# Redistribution et Plasticité pour les Interfaces Utilisateurs 3D : un Modèle Illustré

Jérémy Lacoche, Thierry Duval, Bruno Arnaldi, Eric Maisel, Jérôme Royan

# Redistribution et Plasticité pour les Interfaces Utilisateurs 3D: un Modèle Illustré

Jérémy Lacoche, Thierry Duval, Bruno Arnaldi, Eric Maisel and Jérôme Royan

**Abstract**—In this paper we propose a model to handle redistribution for 3D user interfaces. Redistribution consists in changing the components distribution of an interactive system across different dimensions such as platform, display and user. Our work is based on previous models that ease the creation of 3D plastic user interfaces, interactive systems that can handle context of use modifications while preserving usability. We extended these models in order to include redistribution capabilities. The final solution lets developers create applications where 3D content and interaction tasks can be automatically redistributed across the different dimensions at runtime. The proposed redistribution process includes an automatic detection of these platforms and a meta-user interface to control the redistribution granularity. In order to illustrate this model, we describe three different scenarios of redistribution between a tablet and a CAVE for a 3D application. We show how redistribution can be used at runtime to combine these platforms, to switch seamlessly from one platform to another one and last how redistribution can be used to create a collaborative context of use.

**Index Terms**—Plasticity, Redistribution, 3D User Interfaces

## 1 Introduction

Today, users have access to a wide variety of platforms such as mobile devices, desktop computers and immersive systems. Therefore, users are more frequently confronted with situations where they have to move from one platform to another [8] or to combine them. These possibilities directly refer to "distributed user interfaces" (DUI) and redistribution. A DUI is a user interface whose components are distributed across different dimensions such as platforms, displays and users [9] [18]. For instance, these components can be widgets, interactors, or content. The redistribution capability of an interactive system refers to its property to change statically or dynamically its components distribution [4]. It can include migration and replication mechanisms. Redistribution is directly linked to the plasticity concept which comes from 2D user interfaces. Indeed, plasticity is defined as the capacity of an interactive system to withstand variations of both the system physical characteristics and the environment while preserving its usability [22]. In 3D, some solutions exist for the creation of reconfigurable applications [10], adaptive ones [16] and some recent approaches tend to bring plasticity to 3D [12] [15]. Code interoperability and usability continuity whatever the context of use has to be guaranteed to be considered as plastic. The plasticity property is needed to handle redistribution, indeed, as the input and output capacities may vary from a platform to another one the components migration or replication implies adaptations of these components.

Redistribution and plasticity have already been well explored for 2D user interfaces but less for 3D. However, in the last few years interest for 3D user interfaces has grown. This kind of interactive systems includes Virtual Reality (VR) and Augmented Reality (AR) applications. This new trend is possible thanks to the improvement in graphics performance of devices such as PCs or smartphones and also thanks to the generalization of VR and AR devices. In order to ease the implementation of such applications, our approach proposes to consider redistribution for 3D user interfaces.

Our contribution is a new model for developers to help them in the creation of 3D user interfaces that can be dynamically redistributed across different platforms, users and displays. The solution is based on the models presented by Lacoche et al. [15] that let developers create 3D applications independently from concrete interaction devices. At runtime, based on a client-server architecture, new platforms are automatically detected and a synchronization is performed between the different application instances. Furthermore, a meta-user interface is provided to the end user to enable him to control the redistribution process. To illustrate our solution, we present three different scenarios of redistribution between a tablet and a CAVE [7] for a furniture planning application. In these examples, we show how the virtual environment and the interaction tasks can be distributed across the two platforms in order to combine them, to switch seamlessly from one platform to the other one and also to create a collaborative context of use.

This paper is structured as follows: first we review the details of the redistribution concept and we present some related work. To continue, we describe our models for the creation of plastic 3D user interfaces and then how these models have been extended to support redistribution. Next, we present the three examples of redistribution between a tablet and a CAVE for a furniture planning application developed with our models. Last, we conclude and give some directions for future work.

## 2 Related Work

As said, a DUI is a user interface whose components are distributed across different dimensions [9]. For 3D user interfaces we consider three dimensions of distribution from the ones described in [9] and [18]:

- **Display**. The application content is displayed on one or multiple devices. Common examples in 3D for this kind of distribution are multiple display systems such as CAVEs [7].
- **Platforms**. The application runs on a single computing platform or is distributed across multiple ones. These platforms may be heterogeneous (operating system, computing power, devices plugged). For 3D applications, in that category we can talk about cluster approaches which combine connected homogeneous computers to run a VR application with high performances. It can also concern interactive systems where the interactors of a same application are distributed across different platforms.


- *Jérémy Lacoche is with IRT b<>com/IRISA/INRIA e-mail: jeremy.lacoche@b-com.com*
- *Thierry Duval is with Lab-STICC/Telecom Bretagne/IRT b<>com e-mail: thierry.duval@telecom-bretagne.eu*
- *Bruno Arnaldi is with IRISA/INRIA/INSA Rennes/IRT b<>com e-mail: bruno.arnaldi@irisa.fr*
- *Eric Maisel is with Lab-STICC/ENIB/IRT b<>com e-mail: maisel@enib.fr*
- *Jérôme Royan IRT is with b<>com e-mail: jerome.royan@b-com.com*


*Manuscript received 18 Sept. 2014; accepted 10 Jan. 2015. Date of Publication 20 Jan. 2015; date of current version 23 Mar. 2015. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.*

- **Users**. In that case the application is shared by multiple users. This dimension is directly linked to the two other ones as the different participants can use different displays and platforms. In 3D, this dimension directly refers to Collaborative Virtual Environments (CVE). CVE include concepts for sharing virtual worlds between different platforms and users.

Redistribution consists in changing the distribution of an interactive system on these different dimensions. According to Demeure et al. [8], redistribution can be system-initiated (the system performs automatically the redistribution), user-initiated (the user initiates and parametrizes the redistribution), or mixed-initiated (the user and the system collaborate to perform the redistribution). According to Calvary et al. [5], redistribution can be performed on the fly (at runtime) or between sessions and the granularity for distribution may vary from application to pixel level:

- At **application level**, on the platform or user dimension, the application is fully replicated or fully migrated on a distant platform. The application may be adapted to its new context of use which can include platform capabilities and user preferences. Full replication implies state synchronization to maintain consistency between the different instances of the application. On the contrary, for a full migration, each platform runs its own independent version and no synchronization is performed. For instance, Bandelloni and Paterno [2] present a bank 2D application which can fully migrate from a PDA to a PC while keeping the application runtime state during the process.

- At **workspace level**, workspaces can be redistributed on the platform, display and user dimension. A workspace is an interaction space that groups together interactors that support the execution of a set of logically connected tasks. For instance, the painter metaphor [20] includes two workspaces: the palettes of tools on a mobile device and the drawing area on an electronic white board.

- At **domain concept level**, physical interactors can be redistributed on the different dimensions. In 3D, it corresponds to the interaction techniques and widgets. For instance, BUILD IT [19] is a tool dedicated to the design of factories. It is composed of two projective displays. A horizontal one allows the users to have a 2D view of the factory and provides them 2D interaction for object manipulation. A vertical display provides a perspective view of the result. In the same way, in [17], physical interactors for navigation, pointing and application control are distributed on a tablet in order to interact with content in an immersive system. In these two cases the system distribution is not performed automatically as it has only been designed to work with these two platforms.

- At **pixel level**, view continuity is ensured across different displays thanks to a distribution on the display and the platform dimensions. In 3D, this kind of distribution is performed for multiple display systems such as CAVEs [7]. In this case, an application can be distributed on a cluster of PCs and rendered on multiple displays with view continuity.

In order to handle redistribution on the different dimensions and at the different levels of granularity, solutions designed for 2D user interfaces can be found. For instance, VIGO [14] is an architecture that supports ubiquitous instrumental interaction among multiple devices and computers. The 4C reference framework [8], introduced by Demeure et al., is divided in four dimensions: computation, communication, coordination, and configuration that capture the what, when, who, and how aspects of the distribution. It provides a meta-user interface in order to control the redistribution process. To continue, Melchior et al. [18] propose a peer-to-peer architecture for the creation of DUIs. It includes mechanisms for widgets migrations and for the adaptations of the widgets representations and interactions according to the context of use. Moreover, ZOIL [23] is a software framework for the developement of post-WIMP ("Windows Icons Menus Pointer") distributed user interfaces. It proposes a client server architecture with a transparent persistent mechanism for the synchronization between the different platforms.

In the field of 3D user interfaces, solutions to create DUIs also exist but they mainly focus on specific cases and do not let the end-user control the redistribution process at runtime. One specific case handled in 3D and cited before is the case of clusters of computers that manage multiple display systems such as CAVEs [7], Holostages, or Workbenches. In that case the system distribution is performed on the platform and display dimensions. The VR Juggler [3] framework and MiddleVR [1] propose such solutions. The second specific case handled in 3D is the field of CVE which need a distribution at the platform and user levels. It implies a state synchronization between the different users platforms in order to maintain a consistent application. Some architectures for CVE are reported in [11].

In this paper, we propose a solution that can handle distribution on the platform, display and user dimensions that consider the 3D specificities. In our case, the redistribution is user-initiated and controlled with an integrated user interface. We focus on redistribution for 3D user interfaces at application, workspace, and domain concept levels. Pixel level is not covered. Indeed, we consider that handling redistribution at the pixel level with high performances expectations is already a mature field of research while the other levels are less explored in 3D. The proposed solution can be interfaced with modern 3D frameworks, especially game engines in order to be easily integrated in the 3D developers and designers work-flow. One of the advantages of our approach is that any application developed with our models automatically benefit from redistribution capabilities.

## 3 Application Model for Plasticity

The link between redistribution and plasticity can be considered as bidirectional. Indeed, a modification of the context of use may imply a modification of the system distribution triggered by a plasticity mechanism. As well, a modification of the system distribution may also imply the need for adapting the application in order to fit the capacities of a new platform or the preferences of a new user. Most approaches to handle redistribution and plasticity are dedicated to 2D user interfaces and do not address new issues introduced by 3D user interfaces. Indeed, in 3D the user is interacting with more complex content, including 3D meshes with complex materials and behaviors. Furthermore, it includes a wider range of possible interaction devices and interaction techniques. Our approach aims to target these different issues.

That is why, in order to design 3D applications that handle redistribution, our application model represented in Figure 1 is based on the plasticity models for 3D user interfaces presented by Lacoche et al. [15]. First, an application is described with a set of high level interaction tasks. For 3D user interfaces, according to Hand [13], these tasks belong to three categories: selection and manipulation, application control, and navigation. The different tasks represents at a high level the application behavior and possibilities independently from the concrete implementation of the application. Dependencies between the tasks can be described by the developer. For instance, an application control task with a menu needs a selection task, therefore the two tasks are defined as dependent. These needed tasks and the dependencies must be provided by the application developer or the designer. The tasks can define different functions (the task events) that constitute the application logic such as adding an object into the scene or loading a new scene configuration, etc. Second, the application is described with its virtual environment. The virtual environment is composed of visual (3D content) and sound assets. Its edition is separated from the tasks. It can be edited separately, for instance in a game engine editor, or loaded with an X3D file depending on the implementation of the models used. In our case, we use an implementation based on Unity3D [2].

[1] http://www.middlevr.com/middlevr-sdk/
[2] https://www.unity3d.com/

In order to represent the device context of use, Lacoche et al. [15] also describe a device model for the description of any platform. They define a platform as a hardware environment composed of input and output devices and computing units. The goal of this device model is to describe precisely all the devices that can be used for interaction purposes at runtime. The model includes device capabilities, limitations and representations in the real world.

At runtime, the high level tasks are atomically associated with concrete application components according to the encountered context of use. This association is made with a scoring system that takes into account the platform capabilities and the user preferences. The description of this system is not the topic of this paper. A concrete application component is a software element that can be deployed in the final application in order to accomplish a task. For instance, it can correspond to the code for a 3D widget or an interaction technique. Tasks have to expose compatible concrete application components that will be possibly instantiated in the final application. To implement these concrete application components a model is also proposed. This model is a modification of PAC [6] and ARCH [1] models that lets the developer create application components such as interaction techniques and 3D widgets independently of concrete devices and of 3D frameworks. An application component is divided into four facets that decouple its different features:

- The Abstraction: it describes the semantics of the component and the function it can perform,
- The rendering presentation facet is the only facet depending on a 3D framework. It handles graphics output and physics. As said, in our case these components are developed with Unity3D. For a given application component, this facet can also define its representation in the virtual world. For instance, the 3D aspect of a widget will be defined in this facet.
- The logical driver handles input and output devices management. Its main use is for the development of interaction techniques. It implements the way the interaction technique is controlled according to a set of abstract interaction devices.
- the Control: it ensures the consistency between the rendering presentation, the logical driver and the abstraction.

An example that is needed in our example application described in Section 5 is the possibility to select and manipulate 3D objects. Different application components are compatible with this task. First, we can use a 3D-ray based interaction technique. This technique has multiple compatible logical drivers in order to be possibly driven by different kind of devices such as a 6-Dof tracker, a mouse or a gamepad. Second, we can use a 2D cursor for selecting and moving the objects on the screen plane. Different logical drivers also exist to control this technique based on devices such as a mouse and a multi-touch screen.

In the next section, we describe how these models have been extended in order to handle redistribution. These models are used in the redistribution process when the distribution modifications causes context of use changes.

## 4 Extension to Redistribution

The previous models let the developer create an application that can be adapted to the capabilities of a wide variety of platforms but does not include any mechanism to change the distribution of this application. Therefore we propose an extension of these models that makes the integration of redistribution capacities totally transparent and automatic for the developer. The process consists of distributing the high level tasks and the virtual environment across the different dimensions and can target the different granularity levels. The developer's work is to create high level tasks and implement the compatible interaction techniques that can be driven with a wide variety of interaction devices with the help of the models from [15] described in the previous section.

We added a built-in high level task and its corresponding application component in order to allow any developer to add redistribution

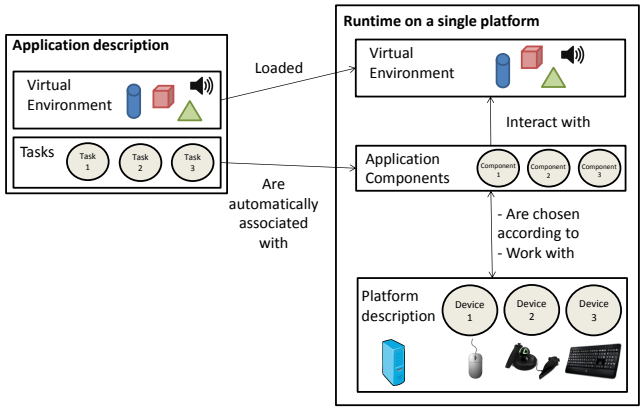


Fig. 1: An application is described by a virtual environment and high level tasks. At runtime, the application runs on a specific platform. Compatible interaction techniques are deployed to achieve the tasks. They are driven by the available devices.

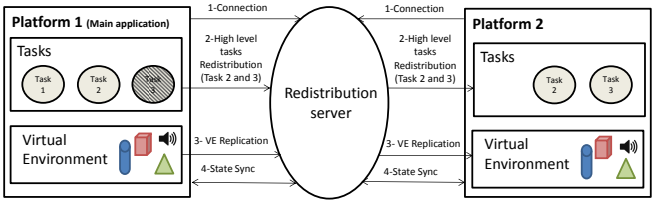


Fig. 2: The redistribution process is performed in four steps. First, the different platforms connect to the redistribution server. Then, the user initiates a new distribution of the system with the meta-user interface. Here he chooses to replicate the task 2 and to migrate the task 3 to a second platform. For these two tasks, compatible application components are automatically deployed on the second platform that fit its capabilities. The third step consists in replicating the VE from the first platform to the second one. Last, the redistribution server ensures state synchronization between the two platforms.

capability to his application. The application component for redistribution is also defined with our extension of the PAC and ARCH models described in Section 3. No logical driver is defined as no specific interaction device is needed by this component. The abstraction facet contains the redistribution logic and the rendering presentation facet contains the parts that are dependent to the target 3D framework. Regarding the process, redistribution needs a connection mechanism between the different platforms. This is needed for platforms discovery and state synchronization. To do so, we use a client/server architecture where the different platforms can register. For now, this feature is implemented with the network capabilities of the target 3D framework. Therefore, it is integrated into the rendering presentation facet. We chose this solution in order to rapidly create prototypes. However, as future work, this mechanism could become independent of the 3D framework and implemented in the abstraction facet. As proposed in the 4C reference framework [8], this component implements a meta-user interface for platform registration and to control the redistribution process. In our case, the redistribution is performed at runtime and is user-initiated. Indeed, the meta-user interface is proposed to the end-user of the application. The interface can be shown and hidden at runtime. The redistribution process is then performed in four different steps as shown in Figure 2.

The first step consists of connecting to the redistribution server. The IP address of the server can be given in the meta-user interface or with an XML configuration file. This step must be performed on the platform where the application is running and on each platform that must

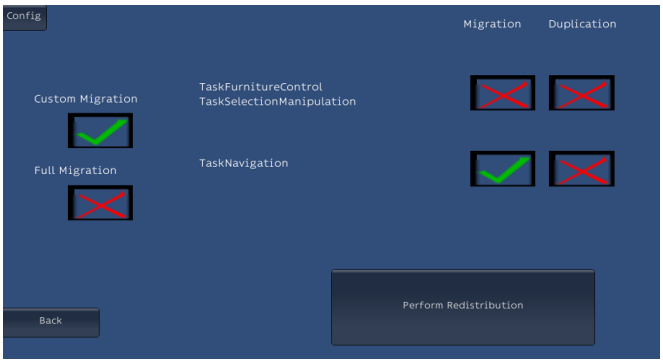


Fig. 3: The part of the meta-user interface that controls the redistribution of high level tasks. In this example, three tasks can be redistributed: navigation, selection/manipulation and application control. The last two ones are dependent.

be available for redistribution. On these distant platforms, an empty application runs which contains the framework that handles redistribution and can run an application according to the models presented in Section 3.

The second step consists of configuring the desired redistribution with the meta-user interface. First, the user chooses the platform on which the application will be redistributed from a list of available ones. In our case, the basis of the redistribution process is made on the platform dimension. However, as each platform may manage another display and may be used by another person, user and display dimensions can also be targeted. Then, the user configures the high level tasks distribution across the two platforms. As shown in Figure 3: multiple choices are given to the user in the menu:

- Full migration: all tasks migrate. Each platform runs an independent version of the application.
- Partial migration: the user chooses which task(s) will migrate to the distant platform. The application is distributed and so shared between the two platforms.
- Partial replication: the user replicates some tasks to the distant platform. He will able to perform these tasks on the two platforms within the same shared application.
- Full replication: all tasks are replicated and can be performed on different platforms in the same shared application.

Dependent tasks have to be redistributed together. Therefore, they are grouped into the menu as shown in Figure 3. In this figure the application control task is dependent to the selection and manipulation task. On the other platform, thanks to the plasticity models described in the previous section, an application component is automatically associated with each redistributed task. As said, these components are chosen in order to fit the platform capabilities in terms of device availability.

When the redistribution of tasks has been done, the third step consists in fully copying the virtual environment to the distant platform. This virtual environment includes 3D meshes, their materials, and sound assets. To perform this copy, we consider three solutions. For now, only the first one is implemented in our solution.

- Assets are known in the distant platform. Only the names are transmitted.
- Assets are not known but can be downloaded from a distant server. In this case, URLs are provided.
- Assets are not known. For instance in a case of a 3D painting application, the user is editing a new 3D content. Here, assets can be streamed over the network.

The last step consists of synchronizing the different platforms. In case of full migration, no synchronization is performed because each platform runs an independent version. The synchronization is performed as long as all platforms are connected to the redistribution server. Two kinds of information are synchronized between the different instances of the application. First, the 3D objects transforms are synchronized in order to maintain a consistency between the different 3D worlds. Second, the events of high level tasks are also synchronized. The events constitute the logical implementation of the application and and have to be synchronously performed on each application instance. These events are transmitted with their corresponding parameters through the network as text messages in order to be triggered distantly. An example of an event given in the example application in Section 5 is the addition of a 3D object into the scene.

## 5 Examples of Redistribution

In order to illustrate the redistribution possibilities offered by our models, we present different use cases that are based on a furniture planning application. This application consists in laying-out an empty room with furniture. Its goal is to help people to plan the use of particular premises. According to the application model described in Section 3, at the task level, the application is composed of three tasks. First, a navigation task is needed in order to navigate within the room. Second, we need an application control task for adding furniture into the room with the help of a menu. Last, we need a selection and manipulation task for moving furniture and for menu selections. These two last tasks are defined as dependent: indeed selection possibilities are needed for interacting with the menu. In these different cases we use two platforms. First, we use a mobile device which is an Android tablet. Then, we use an immersive system, a CAVE [7] with dimensions: 9.6m length $\times$ 3.1m height $\times$ 3.0m width. MiddleVR is usedTo handle the different screens and clustering. Even if they are not present in these examples other platforms could also be considered such as Head-Mounted-Displays (HMD) and desktop environments.

As described in section 4, the redistribution process starts with the connection of the tablet and the CAVE system to the redistribution server. For all the presented cases, the application is first launched on a tablet. The point of view is chosen by the navigation task according to the platform capabilities described in the device model of the application [15]. In the navigation task a function checks which are the properties of the display device used in order to set the first position of the main camera. Here, a plan view of the scene is chosen for the tablet as shown in Figure 4a. According to the automatic adaptation process described in Section 3, one concrete application component is deployed for each needed task. Each component is chosen in order to fit the platform capabilities. First, for the application control task, a 2D menu is instantiated with the list of furniture that can be added. According to its implementation the menu can be hidden if needed. For the manipulation task, an interaction technique based on the multi-touch capabilities of the tablet is deployed. With this technique the user can translate the objects onto the floor with one finger and rotate them around the up axis with two fingers. For the navigation task, a pan and zoom navigation technique is deployed. With the multi-touch capabilities, the user can translate the point of view and can zoom the scene while keeping the plan view of the room.

### 5.1 Redistribution for platform switching

Today, users are more frequently confronted with situations where they have to move from one platform to another [8]. This is one scenario possible for our furniture planning application. Indeed, the application may be used on a wide variety of different platforms such as desktop environments, smartphones, immersive systems, touch tables, etc. All platforms do not offer the same possibilities and therefore some can be more adapted to specific needs. Therefore we want to ensure for the end user seamless transitions between these different platforms. This example demonstrates how the redistribution capabilities of our solution can ensure usability continuity during these changes of hardware environment. In this scenario the redistribution is performed on the platform and display dimensions and at the application level.

First, the user is interacting on the tablet at his desk. With this tablet he can also work while mobile. All the tasks are available as corresponding application components are instantiated. However, the tablet only offers a 2D plan view of the result and the user would like to have a 3D view at scale one in order to better perceive the volumes. To do so, an immersive system is available: a CAVE. The meta-user interface allows the user to perform a full migration of his application to this platform. The application totally migrates to the CAVE, all tasks and all contents, nothing remains on the tablet. The user can now be immersed at scale one an continue to fine-tune the layout of the room. Usability continuity is ensured thanks to our plasticity mechanisms. The application is adapted to the target platform. Indeed, as described in Section 3, application components are chosen according to the new platform capabilities. In that case, a 3D ray-based manipulation technique is deployed. The position and the rotation of the ray are set with the tracked flystick and its buttons are used for object selections and to change the ray length. For the navigation task, a walking navigation metaphor is deployed. The tracked head position combined with the flystick joystick are used to move the point of view. For the application control task, a 3D movable menu is deployed. The 3D ray is used to interact with this menu.

When the user has finished his work he may want to continue his work while mobile. For example, he would go showing the result to a colleague. Therefore, the meta-user interface is also available in the CAVE and so the inverse process is also available.

### 5.2 Redistribution for platforms combination

This example demonstrates how redistribution can be used in order to combine different platforms. In that case redistribution is performed on the display and platform dimensions and at the domain concept level. Our example is based on the World-In-Miniature technique [21] which provides the user with a handheld model of the virtual environment at a smaller scale. It can be used for manipulating virtual objects or for navigation. This miniature representation is directly rendered in the virtual world; here, we propose to distribute this technique onto a tablet in order to control the furniture planning application in a CAVE system. The user will be able to interact with the tablet while being immersed at scale one in the CAVE. This use case can be useful for novice users who are not confident with 3D interactions and may prefer more commons multi-touch interactions.

The user chooses a partial migration to the CAVE, only the navigation task migrates to the distant platform. Other tasks remain on the tablet. This choice is made with the meta-user-interface as shown in Figure 3. In the CAVE, the navigation task places the point of view inside the room in order to immerse the user in it. As described in Section 5.1, an interaction technique based on a walking metaphor controlled with head tracking and a joystick is deployed in the CAVE for this navigation task. At this time the application is distributed on two platforms and displays. First, as shown in Figure 4a, on the tablet the redistributed World-In-Miniature. The virtual world is displayed at a lower scale with a plan view. Moreover, as said before, a 2D menu for application control and a multi-touch interaction for selection and manipulation are deployed. Second, as shown in Figure 4b, at the same time the user is immersed at scale one into the room in the CAVE and can navigate in it. Our transparent synchronization mechanism ensures the consistency between the two parts of the application. Indeed, the synchronization of the 6 DoF transforms of the objects between the two platforms ensures consistency when the user moves an object on the tablet. As well, the command for adding an object into the room is also synchronized.

This scenario can be useful for novice users not comfortable with 3D interactions. Indeed, the user can interact with the usual and easy-to-use multi-touch capacities of the tablet while being immersed at the same time in the 3D world with the CAVE.

### 5.3 Redistribution for collaboration

In this example, we demonstrate our redistribution process can be used in order to create a Collaborative Virtual Environment (CVE). Here, redistribution is performed on the user, platform and display dimensions and at the application level. Indeed, the replication capabilities included in our solution lets any user to start at any time a collaboration with another person using a different platform. With this feature any application developed with our models including the furniture planning one automatically benefits from collaboration capacities.

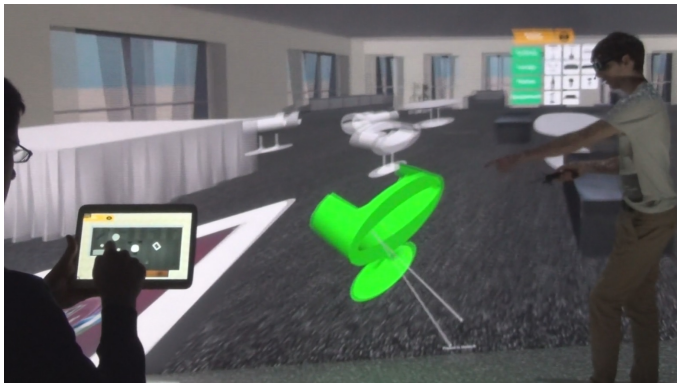

Fig. 5: With a full replication from its tablet the user can start a collaboration with a colleague in a CAVE

The first user has performed a first configuration of the empty room with his tablet and now wants to share his result and wants to finish it with another user. Therefore, as shown in Figure 5, he performs a full replication from the tablet to the second user platform: the CAVE. All tasks are replicated, navigation, selection and manipulation, and application control. Therefore the two users have now the same interaction capabilities. The application components instantiated for these different tasks are the same than for the two scenarios described in the two previous sections. In this case, the collaboration is asymmetric as the two persons are using different platforms and different interaction techniques. However, the plasticity property of the system ensures usability continuity between the two platforms, the interaction capabilities remain the same. A collaboration with two similar systems could also be performed. Here, the collaboration is co-located, both users are situated in the same place and can directly communicate about the result. However, the collaboration could also be distant. Indeed, our architectures makes possible to have distant connections to the redistribution server. The synchronization performed by the redistribution process ensures a high consistency between the two instances of the application. Both users are interacting in the same virtual environment. In order to provide awareness about the activity of the distant user, for now only the view frustums of each user is represented in the virtual environment. Future work could include different awareness mechanisms, for instance trying to make the distant user perceive his current context of use.

## 6 Conclusion and Future Work

In this paper we introduce a new model to handle redistribution for 3D user interfaces. Based on previous work on plasticity for 3D user interfaces, our solution eases the development of 3D user interfaces with redistribution capabilities. Redistribution can be performed on the display, platform and user dimensions and can target three levels of granularity: application, workspace, and domain concept levels. Our approach is based on a client-server architecture. Redistribution can be performed at runtime by the user with an integrated user interface: the meta-user interface. Plastic models ensure usability continuity whatever the new distribution chosen. The distributed application will fit each target platform properties. With this approach, any application developed with our models automatically benefits from redistribution capabilities.

To illustrate these possibilities, we have presented three examples of redistribution on different dimensions and at different levels for a furniture planning application. These examples show how redistribution can be used to switch from a mobile platform to an immersive one,


(a) (b)

Fig. 4: The redistributed World-In-Miniature: an example of redistribution that demonstrates how it can be used for platform combination. Here the redistribution is performed between a CAVE and a tablet. (a) On the tablet, the user has a plan view of the virtual world at a reduced scale with 2D interaction capabilities. (b) At the same time, the user is immersed at scale one in the same shared virtual world in a CAVE.

to combine these two platforms, and finally to create a collaborative context of use between them.

Future work could consist of automating the redistribution process in order to be in some cases system-initiated or mixed-initiated. Indeed, for now the process is only user-initiated with the help of the meta-user interface. For instance, this kind of approach could consist in finding the right platform or the right user for each task according to the platforms capabilities and the user preferences.

## REFERENCES


[1] A Metamodel for the Runtime Architecture of an Interactive System: The UIMS Tool Developers Workshop. *SIGCHI Bull.*, 24(1):32–37, Jan. 1992.

[2] R. Bandelloni and F. Paternò. Migratory user interfaces able to adapt to various interaction platforms. *International journal of human-computer studies*, 60(5):621–639, 2004.

[3] A. Bierbaum, P. Hartling, P. Morillo, and C. Cruz-Neira. Implementing Immersive Clustering with VR Juggler. In *ICCSA 2005*, pages 1119–1128, Berlin, Heidelberg. Springer-Verlag.

[4] G. Calvary, J. Coutaz, O. Daassi, L. Balme, and A. Demeure. Towards a new generation of widgets for supporting software plasticity: the comet. In *Engineering Human Computer Interaction and Interactive Systems*, pages 306–324. Springer, 2005.

[5] G. Calvary, J. Coutaz, D. B. Thevenin, L., M. Florins, Q. Limbourg, N. Souchon, J. Vanderdonckt, L. Marucci, F. Paterno, and C. Santoro. The CAMELEON Reference Framework. *Deliverable D1.1*, 2002.

[6] J. Coutaz. PAC, on object oriented model for dialog design. In *Interact'87*, 1987. 6 pages.

[7] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The CAVE: Audio Visual Experience Automatic Virtual Environment. *Commun. ACM*, 35(6):64–72, June 1992.

[8] A. Demeure, J.-S. Sottet, G. Calvary, J. Coutaz, V. Ganneau, and J. Vanderdonckt. The 4C Reference Model for Distributed User Interfaces. In *ICAS 2008*, pages 61–69, March.

[9] N. Elmqvist. Distributed user interfaces: State of the art. In *Distributed User Interfaces*, pages 1–12. Springer, 2011.

[10] P. Figueroa, M. Green, and H. J. Hoover. InTml: A description language for VR applications. In *Proceedings of the Seventh International Conference on 3D Web Technology*, Web3D '02, page 5358, New York, NY, USA, 2002. ACM.

[11] C. Fleury, T. Duval, and V. Gouranton. Architectures and Mechanisms to Maintain efficiently Consistency in Collaborative Virtual Environments. In *SEARIS 2010*, Mar.

[12] J. Gonzalez-Calleros, J. Vanderdonckt, and J. Muoz-Arteaga. A structured approach to support 3d user interface development. In *ACHI 2009*, pages 75–81, Feb.

[13] C. Hand. A survey of 3D interaction techniques. In *Computer graphics forum*, volume 16, pages 269–281, 1997.

[14] C. N. Klokmose and M. Beaudouin-Lafon. VIGO: Instrumental Interaction in Multi-surface Environments. CHI 2009, pages 869–878, New York, NY, USA, 2009. ACM.

[15] J. Lacoche, T. Duval, B. Arnaldi, E. Maisel, and J. Royan. Plasticity for 3D User Interfaces: new Models for Devices and Interaction Techniques. In *EICS 2015*. ACM.

[16] I. Lindt. *Adaptive 3D-User-Interfaces*. PhD thesis, 2009.

[17] D. Medeiros, F. Carvalho, L. Teixeira, P. Braz, A. Raposo, and I. Santos. Proposal and evaluation of a tablet-based tool for 3D virtual environments. *SBC*, 4(2):31, 2013.

[18] J. Melchior, D. Grolaux, J. Vanderdonckt, and P. Van Roy. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In *EICS 2009*, pages 69–78. ACM.

[19] M. Rauterberg, M. Fjeld, H. Krueger, M. Bichsel, U. Leonhardt, and M. Meier. BUILD-IT: a planning tool for construction and design. In *CHI 1998*, pages 177–178. ACM.

[20] J. Rekimoto. Pick-and-drop: A Direct Manipulation Technique for Multiple Computer Environments. UIST 1997, pages 31–39, New York, NY, USA. ACM.

[21] R. Stoakley, M. J. Conway, and R. Pausch. Virtual reality on a WIM: interactive worlds in miniature. In *CHI 1995*, pages 265–272. ACM.

[22] D. Thevenin and J. Coutaz. Plasticity of user interfaces: Framework and research agenda. In *Proceedings of INTERACT*, volume 99, page 110117, 1999.

[23] M. Zöllner, H.-C. Jetter, and H. Reiterer. *ZOIL: A design paradigm and software framework for post-WIMP distributed user interfaces*. Springer, 2011.