someone (AND relation). It is disabled when the motion sensor detects no one. (b) The behaviors are converted in a Petri-Net based scenario model. The two Petri-Nets correspond to the two links shown in the first figure. (c) Using software model, the same automation scenario can be used for simulation and deployment purposes. On the left, the plug (and so the lamp) is enabled when the agent enters the virtual room. On the right, the real lamp is enabled when a real user enters the real room.*

scenario for developers and designers. Second, the implementation of #SEVEN that we use provides an execution engine for running the created scenarios at runtime. Third, this implementation also provides a 2D editor integrated in Unity3D to edit the scenarios generated by the VR tool. It allows expert users to define more complex relations between the objects than just mappings between sensors and actuators. Regarding the anti-intrusion behavior that we want to be deployed in our lab, it cannot be configured in VR as we do not provide visual components to check the hour yet. Only a single mapping between the motion sensor and the alarm can be performed in VR. Then, a sensor can be added to the generated Petri-net in the 2D editor to check the current time before checking the motion sensor state. For now, such a modified scenario cannot be edited back in VR as we do not provide equivalents 3D components to all #SEVEN sensors and effectors components.

5.2. Behaviors Simulation, Deployment and Monitoring

Once the different automation behaviors have been designed they can be used in a coordinated way with the digital twin. As described in Section 3, by editing the XML configuration file of the digital twin, it is possible to select the facets that are instantiated for each connected object (The Abstraction and Control facets are always instantiated). Exploiting this digital twin and its behaviors is performed in the same VE but the application needs to be restarted when changes are made in this XML file. This property of our model allows developers and designers to easily navigate on the virtual-real continuum giving them multiple perspectives.

First, the digital twin can be used for **Simulation** purposes in the same VE. Such a simulation can be used to validate the behaviors before deploying them in the real world. For instance, for a building that is not constructed yet, automation behaviors can be defined and validated in order to be ready at the end of the construction process. Here, only the Simulation and Virtual facets of each agent are instantiated. Then, each object only reacts to the virtual world. For this simulation, we have integrated a dedicated panel with multiple tools in the 3D menu attached to the user's hand. For instance, the user can target a virtual agent present in the VE and selects its destination with its 3D ray. This agent then walks to this destination. This agent can be used to validate that our light is correctly turned on when someone enters the room of our lab as shown in Figure 5c. Virtual smoke can also be added in the VE allowing us to validate our security scenario described at the beginning of this Section. Simulation is particularly relevant in that case as such event could be difficult to reproduce in the real world. In addition, it is possible to validate our anti-intrusion scenario by setting the hour in the simulation panel and by controlling the agent. This panel allows the modification of the room temperature as well. Additional simulation tools will be added in the future to support other use-cases.

Second, **deployment** in the real world of these behaviors can also be performed. Here, only the Distant Controller facets are instantiated. The digital twin is then passive and not synchronized. The VE is not relevant and 3D rendering can be disabled. Only the Petri-Net behaviors are executed. As explained, communications between these behaviors and real objects are abstracted by the Abstraction and Distant Controller facets of each agent. As an example, Figure 5c demonstrates that our scenario is correctly executed in the real world. The door and motions sensors react to a real person entering the room and the plug and the lamp are enabled.

Third, the digital twin can also be used for **monitoring**. When the Distant Controller and the Virtual View facets are instantiated for a given object, its rendering is synchronized with its real world properties as described in Section 3. In that case, the behaviors are still deployed in the real world and someone immersed in the VE can monitor their progress. This capability of our model allows to control the state of a smart environment in real-time in the VE. For instance, in our case, the virtual door is rendered as open when the real one is open and the virtual temperature sensor displays the temperature captured in the real world by the real one. The virtual and the real lights are also synchronized even if this lamp is not a connected object. Its status depends on the plug. To maintain this synchronization, a link must be defined with our automation tool between the state of the plug and the activation status of the lamp.

Last, **merging simulation and deployment** features is also possible. Indeed, by instantiating all types of facets, a given object can

be rendered in the VE and can react and act both on the virtual and the real worlds. It gives users interesting capabilities to impact the real world from the virtual one and vice versa. It corresponds to the concept of Dual Reality introduced by Lifton and Paradiso [LP09]. For instance, as it is possible to add a virtual smoke in the VE, a building manager could use this feature for a fire drill in the real world. Similarly, in our case the virtual agent could be used to turn on the light in the real world in order to fake a human presence.

6. Conclusion and Future Work

To conclude, in this paper, we propose a pipeline for managing smart environments. It includes a software model for the creation of proxies for simulated and real connected objects. This model is leveraged by an AR tool to capture the geometry and the object configuration of an environment and by a VR tool to configure its behavior. This VR tool is completed by Petri-net editor taken from the state of the art in order to allow expert users to complexify the created behaviors. Using these tools combined with our model, we demonstrate that a digital twin of a smart environment can be created and used for simulation, deployment and real/virtual synchronization purposes. We strongly believe that these MR tools propose an easiest and more efficient way to manage smart environments by involving more natural interactions and more spatial cues.

As future work, our tools need to be evaluated by developers and designers in order to demonstrate this supposed increased efficiency. For that purpose, we plan to compare the performances of our automation editor with a commonly used tool such as Node-RED. Our tools need to be completed and tested on different types of building to provide a fully integrated solution. Indeed, as detailed through the paper, some steps still need to be performed manually in Unity3D or in a 3D modeling tool. Our current focus is to add the possibility to navigate on the real virtual continuum directly at runtime to provide users with seamless sequential prototyping scenarios. We also plan to add an object recognition step in our AR capture tool in order to automatize the placement of objects. Deep learning approaches could be investigated to do so. For instance, Mesh-RCNN [GMJ19] could be used to automatically fulfill the captured environment with its furniture. The VR automation tool could also be transposed in AR with an adapted device such as the Magic Leap One glasses (`https://www.magicleap.com`) as they also include plane detection and a 6doF controller for interactions. Finally, our VE could also be used to prototype interactions with connected objects as in [PPL18]. These interactions could then be tested in VR before being deployed with AR.

References

[AAS*17] ALSHAMMARI N., ALSHAMMARI T., SEDKY M., CHAMPION J., BAUER C.: Openshs: Open smart home simulator. *Sensors 17*, 5 (2017), 1003. 3

[ARCL13] ARIANI A., REDMOND S. J., CHANG D., LOVELL N. H.: Simulation of a smart home environment. In *International Conference on Instrumentation, Communications, Information Technology and Biomedical Engineering (ICICI-BME)* (2013), IEEE, pp. 27–32. 2

[BZDA17] BELLUCCI A., ZARRAONANDIA T., DÍAZ P., AEDO I.: End-user prototyping of cross-reality environments. In *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction* (2017), ACM, pp. 173–182. 2

[CD07] COOK D. J., DAS S. K.: How smart are our environments? an updated look at the state of the art. *Pervasive and mobile computing 3*, 2 (2007), 53–73. 1

[CGBA14] CLAUDE G., GOURANTON V., BERTHELOT R. B., ARNALDI B.: 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. 6

[Cou87] COUTAZ J.: Pac, an implementation model for dialog design. *Interact'87, Stuttgart* (1987), 431–436. 1, 3

[DF11] DUVAL T., FLEURY C.: Pac-c3d: A new software architectural model for designing 3d collaborative virtual environments. In *ICAT 2011* (2011), pp. 53–60. 3

[GMJ19] GKIOXARI G., MALIK J., JOHNSON J.: Mesh r-cnn. *arXiv preprint arXiv:1906.02739* (2019). 8

[GS12] GLAESSGEN E., STARGEL D.: The digital twin paradigm for future nasa and us air force vehicles. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA* (2012), p. 1818. 1

[Gül16] GÜLCH E.: Investigations on google tango development kit for personal indoor mapping. *studies 1* (2016), 3. 5

[HHM13] HEUN V., HOBIN J., MAES P.: Reality editor: programming smarter objects. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication* (2013), ACM, pp. 307–310. 2

[HNR68] HART P. E., NILSSON N. J., RAPHAEL B.: A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics 4*, 2 (1968), 100–107. 3

[JJH*15] JEONG Y., JOO H., HONG G., SHIN D., LEE S.: Aviot: Web-based interactive authoring and visualization of indoor internet of things. *IEEE Transactions on Consumer Electronics 61* (2015), 295–301. 1, 2

[Kop11] KOPETZ H.: Internet of things. In *Real-time systems*. Springer, 2011, pp. 307–323. 1

[KP*88] KRASNER G. E., POPE S. T., ET AL.: A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming 1*, 3 (1988), 26–49. 3

[LCK08] LERTLAKKHANAKUL J., CHOI J. W., KIM M. Y.: Building data model and simulation platform for spatial interaction management in smart home. *Automation in Construction 17*, 8 (2008), 948–957. 2

[LDA*15] LACOCHE J., DUVAL T., ARNALDI B., MAISEL É., ROYAN J.: Plasticity for 3d user interfaces: new models for devices and interaction techniques. In *Proceedings of the 7th ACM SIGCHI symposium on engineering interactive computing systems* (2015), ACM, pp. 28–33. 3

[LP09] LIFTON J., PARADISO J. A.: Dual reality: Merging the real and virtual. In *International Conference on Facets of Virtual Environments* (2009), Springer, pp. 12–28. 2, 8

[MQT14] MQTT V.: 3.1. 1. *Edited by Andrew Banks and Rahul Gupta 10* (2014). 2

[Mur89] MURATA T.: Petri nets: Properties, analysis and applications. *Proceedings of the IEEE 77*, 4 (1989), 541–580. 6

[NYT*06] NISHIKAWA H., YAMAMOTO S., TAMAI M., NISHIGAKI K., KITANI T., SHIBATA N., YASUMOTO K., ITO M.: Ubireal: Realistic smartspace simulator for systematic testing. In *International conference on ubiquitous computing* (2006), Springer, pp. 459–476. 2

[PMBK15] PARK B., MIN H., BANG G., KO I.: The user activity reasoning model in a virtual living space simulator. *International Journal of Software Engineering and Its Applications 9*, 6 (2015), 53–62. 3

[PPL18] PFEIFFER T., PFEIFFER-LESSMANN N.: Virtual prototyping of mixed reality interfaces with internet of things (iot) connectivity. *i-com 17*, 2 (2018), 179–186. 2, 8