

# Prototyping Context-aware Augmented Reality Applications for Smart Environments inside Virtual Reality

Jérémy Lacoche<sup>1</sup> and Éric Villain  
*Orange Innovation, Cesson Sévigné, France*

**Keywords:** Augmented Reality, Virtual Reality, Context-aware Applications.


**Abstract:** Prototyping context-aware augmented reality applications is a difficult task that often requires programming skills and is then not available for everyone. We aim to simplify this process thanks to a new virtual reality authoring tool for the creation of augmented reality applications for smart environments that can adapt to an evolving context of use. To do so, this tool introduces two main novelties regarding previous work. First, it proposes a prototyping step in the digital twin of the target environment where the author can create multiple versions of the content (visual aspect, modality, layout, area of visibility, etc.) and defines for each of them which context of use is targeted thanks to a dedicated diegetic user interface. Second, it includes the possibility to create new context variables with a visual programming approach that can leverage the smart environment sensors and actuators. The created application can then be deployed on various augmented reality devices and can support dynamic adaptations. We illustrate this tool with the creation of an application for smart buildings that can fit the needs of its various occupants. Thanks to a user study, we also present some usability feedback of this tool to assess its relevance and to provide guidelines for the future of this field of research.

## 1 INTRODUCTION

There is a growing interest for Augmented Reality (AR) Applications thanks to the democratization of mobile devices with built-in tracking capacities (ARKit, ARCore) and the technical improvements of AR Glasses (HoloLens, Magic Leap, etc.). At the same time, we are now surrounded by a lot of connected objects. They are present in the smart environments (smart-home, smart-buildings, smart-cities) we tend to all live in, and they form the Internet of Things (IoT): a pretty dense network of physical objects connected over the Internet that can sense the real world and act on it (Kopetz, 2011).

Thanks to these different technologies, ubiquitous computing, and pervasive augmented reality (Grubert et al., 2016) will be soon a reality for many people. While AR can provide users with continuous access to computing services and contextual information, the IoT can provide a lot of information about the current context of use thanks to its variety of sensors and actuators. Context-awareness in AR can benefit the end-users by always showing them the information that they need at the right time, and at the right place, and avoiding their distraction and cognitive overload

(Lindlbauer et al., 2019). It can also be used to only provide users with the services that correspond to their access rights. For instance, regarding a smart building augmented reality application, we would not propose the same services to the building managers, to the occupants, and to the visitors. In this paper, the context of use refers to information about the target platform, the user, and the environment (Calvary et al., 2004) (Grubert et al., 2016). Most of the time, AR applications do not provide any adaptation mechanism and are the same whatever the context-of-use encountered at runtime. Adaptations can concern different features such as visual style, modality, content presentation, service availability, and interaction modes (Krings et al., 2020). Creating context-aware AR applications has lot of challenges such as modeling the context of use, defining when and how each content element is displayed according to this context, and dealing with a constantly evolving context of use. Multiple solutions exist for the creation of such applications and to deal with these challenges such as AARCon (Krings et al., 2020), ACARS (Zhu et al., 2013) and the work of Lindlbauer et al. (Lindlbauer et al., 2019). However, they require the knowledge of a particular framework, the creation of adaptation rules in a dedicated programming language, or the

<sup>a</sup>  <https://orcid.org/0000-0003-3926-7768>

parametrization of a complex scoring system. Thus, they could involve a steep learning curve even for developers and could repel users without development skills.

That is why, in this paper, our main contribution is a Virtual Reality (VR) authoring tool for simplifying the creation of context-aware AR applications for smart environments. Target users of this tool are mainly AR developers, but contrary to previous work it could also target users without development skills such as designers, 3D artists, building managers, etc. Performing the prototyping step in VR requires the capture of a digital twin of the environment, but has the advantage to allow creators to remotely create and modify their applications and save them a lot of physical displacements (Soedji et al., 2020) (Prouzeau et al., 2020). Since the created applications are immersive, we also believe that designing in VR would allow authors to better anticipate results and benefit from better spatial cues than in a desktop editor. A digital twin is a virtual clone that can fully describe a real or potential product (Grieves and Vickers, 2017). In our case, a digital twin of a smart environment is composed of its geometry and its information about its sensors and actuators. Regarding previous work, our approach has two main advantages. First, our VR authoring tool aims to allow creators to prototype AR content and precisely define in which context of use it will be shown without using any programming language. When editing a set of content elements (3D meshes and widgets, texts, audio etc.), that we call here an augmentation layer, the user chooses the context it will correspond to by selecting a set of context variables values in a diegetic user interface. Target adaptations can then concern most of the categories cited in (Krings et al., 2020) such as modality, content visual aspect, and services. Second, it allows all types of users to take advantage of the connected objects present inside the real environment to extend the range of context of use variables that can be taken into account thanks to dedicated 3D interaction metaphors. With our tool the author can easily define the range of action of each variable in the 3D space. We propose to analyze the possible benefits of this tool and to determine how it could be improved thanks to the development of a smart building AR application and thanks to a user study based on a panel of expert users.

Our paper is structured as follows, Section 2 presents some related work. Then, Section 3 introduces how context-adaptations can be defined in our tool and how the application can then be deployed on AR devices. Section 4 demonstrates how our tool can benefit the creation of an AR application for smart

building occupants. Then, Section 5 presents a user study that allows us to provide first feedback for our solution and identify some research perspectives. Finally, we conclude and give some opportunities for future work.

## 2 RELATED WORK

To simplify the creation of AR applications, WYSIWYG (What You See Is What You Get) tools aim to allow a creator to prototype applications in the target environment. Lee et al. (Lee et al., 2004) propose such a solution where a user can edit AR content, including its visual aspects, and behaviors with an AR device. Thanks to 3D interactions, content creators can rapidly prototype an interactive 3D content in situ. Similarly, MARVisT is a tablet-based AR authoring tool (Chen et al., 2019) that allows non-expert users to create glyph-based visualization AR applications. Recent commercialized solutions such as Microsoft Guides (Microsoft, 2021) and Minsar Studio (Minsar, 2021) push this concept to the consumer market. Other approaches such as Corsican Twin (Prouzeau et al., 2020) and CAVE-AR (Cavallo and Forbes, 2019) propose to edit AR content in a digital twin of the real environment in VR. Similar approaches are proposed in (Soedji et al., 2020) and (Pfeiffer and Pfeiffer-Leßmann, 2018) for the creation of AR applications for supervising and controlling smart environments. Such VR tools benefit from the accuracy of VR 6DoF controllers, save users from physical displacements in large environments and provide users with a preview of their content by simulating AR devices inside VR. They also allow users to remotely edit their content in an iterative way without having to have constant physical access to the real environment. All these authoring tools can benefit users without development skills. However, for now, they lack adaptation features: the content will be the same whatever the context of use encountered at runtime.

Grubert et al. (Grubert et al., 2016) introduce a taxonomy of context-aware augmented reality where they identify the different context sources and adaptation targets. They also introduce the concept of Pervasive Augmented Reality as a continuous, omnipresent, and universal augmented reality experience. Such an experience would require filtering and adaptation mechanisms if we don't want users to be overwhelmed by non-relevant 3D content in their daily life. This requirement can for example be illustrated by the anticipation short film of Keiichi Matsuda: Hyper-Reality (Matsuda, 2016). Julier et al. (Julier et al., 2000) proposed one of the first solu-

tions for content filtering for augmented reality applications. It relies on a scoring system that computes the importance of each augmentation according to the current task of the user. It then adjusts the opacity of each element from completely opaque to completely invisible. Similarly, Lindlbauer et al. (Lindlbauer et al., 2019) propose an optimization process to dynamically adapt the AR widgets shown to the user. Adaptations concern how much information is shown to the user, the level of detail of each widget, and where they are placed. These adaptations are chosen by an optimization process according to the user's current cognitive load and knowledge about their task and environment. To continue, CAMAR (Context-aware Mobile Augmented Reality) (Oh et al., 2009), is based on the UCAM framework (Oh and Woo, 2007) and allows the creation of Context-Aware mobile augmented reality experiences. Based on sensors and services that capture the user context, the content of the application is chosen and customized. Users can then share their personalized experiences with other selected users. Then, ACARS (Zhu et al., 2013) (Authorable context-aware AR System) is a solution for the creation of context-aware AR maintenance applications. AR developers can create context-relevant information via a desktop 2D user interface and maintenance technicians can also edit the created content on-site with their AR device. The context of use is represented as an ontology and SWRL rules (Horrocks et al., 2004) are used to check this context and trigger information presentation adaptations. These graphical authoring tools could simplify the work of content creators, but handling SWRL could require a steep learning curve for novice users. More recently, Krings et al. proposed the AARCon framework (Krings et al., 2020) for "Android-based framework for Augmented Reality with Context-Awareness". The solution relies on three main components: Context Monitoring, Adaptation, and Decision Making. Content sensors and adaptation mechanisms can be implemented through a dedicated API. Adaptations are triggered by the decision-making process when its conditions are met and can for instance impact the interaction techniques and the content. These conditions correspond to the values of the context sensors that can contain information about the platform, the user, and the environment. For now, this solution does not come with an authoring tool and is then dedicated to developers.

Regarding these solutions, our goal is to provide developers and designers with a solution that proposes both the advantages of WYSIWYG tools and context-aware frameworks. As we target smart environments, we aim to allow end-users to leverage the

IoT to infer the context information they need. We also want to investigate if configuring such adaptations in a visual tool would simplify the work of developers and designers when creating AR content.

### 3 CONTEXT-AWARE VR AUTHORIZING TOOL AND AR DEPLOYMENT

We propose a VR authoring tool that allows users to prototype augmentation layers for each given context of use they want to consider. The content is edited in the digital twin of the real environment, a virtual replica that is composed of its geometry, the positions, and the types of its connected objects. The created content can then be seamlessly deployed to AR devices.

The creation workflow of our solution includes this VR tool and the AR deployment step. It can be broken down into four different steps.

1. First, a digital twin of the target smart environment is initialized. In this paper we do not address this issue and we rely on existing solutions. In our case study described in Section 4, the digital twin was built from connected objects' positions and types collected on-site thanks to an AR tool inspired from the work described in (Soedji et al., 2020) and from the Building Information Model (BIM) file of the building. Regarding the geometry, other approaches could be used such as photogrammetry.
2. Second, developers pre-define and pre-develop context variables and how their value are computed at runtime as well as content that can be used in the final application. The context of use is then defined by the values of all these context variables arranged in a tree as detailed in Section 3.1. It can contain information about the user, the environment and the hardware platform. The content can correspond to 3D meshes, audio files or widgets for controlling smart objects as detailed in Sections 3.2 and 3.4.
3. Third, a content creator (developer, designer, building manager, etc.) prototypes augmentation layers in VR in the digital twin and defines when they will be shown according to the context of use with a dedicated diegetic user interface as defined in Section 3.2. Here, an augmentation layer is a set of interactive content elements precisely placed in the target environment. New context

variables can be created with dedicated 3D interaction metaphors by exploiting the connected sensors and actuators of the environment as detailed in Section 3.1. With this feature, the context definition is not static and is not only available for developers.

4. Last, the created interactive content is deployed in the real world with AR devices. The displayed content is dynamic as each layer is hidden or shown according to the context of use encountered at runtime.

### 3.1 Context of Use Model

To define the context of use, we rely on a tree-based representation as shown in Figure 1. It breaks down the context into categories and sub-categories. Each of them corresponds to a variable that can take multiple values. An example of a category given in Figure 1 is the role of the end-users. An example of a sub-category is the task that can be associated with the security guard role. In the same way, the device type category could also be divided into subcategories in order to precisely define which device model is targeted. A given context of use is then characterized by the set of value nodes that are activated or not. This can be compared to the set of conditions of a rule in a rule-based approach. For instance, a possible encountered context of use could be: ((**Device:** HMD), (**Role:** Visitor), (**Evacuation:** False)). We chose this representation over an ontology-based approach such as in (Horrocks et al., 2004) for several reasons. Indeed, the first advantage of this tree representation is that it is easy and quick to define and extend. Then, the second advantage is that it can be mapped onto a diegetic user interface as shown in Figure 2a to be

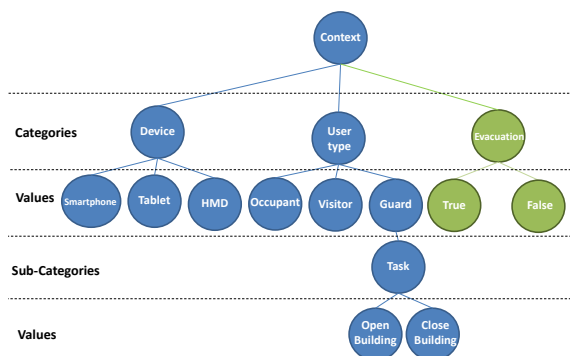


Figure 1: The context-tree used in our use-case described in Section 4. Blue nodes correspond to pre-defined context variables (implemented through a dedicated API) while green nodes correspond to context variables defined in the editor that are linked to connected objects.

easily understood by all kinds of users. This interface corresponds to a submenu of the diegetic user interface placed on the author’s non-dominant hand. When creating a given augmentation layer as detailed in Section 3.2, the user defines through this interface the targeted context of use by selecting its set of conditions. These advantages are consistent with our goal to simplify and accelerate the creation of context-aware AR applications for content creators. The disadvantage of not using ontologies for describing the context of use is to not being able to exploit their advantages to describe complex situations and relations that could be helpful for the most advanced scenarios.

We propose two methods to define this context tree. The first one is dedicated to developers and requires the use of a programming language and, the second one is integrated into our VR authoring tool and is available for all kinds of users.

The first one is a static approach where a developer can define with a dedicated API the structure of the tree and provide the different sensor’s components to detect the value of each variable at runtime. The value of each context-variable can then evolve at runtime according to the code implemented by the developer. This is similar to the approach used in the AARCon framework(Krings et al., 2020). In Figure 1, the blue nodes of the tree were pre-defined. This first approach provides developers with a lot of freedom for creating context variables and for defining how they will be set at runtime. It could allow them to develop simple mechanisms, such as connecting some variables values to an existing database, as well as more complex ones such as using artificial intelligence algorithms to deduce other variables values. As an example, for the ”Device” variable shown in Figure 1, we used this API to implement a program that checks at runtime the device physical model that runs the application and then determines accordingly its value between ”Smartphone”, ”Tablet” and ”HMD”.

Secondly, we propose a new dynamic approach where the author can add new boolean context variables in our VR authoring tool and define their range of action in the 3D space without recompiling any piece of code. Their values will depend on the state of different connected objects of the environment. To do so, the author can visually link connected objects data to the value of a newly created context variable. This is performed with visual programming metaphors that can be compared to the ones proposed in (Lacoste et al., 2019) and (Wang et al., 2020). The context variable is added from the same menu and is represented inside the virtual environment as a sphere. Virtual links can be created between this object and some connected objects. Virtual logical gates and routes

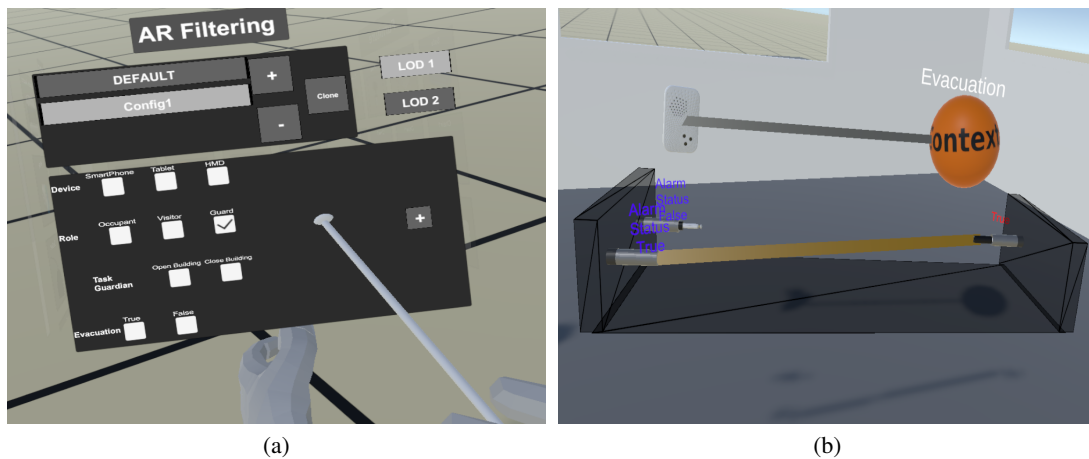


Figure 2: Context variables in our VR editor. (a) The tree detailed in Figure 1 is mapped onto a diegetic interface in the VR editor. The user can select which context of use to target for a given augmentation layer. (b) Mapping of an evacuation context value (green nodes in Figure 1) to the activation state of a connected alarm.

can then be edited to map the context value (True or False) to the data of the linked objects. Boolean as well as quantitative data from the connected objects can be used. In Figure 1 the green nodes correspond to a newly added variable: an evacuation scenario. As an example, in Figure 2 we show the mapping between the value of this context variable and the activation status of an alarm. At runtime this context variable will be set to true as soon as this alarm is activated. By default, each context variable applies everywhere in the environment, but it is also possible to define their range of action in the 3D space. Indeed, the author can define an area of application composed of four corners, placed on the floor, for each created variable. For instance, we could be interested in associating each building space with a noise context variable by checking microphones' decibel levels. Depending on their value, we could prefer visual to audio augmentations in the corresponding area.

With this tree-based representation and these two ways to extend it, our VR editor allows users to take into account different kinds of content information about the end-user, the environment and the target platform. Our visual programming approach to infer context information from the IoT makes the definition of the context of use available for non-developers. It also simplifies the association between a context variable and a given range of action in the environment compared to previous work.

### 3.2 Defining Augmentation Layers for Each Defined Context of Use

Once this context of use has been modeled, the author can edit several augmentation layers and define

for each of them the target context of use by selecting a set of context variables (conditions) from the user interface. While WYSIWYG approaches for the creation of AR content can be found in previous work, this is the first one that can take into account the context of use during a prototyping step performed at scale one.

Each layer is a group of augmentations that are added to the scene from another submenu that contains 3D widgets, UI elements, text components, audio files, and 3D meshes. Most of these augmentations can be placed anywhere in the scene. As our solution is built for the creation of smart environments applications, some augmentations can be associated with connected objects. These specific augmentations are dedicated to their supervision and control. We propose a drag and drop interaction technique to associate these augmentations with their compatible objects. Compatible objects are highlighted with a colored halo when the user manipulates an augmen-



Figure 3: An example of augmentation for controlling a connected light. This augmentation is part of an augmentation layer. The green polygon corresponds to the area where this augmentation will be visible from the user.

tation. An example of augmentation is given in Figure 3, it corresponds to a 3D widget for controlling a connected light. Regarding interactive augmentations and more specifically the ones that can be associated with connected objects, they need to be pre-defined by a developer as detailed in Section 3.4.

For each augmentation, multiple parameters can be edited. First, as in similar tools cited in Section 2, each augmentation can be precisely positioned, scaled, and rotated. A billboard mode can also be enabled or disabled. For text components, the displayed text and color can be edited. For audio files, the sound volume can be adjusted and the user can define if they are automatically played and in loop mode or not. Then, each augmentation can also be associated with a visibility area as shown in Figure 3 in order to take into account the user location. As for the context variables, this area is composed of four corners that can be moved on the floor. At runtime, the augmentation will be displayed as long as the user is located in this area. For audio files in auto-play mode, the sound is played when the user enters this area. Each augmentation can also be associated with a second level of detail, it corresponds to the augmentation that will be seen when the user is located outside the visibility area of the corresponding augmentation. As an example, for the augmentation shown in Figure 3, when the user is close to the light we could display its full control interface, and the second level of detail could just display a light icon to indicate its presence when the user is far from it.

As such smart environments can be large and composed of a lot of objects. We propose multiple mechanisms to improve the author's workflow. We propose a copy-paste feature to duplicate a connected object augmentation and its parameters for other selected compatible objects. As well, a full augmentation layer can be cloned and associated with a new set of context variables values. As an example, a user could first edit a default configuration and then clone and adapt it for a new target context of use. To continue, the user can create and name pre-defined areas that can then be applied to augmentations as visibility area and to context variables as action range. It avoids users to create multiple times the same area for multiple objects. Some areas could also be automatically initialized at launch time, for example by extracting rooms from a BIM. Soon, we also plan to add a World-In-Miniature feature as proposed in Corsican Twin (Prouzeau et al., 2020).

Each augmentation layer is also associated with an adaptation policy parameter that defines how it is applied at runtime according to the encountered context of use. It provides content creators with multiple pos-

sibilities to define how and when their augmentations layers will be shown and hidden at runtime. Indeed, this parameter impacts the runtime adaptation process and can take three different values and can be set in the user interface:

- **Concurrent:** the augmentation is activated when all its conditions are met and the number of its conditions is the highest compared to the other "concurrent" layers.
- **Stacked:** the augmentation layer is automatically activated when its conditions are met. It can be displayed at the same time as other concurrent augmentation layers. For instance, as illustrated in Section 4, whenever the role of the user is "guard" we activate 3D widgets to display the data of air quality sensors.
- **Priority:** the augmentation layer is automatically activated when its conditions are met. All other concurrent and stacked layers are automatically disabled. Other priority configurations can still be shown. An example of such a layer is illustrated in Section 4 for an evacuation scenario.

With this approach, the author can address most of the adaptation targets cited by Grubert et al. (Grubert et al., 2016) (visual aspects, style, modalities, services) and can have control over the adaptation process to precisely define the behavior of its application at runtime. However, for now, adaptations do not concern the interaction techniques even if some of the chosen pre-developed 3D widgets can involve different input modalities such as gesture and voice.

### 3.3 End User AR Application

Regarding AR deployment, we use a similar approach as the one described in (Soedji et al., 2020). The output of the VR editor is an XML file that describes the different AR augmentation layers and that is interpreted by a dedicated AR player at runtime. Similarly, we also propose a VR player that can simulate AR devices in order to assess an application before its deployment in the real world.

We enhanced this solution by including our adaptation mechanisms. The architecture of this process is inspired by the AARCon framework (Krings et al., 2020). It is composed of the same three main components. First, a *context monitoring* module constantly collects the data from the pre-developed context sensors. It also solves the values of the context parameters visually programmed in the VR editor according to data coming from the connected objects. Secondly, a *decision-making module* constantly checks

the context values and determines which augmentation layer to enable and disable according to their adaptation policy parameter (stackable, concurrent, priority) and if their conditions are met or not. Regarding concurrent and stackable layers, as the context of use can evolve at runtime, conditions need to be met for a given amount of time in order to avoid constantly changing augmentations. Indeed, flickering connected objects data could imply constant modifications of some context variables. This parameter can be configured by developers and is set by default to 10 seconds. We do not wait for this delay for priority layers. Last, the *adaptation controller* module instantiates, shows, and hides the augmentations when asked by the decision-making module. It also adapts the visibility and the LOD of each augmentation element according to the user position as defined within the VR editor. Indeed, as detailed in Section 3.2, for each augmentation associated with a visibility area, two levels of detail can be defined in our VR editor, one visible when the user is inside this area, one when the user is outside. Then, the *adaptation controller* just checks the position of the user to enable or disable each level of detail. The result is an AR application that can dynamically react to the context of use modifications encountered at runtime.

### 3.4 Implementation

Our solution is currently developed with Unity 2019.4 LTS<sup>1</sup>. The VR editor is developed with the OpenVR plugin and has been tested with an Oculus Quest 2 in link mode. Interactions with the UI elements and with the objects are performed with a 3D ray-based selection and manipulation technique. The editor supports the import of pre-developed augmentations exported in Unity AssetBundles, the creation of 3D texts, and the dynamic import of GLTF 3D models and .WAV audio files. Compatibilities between augmentations and connected objects need to be specified in an XML configuration file. All these files are stored in a dedicated Dropbox repository and accessed through Dropbox HTTP API<sup>2</sup>. This allows developers to easily extend the visual elements that can be instantiated in the editor and also simplify the deployment on AR devices.

For AR tracking, our solution relies on the AR-Foundation plugin<sup>3</sup> in Unity which allows us to deploy to Android (ARCore) and iOS (ARKit) compati-

<sup>1</sup><https://unity.com>

<sup>2</sup><https://www.dropbox.com/developers/documentation/http/overview>

<sup>3</sup><https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/>

ble devices. We can also deploy to the Magic Leap One and to the HoloLens thanks to their dedicated SDK available for Unity. For pose estimation in the real environment, our solution uses a natural image marker for initialization and then relies on the internal SLAM of the device. Our objective for future work is to avoid the use of this marker and to use the digital twin of the environment for dynamic relocalization at runtime. Communication with real connected objects is ensured through different network protocols such as Wi-Fi and Z-Wave. These players can be extended with a dedicated API to configure how the context variables defined in the context tree are captured at runtime.

## 4 CASE-STUDY: AR IN SMART-BUILDING

As a case study, we developed a context-aware AR application for a smart office building. This application aims to provide users with adapted services according to their role, device and situation. It is based on the context-tree detailed in Figure 1. Here, we mainly focus on two roles: office worker and security guard and on a particular situation: emergency evacuation. This emergency evacuation scenario allows us to demonstrate that our solution can support dynamic adaptations at runtime according to the connected objects data. We performed a preliminary interview with security guards in order to determine their constraints and their expectations regarding an AR assistance tool. The office workers features are for now proposals that still need to be assessed by potential users. With our solution, a first version of the application can be designed by a 3D developer in our VR editor. Then it can be enhanced and maintained over its lifetime by the building manager in the same tool which would have been more difficult with previous work. Regarding the context variables, role and tasks are collected through a user authentication process. The information about the user's device are given by the AR player. A video of this case study and how it can be prototyped with our tool can be viewed at the following link: [https://youtu.be/zzeLp6\\_Q6G0](https://youtu.be/zzeLp6_Q6G0).

### Office Worker

In order to assist office work in their daily tasks, our applications provides two different AR services. First, as seen in Figure 4a, a first set of augmentations is dedicated to meeting rooms. The name of each room is displayed just in front of the door and is only visible in the corridor. When the user is getting

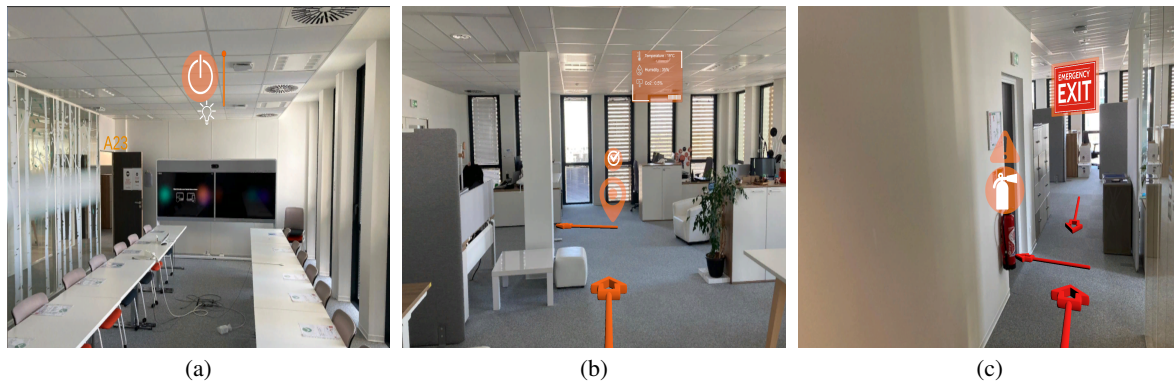


Figure 4: Example of augmentations adaptation in our smart office building application (a) Office worker can control the connected lights in each meeting room with a dedicated 3D user interface (b) For security guards, the patrol is displayed with arrows and check points. 3D user interfaces are displayed to check the data of air quality sensors. (c) During an emergency exit, we display to all users the path to each exit and we indicate where are the fire extinguishers.

close to the room, we display the information about the ongoing meetings. To do so, each meeting room is considered as a connected object in our digital twin, and can be associated to a dedicated augmentation as detailed in Section 3.2. Then, at runtime, these augmentations have access to each room booking information. A light control augmentation is also visible in each meeting room to set the On/Off status and the dimming as shown in Figure 4a. Secondly, regarding the current COVID-19 situation, we also display the building circulation flows with 3D arrows and we display a 3D icon on top of each antibacterial gel. Each of these icons is visible in a delimited area of the building. With our authoring tool, we can also include differences according to the user's device. For smartphones and tablets users that may not always look at their device (but could still be tracked), we also include an audio augmentation when they get close to one of these sanitizers. This sound advises them to wash their hands.

### Security Guard

The security guard must perform patrols at fixed times. Each patrol has some specific objectives. Some of the guards can have little knowledge about the building and they need to be assisted in their tasks. In our case, we consider two types of patrols: an "opening patrol" at the beginning of the day and a "closing patrol" at the end of the day. They correspond to the tasks of the security guard shown in Figure 1. During each of them, the guard has to follow a predefined path connecting successive checkpoints. To do so, as shown in Figure 4b, the augmentation layer displays the path to follow thanks to 3D arrows as well as virtual checkpoints. Each checkpoint corresponds to a 3D widget. It allows the guard to manually indicate that he proceeded the corresponding security

checks in this area (in that case the checkpoint disappears). Along the path, 3D widgets are also shown to allow the security guard to make some verification and report issues about fire extinguishers, defibrillators and hand sanitizers. Some augmentations also allow to monitor the data of the air quality sensors present in each room as shown on top of Figure 4b. Thanks to our authoring tool, we also include differences according to the type of patrol. For the opening patrol, we display a 3D augmentation on each meeting room door to indicate that it needs to be opened. As well, for the closing patrol, similar augmentations are also displayed to indicate that each meeting room door needs to be closed. In addition, the same icons are also added to check all access of the building to ensure that it is correctly secured. For the same closing patrol, augmentations are also placed on each elevator in order to check if nobody is stuck.

### Emergency Evacuation

We also include a dedicated augmentation layer for the emergency evacuation of the building. As shown in Figure 4, this layer displays the shortest path to each emergency exit with 3D arrows, and it also displays the locations of the fire extinguishers. This layer requires the creation of a new context variable in the authoring tool as detailed in Section 3.1. In our case, the author can link this context variable to any connected alarm that is present in the building as shown in Figure 2. This augmentation layer is defined as "Priority" as detailed in Section 3.2. It means that, when the alert occurs, the application immediately reacts and the associated layer is activated while all the other ones are disabled. For this situation, we also plan to include differences according to the role of the user. Indeed, the security guard could have additional information to organize the evacuation and perform



additional security checks.

In the future, we also plan to improve this application. We plan to add dedicated features for visitors such as guiding information. We also plan to add the role of building manager. Such a role could benefit from augmentations to monitor and control all the connected objects of the building. Other connected objects data could also be exploited to perform adaptations. For example, as detailed in Section 3.1, context variables can be associated to delimited areas, we could imagine creating an intrusion context variable for each area of the building. Such variables would be linked to the presence sensors of this area. When such a situation is encountered we could display to the security guard the area to check with 3D elements. A similar example could concern meeting rooms. As such rooms are considered as connected object in our digital twin of the building, the author could check the availability of a meeting room in a given area by linking their availability to a new context variable. Then, he could indicate with dedicated augmentations where the end-user can find an available room.

## 5 USER-STUDY: COLLECTING FEEDBACK FROM IMMERSIVE CONTENT CREATORS

We did not have access to any comparable tool so we did not perform a comparative user study. We then performed a qualitative study of our solution to provide feedback about its usability and get areas of improvement and perspectives for this field of research. In this study, we asked users to create multiple augmentation layers for different contexts of use in our VR tool in a delimited area (450m<sup>2</sup>) of a smart building. These layers are based on our case study described in Section 4. For this user study we chose a panel of "expert" users. Here we considered as experts, users with development skills and knowledge about VR, AR and connected objects. Indeed, first, they are the main target for the use of our tool even if we think that it could also be exploited by users without development skills. The main goal of our solution is to save such potential users time during the creation of context-aware AR applications. Secondly, we also wanted first usability feedback from experienced users before testing the tool with more novice users. We consider their expert assessment valuable to improve the usability of our tool and identify the features that could be difficult to use by novice users. They are also interesting to raise new research per-

spectives. The experimentation was performed with the Oculus Quest 2 connected to a laptop PC (RTX 2080, Intel Core i9-9900K, 32Go RAM). Our application was running smoothly at 72 frames per second.

**Participants.** Our experimentation panel consisted of 10 subjects (age: M=41.2, SD=9.6), 9 males and 1 female. All of them had development skills and at least basic knowledge about VR, AR, and connected objects. 4 of them were working in the field of VR/AR content creation. As detailed, this panel of expert users was chosen on purpose.

**Procedure.** In the first step of the experimentation, users could get familiar with the different concepts and interactions by reading documentation and by testing the tool in VR for 10 to 15 minutes.

After the training step, users had to read instructions about the four augmentations layers they would have to create in VR. These four layers corresponded to the four scenarios described in Section 4: a layer for the workers, two for the guardians (opening and closing patrols), and one for the building evacuation. Users had to exploit most of the features of our VR editor that are described in this paper. Each layer consisted of 10 to 25 augmentation elements. Some of them had to be associated with a visibility area. The number of added elements also depended on the user. As an example, some users added more arrows than others for the evacuation layer. For each layer, users had to select its associated context variables values and its adaptation policy. For the evacuation layer, users had to create a new variable and link its value to a connected alarm.

After reading these instructions, each user could then proceed to the testing phase in VR. During this step, users could ask questions whenever they wanted in case they forgot the different instructions. They could also take a break if needed.

**Collected Data.** Completion times were recorded, but are not deeply analyzed here. Indeed, 3 users did not know the real building before the experimentation. Moreover, as said, some users were more careful than others. All of them did not produce the same result. Users spent between 20 and 35 minutes inside the tool during the testing phase. At the end of the experimentation, users were asked to rate 11 affirmations (shown in Table 1) on a Likert scale from 1 (strongly disagree) to 5 (strongly agree). For each affirmation, users could write comments to justify their answer.

Table 1: Mean and standard deviation for each affirmation given by the users at the end of the experimentation.

ID	Affirmation	Result
A1	I appreciated the experience	M=4.6 SD=0.52
A2	The tool seems adapted to the creation of smart environments applications	M=4.4 SD=0.52
A3	The tool can save time for content creators	M=4.6 SD=0.52
A4	I understood the different concepts and succeeded in using them	M=4.4 SD=0.70
A5	The tool allowed you to precisely define the augmentations according to the context	M=4.5 SD=0.70
A6	It is easy to anticipate how the application will behave according to the context	M=4.1 SD=0.74
A7	It is easy to extend the context with the connected objects	M=4.3 SD=0.48
A8	The tool could be used by users without development skills	M=4.3 SD=1.06
A9	You would have preferred performing this configuration in augmented reality	M=2.5 SD=1.08
A10	You would have preferred performing this configuration with a 3D editor (Unity, Unreal, Blender, etc.)	M=1.9 SD=0.99
A11	You would have preferred performing this configuration with a system more close to a programming language (dedicated framework, high-level rules, scoring system, etc.)	M=1.7 SD=0.82

**Results and Discussion.** The mean and the standard deviation for each affirmation are provided in table 1. Globally the tool was appreciated (A1) and participants seemed to confirm its value for the creation of context-aware AR applications for smart environments (A2, A3). Regarding A1, a lot of usability improvements were suggested by participants such as "wrong feedback colors", "text size too small", "complicated manipulations with the 3D-ray" and "occlusion issues between UI elements and the digital twin walls". These comments do not question our methodology but highlight that improving the usability of the tool is essential if we want to make it available for all kinds of users. For A2, one user suggested that "the tool could be used in the context of a museum to create adapted tours depending on the user profile". Most users were confident about their understanding of the different concepts introduced in the tool (A4). This result suggests that our tool has a rapid learning curve. One user still commented that they were "a lot of concepts to assimilate", but he "got used to it". Another one said that the notion of layer was "hard to understand at the beginning". On the contrary, One user also said that the "learning time was short". Moreover, some users still asked questions about the adaptation concepts during the testing phase. Most of them concerned the understanding of the adaptation policy (concurrent, stacked, or priority). We think that the training step was not long enough to completely assimilate all capabilities of the tool. It then raises the necessity to evaluate the time needed to acquire complete autonomy with the tool.

We believe that this time could be different between users depending on their level of expertise in development, AR, VR, and connected objects.

Users felt comfortable exploiting the context of use (A5) and extending it with the proposed interaction metaphors (A7). One user still said that "it could get more complex with a lot of context variables". Some users also commented that sometimes they forgot which layer they were currently editing. Indeed, the active layer could only be determined from the dedicated menu. Feedback about the active layer could be always displayed in the user's field of view to overcome this issue. Regarding A7, one user commented that it was "difficult to identify the connected objects and to differentiate them from the other objects of the scene". We could imagine giving the possibility to the author to highlight the objects when needed to quickly identify where they are. As well, this comment raises the need to improve our current implementation but does not concern our methodology. Another one said that "it requires to have some knowledge about the data that can provide each type of connected object". Indeed, basic knowledge about the sensors of the building and their capabilities is a prerequisite for the use of this feature of the tool. To continue, users also agreed that it was easy to anticipate how the application would behave at runtime according to the context of use (A6). However, this rating can be moderated by some observations and comments. As an example, after defining a visibility area for a given augmentation a lot of users expected that it would immediately hide after moving outside

of this area. Some users commented on the need for a *"testing mode"* and one commented that he would have liked to *"immediately assess the behavior of the application in AR"*. As explained in Section 3.3, a testing mode in VR is implemented in our solution, but was not proposed during the experimentation as we wanted to focus on the design step. These observations and comments confirm the interest in such a tool when editing context-aware AR applications in VR. Authors need to be able to immediately test their application.

The panel tended to agree that the tool could be exploited by users without development skills (A8). This is particularly important for us as we aim to make it available for such users. Some comments still raised some warnings such as *"the concept of variable could be difficult to understand"* as well as *"the logical gates and routes to connect context variables and connected objects could be considered complicated"*. Moreover, this result can also be moderated as our panel was only composed of users with development skills. This result is encouraging, but needs to be confirmed with novice users. Indeed, designers, 3D artists, building managers are also possible target users.

A9 was the affirmation from which we obtained the most diverse answers. The result tends to say that globally users were between slight disagreement and neutrality. By analyzing the comments, we can find arguments in favor of both possibilities: editing in VR in the digital twin or editing in AR in situ. One user raised a warning about *"the complexity to create the digital twin of the building"*. Such an issue could be addressed with a fully automatic digital twin capture tool. One user also told us that it would depend on the size of the building: *"for a small area it would be easier in AR, but it would be faster in VR for a large building"*. Two users also commented that *"doing it in VR avoid disturbing the building occupants"*. Both approaches have advantages and drawbacks, but regarding these comments, they could also be considered complementary. Regarding our solution, this result suggest that our VR authoring tool might not satisfy every user. The same authoring features could then be implemented on AR glasses (HoloLens, Magic Leap, etc.) to let them choose the platform they prefer.

To finish, participants tended to disagree with affirmations A10 and A11. They would not have preferred to perform the same task in a 3D editor or with a dedicated framework. Of course, these affirmations could have differed in a comparative study where participants could have tried another approach. However, some comments can be cited to illustrate the partici-

pants' answers. Regarding A10, one user commented that he *"would have been more precise with a 3D editor"*. Two users also told us that they think that such an editor would require an increased learning curve. Regarding A11, two users said that using a programming language could help to deal with the most complex situations, for instance, when *"there are a lot of context variables"*. One user also mentioned that our VR editor could be *"complementary to a 3D editor and a dedicated framework"*.

## 6 CONCLUSION AND FUTURE WORK

To conclude, we propose a VR editor for the creation of context-aware AR applications for smart environments. This editor aims to be adapted to developers, but also to users without development skills such as designers, 3D artists and building managers. We have shown its relevance for the development of a smart-building application that can be adapted to the needs of various users and that can dynamically react to a situation encountered at runtime. We collected feedback about its usability with a panel of expert users. Our tool was globally well appreciated and users succeeded in using it to create AR content that can adapt to multiple situations. The results confirm the relevance to incorporate the context of use management when creating AR content in the digital twin of a smart environment. Moreover, they also highlight that a single tool in VR could not satisfy every user. Therefore, as a first future work we could imagine developing a full software suite with a dedicated framework including our VR design tool but also a 3D editor with a 2D user interface and an AR design tool. Content creators would then choose which one to use depending on the operation they have to perform, their preferences, their needs, and their skills. As detailed in Section 2, such an idea was partly implemented in ACARS (Zhu et al., 2013).

As future work, we also plan to consider adaptation to the target environment topology. Indeed, for now, the AR application will fit only one real environment. Regarding our case study, we would need a solution for retargeting our prototyped application for another building. Then, we plan to perform a second user study with users without development skills in order to determine if our tool can fit their needs and if some improvements should be introduced. To finish, it could be interesting to evaluate our solution when the number of context variables is way larger than the dozen parameters that we consider in our case study. We think that the combinatorial complexity of

manually editing the adaptations in VR editor would raise with the number of possible parameters. A large number of context variables could generate conflicts between layers and some particular context of uses could be unintentionally omitted by the author. Some additional automatic adaptation processes and additional feedback in our tool could be needed in such cases.

## REFERENCES

- Calvary, G., Coutaz, J., Dâassi, O., Balme, L., and De-meure, A. (2004). Towards a new generation of widgets for supporting software plasticity: the "comet". In *IFIP International Conference on Engineering for Human-Computer Interaction*, pages 306–324. Springer.
- Cavallo, M. and Forbes, A. G. (2019). Cave-ar: A vr authoring system to interactively design, simulate, and debug multi-user ar experiences. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 872–873. IEEE.
- Chen, Z., Su, Y., Wang, Y., Wang, Q., Qu, H., and Wu, Y. (2019). Marvist: Authoring glyph-based visualization in mobile augmented reality. *IEEE transactions on visualization and computer graphics*, 26(8):2645–2658.
- Grieves, M. and Vickers, J. (2017). Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems*, pages 85–113. Springer.
- Grubert, J., Langlotz, T., Zollmann, S., and Regenbrecht, H. (2016). Towards pervasive augmented reality.
- Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosz, B., Dean, M., et al. (2004). Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79):1–31.
- Julier, S., Lanzagorta, M., Baillet, Y., Rosenblum, L., Feiner, S., Hollerer, T., and Sestito, S. (2000). Information filtering for mobile augmented reality. In *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, pages 3–11. IEEE.
- Kopetz, H. (2011). Internet of things. In *Real-time systems*, pages 307–323. Springer.
- Krings, S., Yigitbas, E., Jovanovikj, I., Sauer, S., and Engels, G. (2020). Development framework for context-aware augmented reality applications. In *Companion Proceedings of the 12th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 1–6.
- Lacoste, J., Le Chénéchal, M., Villain, E., and Foulonneau, A. (2019). 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*.
- Lee, G. A., Nelles, C., Billingham, M., and Kim, G. J. (2004). Immersive authoring of tangible augmented reality applications. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 172–181. IEEE.
- Lindlbauer, D., Feit, A. M., and Hilliges, O. (2019). Context-aware online adaptation of mixed reality interfaces. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 147–160.
- Matsuda, K. (2016). Hyper-reality. <http://hyper-reality.co/>. Accessed: 2021-03-31.
- Microsoft (2021). Microsoft guides. <https://docs.microsoft.com/dynamics365/mixed-reality/guides/>. Accessed: 2021-03-31.
- Minsar (2021). Minsar studio. <https://www.minsar.app/>. Accessed: 2021-03-31.
- Oh, S., Woo, W., et al. (2009). Camar: Context-aware mobile augmented reality in smart space. In *Proceedings of International Workshop on Ubiquitous Virtual Reality*, pages 15–18. Citeseer.
- Oh, Y. and Woo, W. (2007). How to build a context-aware architecture for ubiquitous vr. In *International Symposium on Ubiquitous VR*, page 1.
- Pfeiffer, T. and Pfeiffer-Leßmann, N. (2018). Virtual prototyping of mixed reality interfaces with internet of things (iot) connectivity. *i-com*, 17(2):179–186.
- Prouzeau, A., Wang, Y., Ens, B., Willett, W., and Dwyer, T. (2020). Corsican twin: Authoring in situ augmented reality visualisations in virtual reality. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pages 1–9.
- Soedji, B., Lacoche, J., and Villain, E. (2020). Creating ar applications for the iot: a new pipeline. In *26th ACM Symposium on Virtual Reality Software and Technology*, pages 1–2.
- Wang, T., Qian, X., He, F., Hu, X., Huo, K., Cao, Y., and Ramani, K. (2020). Capturar: An augmented reality tool for authoring human-involved context-aware applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 328–341.
- Zhu, J., Ong, S. K., and Nee, A. Y. (2013). An authorable context-aware augmented reality system to assist the maintenance technicians. *The International Journal of Advanced Manufacturing Technology*, 66(9-12):1699–1714.