Direct Manipulation of User Interfaces for Migration

José Pascual Molina Massó[1,2], Jean Vanderdonckt[1], Pascual González López[2]
[1]Université catholique de Louvain,
Belgian Lab. of Computer-Human Interaction
Place des Doyens, 1
B-1348 Louvain-la-Neuve, Belgium
{molina, vanderdonckt}@isys.ucl.ac.be

[2] Universidad de Castilla-La Mancha,
Lab. of User Interaction & Software Engineering,
Inst. de Investigación en Informática de Albacete (I3C)
Campus universitario s/n, 02071 Albacete, Spain
{jpmolina, pgonzalez}@info-ab.uclm.es

ABSTRACT
From a topological model of a working environment, MIGRIXML automatically generates a virtual reality environment for controlling the run-time migration of a graphical user interface from one computing platform to another one (e.g., from a desktop to a pocket computer), from one interaction surface to another (e.g., from a laptop to a wall screen) at run-time. For this purpose, any user interface subject to migration is described in USer Interface eXtensible Markup Language regarding its look & feel as well as the platforms and the surfaces involved in the migration. Each interface, in part or in whole, can be attached to a platform or a surface, detached from it, and migrated across platforms or interaction surfaces. Instead of communicating data and code during the migration, the description of the user interface of concern is wirelessly passed from one platform to another one to be regenerated on the target platform. To ensure a continuous control of the run-time migration, MIGRIXML automatically generates a world model representing the context of use where the source/target platforms/interaction surfaces are represented. Finally, migrating a user interface becomes as natural as its direct manipulation from one platform to another exactly in the same way as it is done on a single platform.

ACM Classification: H5.2 [**Information interfaces and presentation**]: User Interfaces - *Graphical user interfaces*.

General terms: Design**,** Languages

Keywords: Virtual environment, migration

INTRODUCTION
End users of modern interactive systems are confronted with a wide variety of computing platforms to support their interactive tasks ranging from the mobile phone to a wall screen [2]. Not all these platforms are appropriate for every context of use [3]: the mobile phone or a PocketPC are more appropriate when the user in mobile, moving *with* the platforms, while desktop and wall screens are more appropriate when the user is stationary, perhaps moving *across* platforms. A same platform can even serve in both situations: a laptop is useful the user is moving and when she is working at a static place. Due to these very different platforms and due to the various working conditions, the user may need migrating tasks and data between platforms [12,15].

The *migration* of a User Interface (UI) is hereby referred to as the action of transferring a UI from one source location to a target one. A location could be any computing platform, an interaction surface [7] or an interaction space [1]. Therefore, a migration could be interpreted as transferring a UI for instance from a desktop computer to a handheld device.

A UI is said to be *migratable* if it holds the migration ability. A migration is said to be *total*, respectively *partial*, when the whole interactive application, respectively the UI, are migrated [1,2,3]. If the UI is decomposed into two components, the control which is responsible for the UI behavior and the presentation which is responsible for presenting information to the user, *control migration* [3] migrates only the control component while the presentation remains. In *presentation migration* [3], the situation is the inverse: the presentation component is migrated while the control remains on the source platform. When it is *mixed* [3], different parts of both the control and the presentation are migrated.

To support all these different cases of migration, a special UI is required that will perform the required steps to conduct the migration, such as identification of migration possibility, proposal for migration, selection of migration alternative, and execution of the migration itself. Since these types of migrations and underlying steps require complex handling of UI events and procedures, the UI responsible for migration is even more complex and not always visible to the eyes of the end user. This UI is referred to as the *meta-user interface* in [1], i.e. the UI for controlling the run-time migration of the UI of the interactive systems. This term will be used throughout the rest of this paper.

In most of research/development projects involving some form of migration [1,2,3,4,5,9,10,12,13,19,20,21,22,23,24], the meta-UI is implemented in very different ways with different manifestations. It is not made explicit whether the meta-UI is *system initiated* (the system initiates the migration), *user-initiated* (the user initiates the migration), or *mixed-initiated* (the user and the system collaborate to perform the migration). In addition, the interaction techniques involved in the meta-UI do not deal directly with the components of the UI to be migrated. This situation may confuse the user when, how and what parts of the UI need to be migrated. To fill these gaps, we developed a meta-UI as a virtual environment for controlling run-time UI migration in all the above situations with the new advantage that all the steps of the migration are graphically represented in virtual reality that mimics the real world.

This paper focuses on the original part of providing a virtual control environment for migrating parts or whole of a UI from one platform to another. Therefore, other aspects such as platform discovery, platform management underlying architecture, etc. are not addressed as they are based on previous work done in this area [3,5,6,10]. The rest of this paper is structured as follows: the following section reviews some work related to migration and closely examines the shortcomings of how the meta-UI is implemented in these works. Section 3 summarizes the contents of the models involved in the migration and the meta-UI in virtual reality we developed. Section 4 details the complete implementation of this meta-UI and the migration process that is controlled behind the direct manipulation of the meta-UI. Section 5 illustrates a case study with two migration types using our meta-UI MIGRIXML.

IUI'06, January 29–February 1, 2006, Sydney, Australia.

RELATED WORK

In this section, we review some work done in the area of UI migration with an emphasis on how the meta-UI was designed and developed.

Probably the first work done in the history of migration is [5]: complete interactive applications can roam over a network thanks to a single migration command enabling migration to a host or a server. The migration is total, mixed, and system initiated. The meta-UI is hard coded in the application that needs to migrate. In [24], a similar system stops and saves a web browser session, migrates it to another browser, possibly located on another platform, and restores the previously saved session. The migration is total, mixed, and user-initiated. The meta-UI is implemented separately. The *MIGRATION* project [2,3] supports a similar migration but across different platforms equipped with a web browser such a mobile phone, a PocketPC, a laptop or a desktop. The migration is partial or total, presentation oriented, and user-initiated. For this purpose, a separate form-based GUI displays previously defined platforms from which and to which the migration is achieved. *CamNote* [1] is a slide manager distributed across a desktop and PocketPC in a cluster: when the PocketPC enters, respectively leaves, the cluster, the slides control is migrated to the PocketPC, respectively returned to the desktop: migration is partial, mixed, and system initiated. The meta-UI is embedded in the whole system, but developed on top of *I-AM*, a run-time infrastructure supporting migration and plasticity that can accommodate several configurations of distribution and migration [7,8]. *Drag & Pop*, *Drag & Pick* [4] are two interaction techniques enabling the user to quickly reach icons across several screens aligned side by side: when the user moves the cursor towards one of those screens, the potential target icons approach the user's pointer. The migration is partial, presentation oriented, and mixed-initiative. The meta-UI is the interaction technique itself, without other control. In *Aura* [9], the information is presented on the wall screen that is the closest to the user depending on her location in the building. The migration is partial, presentation oriented, system initiated and the meta-UI is totally invisible. [22,23] do the same for a collaborative virtual environment. The *Stanford Interactive Mural* system [13] is similar except that wall screens could be hung up on different walls. In *Detachable Interfaces* [10], a portion of a UI can be detached from a UI belonging to one platform to another. Detaching a UI is achieved by dragging a portion of the UI and dropping it outside the UI: the migration could be partial or total, presentation-, control-oriented or mixed, and user-initiated. Since neither the source-target platforms are represented nor the exchange space between them, the user may get lost. This is why *Augmented Surfaces* [19] and *Pick & Drop* [20] rely on a physical space (e.g., a portion of a table) to materialize an area where the UI can be placed before transfer. Again, there is no concrete representation of the target platform although there is a representation of the space between them. To improve this, *Proximal Interactions* [21] provides a screenshot of the target platform as an image on the source platform to support the Drag & Drop. A representation of the environment in the form of a 2D iconic map is found in ARIS [6], showing walls as if they have been pulled down on their back side, and enables relocation of a window by dragging its representation in the map. To aid user orientation, an arrow had to be added in the map to communicate a location and view direction.

In conclusion, we can observe that most meta-UIs developed so far share the following shortcomings: there is no graphical representation of the complete environment in which the migration occurs, a cognitive disruption may arise during migration time as this process is not continuously represented, the meta-UI is hard coded in the system and thus rather inflexible to cope with varying migration conditions and environments, the separation between the UI to be migrated and the data that populate the UI is not always clear, the meta-UI is not open for incorporating new platforms or spaces. In addition, existing environments for UI migration mainly support total migration. When they support partial migration, only contiguous parts could be migrated. UI migration usually occurs for the Web whereas our migration environment is not restricted to HTML. These aspects have been identified as crucial for the usability of multiple monitors tasks: tasks are usually distributed among platforms, but without any consistent way to distribute them [10], the bezel or the physical space between platforms introduces a discontinuity that may disrupts the fluency of the interactive task [4,25].

USIXML: USER INTERFACE EXTENSIBLE MARKUP LANGUAGE

To generate the virtual reality scene representing the migration environment, a suite of models is used describing relevant aspects of the problem in terms of USIXML (USer Interface eXtensible Markup Language), a XML-compliant language [14,28]:

1. The **Concrete User Interface** (CUI) model: decomposes a GUI into Concrete In-teraction Object (CIOs) that are characterized by various attributes (*id*, *name*, *icon*, *content* to save the current value of a widget before migration, *defaultContent*, *defaultValue*) and sub-typed into one of the two categories: *graphicalContainer* for all widgets containing other widgets such as page, window, frame, dialog box, table, or *graphicalIndividualComponent* for all other traditional widgets: text, video, image, radio button, drawing canvas,…

2. The **user** model: decomposes the users' population into a hierarchy of users' stereotypes, each one sharing a series of attributes such as skills, preferences, system experience, task experience, task motivation, abilities to use a modality,...

3. The **platform** model captures relevant attributes for each couple software-hardware platform and attached devices that significantly influence the context of use: a series of physical hardware devices (hardware platform components), a series of software components (software platform), the characteristics of the network, the capability to support wireless applications and browsing.

4. The **environment** model is a contribution in this paper. It describes properties of interest of the physical environment where the user is with the computing platform to accomplish her interactive tasks. Such attributes may be physical, psychological, and organizational. The physical part basically consists of a **scene** model inspired from VRML97/X3D [27,28], the multi-surface interaction ontology [8] and Stanford topology model [13] representing the topology of physical setup of the ambient environment of the user. Each *scene* is physically decomposed into a series of planes that are connected to each other, and which are in turn decomposed into *areas*. An ear is basically an interactive surface. Other components include *physical resources*. Each area may be an interaction surface: a monitor on a table, a wall screen, or any area where the UI is projected and/or recognized with computer vision techniques. Each area is described by attributes like dimensions, capabilities, angle with respect to reference area, relative position, relationships with other objects (i.e., left, right, top, bottom).

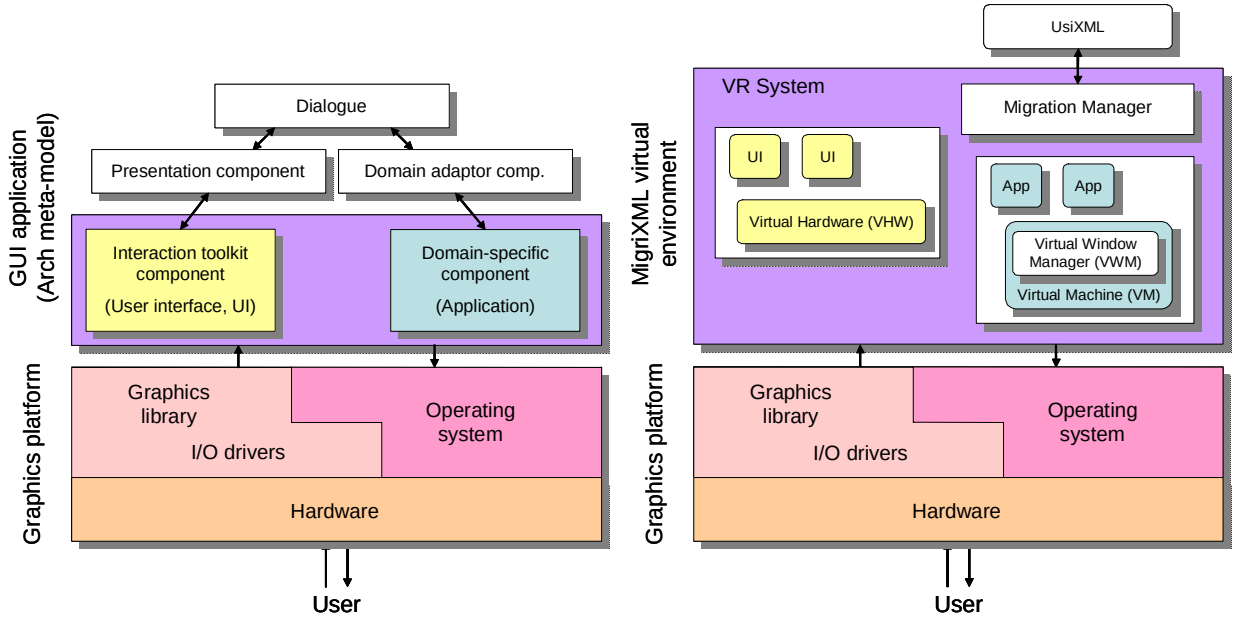

Figure 1. GUI application and platform architecture (left), and MIGRIXML architecture (right).

A VIRTUAL REPRESENTATION OF A REAL MIGRATION

The architecture and implementation of MIGRIXML are now detailed, as a virtual reality system representing the user's real environment, based on the models of previous section: the platforms found in that environment, the UI of interactive graphics applications that are executed on these platforms, and the user. Within this environment, the user is interacting with the platforms and the running applications as if they were their real counterparts. The user selects any application, the related UI emigrates from the source platform and immigrates in the target platform.

The MigriXML Architecture

In order to describe the software architecture of MIGRIXML, we firstly define the concepts of interactive graphical application, platform and user's environment, explaining the models and structures that have been taken into account to create the architecture of MIGRIXML. Beginning with the definition of interactive graphics application, it can be addressed taking the Arch meta-model, also known as the 'Slinky' meta-model [26] for extending the Seeheim model from three to five components:

1. *Domain-specific component*: also known as the functional core or the application, it manages the information of the system, and carries out the functionality that the system offers.
2. *Domain adaptor component*: it offers a unified, generic view of the functionality that implements the functional core, hiding the differences that any component playing that role may have with others.
3. *Dialogue component*: it mediates between the domain-specific and user interface functions. Between both extremes, this component is responsible for the task sequencing and for matching between the domain formalisms and those of the user interface. This component is the keystone of the meta-model, the top of the visual metaphor that this meta-model represents.
4. *Presentation component*: it provides to the dialogue component a set of logical interaction objects. These objects are mapped to toolkit-specific objects, which depend on the platform that executes the application.
5. *Interaction toolkit component*: it is responsible for handling the input/output de-vices, and is usually implemented as a software library or toolkit for UIs.

The Arch meta-model does not require to strictly implement the previous set of components in applications. In contrast, it aims at helping developers choosing the software structure that best suits their projects. That structure will depend on, among other factors, the necessity to adapt the application to different platforms, so if the software is well divided into different modules, the adaptation will be less costly. In our case, that meta-model turns out to be quite suitable to describe the interactive graphics applications as its components fit the previously USIXML introduced models. For example, the presentation component corresponds to the USIXML CUI model to a large extent, as the latter is an abstraction of the different widgets that toolkits offer, the FUI specific objects.

Following up with the definitions, the **platform** that executes the application and that renders its interface can also be described using a layered structure, identifying two main layers: an upper software layer and a lower hardware layer. On the one hand, the *hardware layer* represents the physical devices used for control and presentation, the physical interface that the machine offers to the user. On the other hand, the software layer consists of the *operating system* and the *I/O drivers*. In the case of *graphics platforms*, that layer usually includes the graphics library, and if the graphics platform is a windowing system, it also includes the *window manager*.

Finally, the platform or platforms that the user makes use of, together with the applications that they execute, are all part of the **user's environment**. In that environment, there are elements that may not be directly related to the computing set up, but they do relate to the user, such as the user's work space and the furniture included in that space. The specific needs of the user's environment at a given moment, or its evolution as the time goes by, can make necessary to remove one application from one platform and make it run in another platform. This means that the given application, which was originally adapted to the current platform, must be adapted in order to be executed in the new computing platform.

MIGRIXML enables the user to carry out that migration process without the physical presence of the involved platforms, in a virtual and interactive way, relying on the USIXML language and the models that have been previously explained.

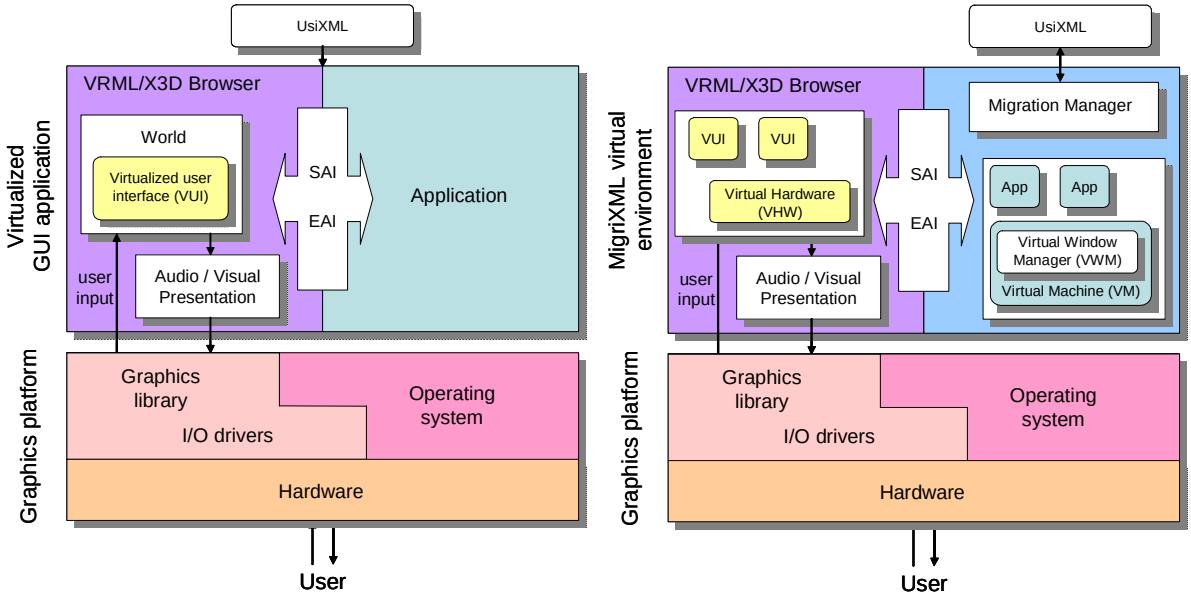

Figure 2. Virtualized GUI application (left), and implemented MIGRIXML structure (right).

It is a virtual reality system and, as such, its architecture is based on a graphics platform that executes a run-time simulation environment. In order to render the virtual representation of the user's environment, platforms and applications, MIGRIXML has been designed integrating these components in its own architecture (Figure 1):

- *User's environment*: it is the virtual world rendered by the virtual reality software.
- *Platform*: The hardware layer is called **Virtual Hardware** (VHW) and it is the visible part of the platform in the virtual world, the part that the user interacts with through the I/O channels that the Virtual Reality system offers. The software layer is named **Virtual Machine** (VM), and it includes the component that is responsible for the application windows, the **Virtual Window Manager** (VWM).
- *Interactive graphics application*: Its interaction toolkit component is substituted by a component that renders the UI in the 3D space where the virtual world exists. The domain-specific component is executed by the platform emulator, that is, the virtual machine introduced in the previous point.

The last component of MIGRIXML architecture is called **Migration Manager** (MM), whose role in this Virtual Reality simulation is to respond to use's requests for migration of interactive graphics applications from given platforms to selected targets. The migration manager exploits the USIXML specifications of these applications.

Implementation

When implementing MIGRIXML, it was decided to use the VRML97/X3D languages [28], which are standards for the description of Web-oriented virtual worlds. That decision was made taking into account their added flexibility, as there are many browsers that are able to interpret those languages and these browsers can be found for different operating systems, most of them free and with entry-level requirements, mainly a standard PC with 3D acceleration card. Besides, these languages can also embed code in the scene graph, and provide access from external code to the browser and the scene by means of a programming interface. For that reason, the audiovisual description and rendering of the user's environment is done using these Web3D standard languages, as well as the platforms' hard-ware and the user interfaces of the applications that these platforms execute.

In the particular case of the UIs –the interaction toolkit component, according to Arch meta-model-, their implementation is based on a set of PROTOs called **VUIToolkit**, which has been developed in both VRML97 and X3D versions [16]. The software structure that was used for the prototypes that are part of this toolkit is partly based on the work and developments of the former VRML Widget Working Group [27] and one of the original characteristics of this toolkit is that it trans-forms the standard plain 2D widgets into a truly 3D representation. In contrast to other approaches that are based on using the 2D graphics output of applications as image textures in the 3D virtual world, every widget of the VUIToolkit has real depth, they do not represent their behavior in a real 3D space.

This implementation of the widgets is aimed to match much better the mental model of the user. For example, if the user presses a button, it moves along the third dimension as the user would expect, instead of showing a predefined sequence of images that simulates that movements with 2D drawings. To state that difference, we use the term **Virtualized User Interface** (VUI) to refer to a 3D UI. Anyway, the selection of VRML97/X3D languages and the VUIToolkit for the implementation of MIGRIXML was not arbitrary, as the VUIToolkit was developed starting from the object classes described in the concrete model of USIXML language. Each prototype of VUIToolkit has a list of parameters that was first made including the attributes of its corresponding USIXML class, then adding new attributes as needed for transforming the concrete interface object –independent of any toolkit- into a final user interface object rendered in a 3D virtual world. Most important, the interactive graphics interface specified using USIXML language can, therefore, be transformed into a VRML97/X3D-virtual world in an automated way, just making use of the set of prototypes provided in the VUIToolkit (Figure 2).

As for the domain-specific component of the interactive graphics, Javascript and Java languages are used, as most VRML97/X3D can interpret Javascript code and execute Java code. This is due to the fact that both VRML97 and X3D specifications describe for these languages two programming interfaces to access the scene

graph, named SAI (Scene Authoring Interface) and EAI (External Authoring Interface). Be-sides, Java can easily be executed in different platforms thanks to its own binary for-mat and the use of a Java Virtual Machine for each particular platform. This characteristic of Java allowed us to leave aside, for this first implementation of MIGRIXML, the complex implementation of emulators for each specific platform. We added to each platform a virtual window manager as the component that simulates the windowing system, including special functionalities to manage our virtualized UIs. Clearly, the implementation of MigriXML would not be finished without the inclusion of the migration manager in the system. As its commitment has already been explained in the previous section, it will be omitted here. Next section will describe in depth the migration process, highlighting the actions performed by this component.

Migration Process in Detail

The migration process is divided in four stages (A, B, C and D), each one having a finite set of steps, representing a total amount of 14 steps (Figure 4). In the following sub-sections, that sequence of steps will be explained in an ordered way.

Stage A: Select an interactive application. In MIGRIXML, a migration process starts with the selection of the graphics application, action that is performed by the user in an interactive way. To do so, the user presses the button (M) –which stands for 'migrate'- that can be found in the button bar of the application window (a VUIToolkit window). As a result, the user 'grabs' the window, and from that very moment the window will follow the user wherever he or she points at within the screen of the source platform (step 1). That action is transmitted to the corresponding virtual window manager by means of sending out an event (step 2). Then, that component forwards the information to the migration manager of MIGRIXML environment (step 3).

Stage B: Select target platform. Once the migration manager receives the message from the virtual window manager, it broadcast a message to all platforms (step 4), which changes their current state to 'wait-for-selection' state. In the audio-visual part of each platform –the virtual hard-ware-, a TouchSensor is activated waiting for the user to perform a selection action, clicking on the target screen (step 5). Meanwhile, the user can point at whatever screen, and the virtualized application window will follow the cursor, being rendered according to the resolution and definition of the pointed screen. Just after selecting a platform (step 6), the virtual hardware casts an event to the virtual window manager (step 7), which proceeds by forwarding a message to the migration manager (step 8).

Stage C: Migrate to target platform. In this third stage, the selected application is re-generated according to the target plat-form. As a first step in this stage, the migration manager sends a command to the source platform (step 9), meaning that the application must be detached from it. Then, it also broadcasts a message to the rest of platforms, asking them to change to normal execution (step 10, not shown in figure for clarity reasons). The last step is carried out by own migration manager, which, taking the USIXML application specification as a source, re-generates the application for its execution in the new platform (step 11).

Stage D: Adapt to the target platform. This last stage is entered when the migration manager asks the target platform to host the re-generated application and its related user interface (step 12). The virtual window manager of the target platform launches the application, and its virtualized user interface is nested in the scene graph that represents the virtual hardware of the new platform (step 13). To achieve a correct visualization in the target screen, the user interface is adapted to it, reducing the size of the application window if it is larger than the screen itself, and applying the colors of the desktop theme of the target windowing system (step 14).

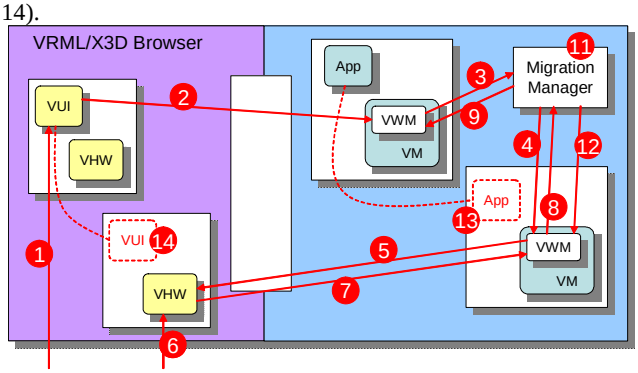

Figure 3. Steps involved in the migration process.

A CASE STUDY

In this section, a case of study will be used to exemplify the process of creation of the virtual world that, in MIGRIXML, represents the user's environment, the platforms and the applications that run in them. This case study will also illustrate in detail how migration processes are performed, from emigration to immigration.

User's Environment and Platforms

In this case of study, the user's environment is a small office, where five different platforms are at the disposal of the user to carry out his or her tasks: one PC, two lap-tops, one Pocket PC and a portable projector. The USIXML language is used to describe that environment and the platforms, saving in a XML specification the details that concern the developers involved in the case of study, such as software engineers and user interface designers. From that specification, it is generated a set of VRML97/X3D prototypes that will be used by MIGRIXML to represent the virtual world. Each of these prototypes has associated an interactive audio-visual representation, which is not the result of an unmanned automatic process that takes the specification as a source, but the result of a structured process of creation that is carried out by a virtual environment designer or a team of them. Even though the development of Virtual Reality systems is usually done relying on the experience and intuition of the designers, the development of the virtual environment of MIGRIXML is carried out following a concrete methodology, which can be summarized in three stages:

1. *Requirements:* Designers starts their work studying the USIXML specifications, in order to know the details of the user's environment or the platform whose virtual model must be created. A set of objectives and constraints are identified, which in this case are partly related to the characteristics of the VRML97/X3D languages, the world browser, and the simulation platform.
2. *Preparation:* A compilation of audio-visual material is carried out, such as photographs, sound or video recording, and any other material worth to be used later to produce the 3D models. Dimensions are measured, and sketches and mock-ups are created, such as paper or electronic prototypes.
3. *Design, test and optimize:* 3D creators, under the supervision of interaction experts, model the geometry of the objects, apply colors and textures, and add sounds and interactive elements to them. The outcome is tested, optimized in an iterative fashion, until an acceptable visualization is reached within the limits given by the objectives and constraints fixed in the first stage.

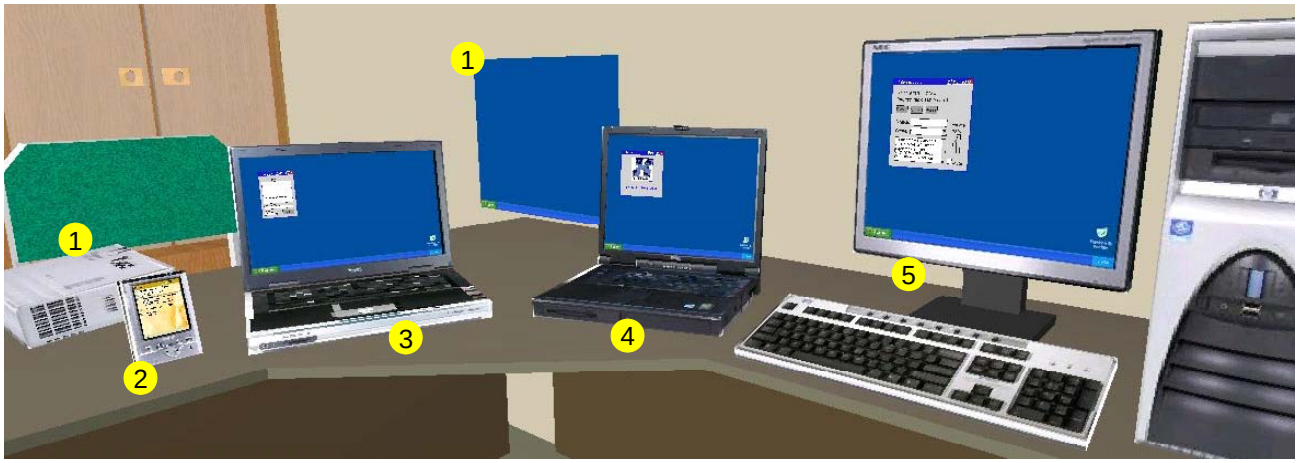

Figure 4. Overview of the user's environment of the case study, showing the five platforms.

No.	*Platform*	*Image size*	*Image ratio*	*Max image resolution*	*Image resolution*
1	Panasonic PT-LB10SU	¾ d*	4:3	800x600	800x600
2	Toshiba PocketPC e750	3.8"	3:4	240x320	240x320
3	Acer Aspire 2000	15"	16:10	1280x800	1280x800
4	Dell Latitude C840	15"	4:3	1600x1200	1024x768
5	NEC LCD1960NX	19"	5:4	1280x1024	1024x768

Table 1. Screen characteristics of the five platforms of the case of study.

The result of applying the previous methodology to the case study can be seen in the screenshot given in Figure 4, showing the user's environment and the five platforms (Table 1). Numbers have been added so that the visual representation of each platform can be easily matched with the screen characteristics given in the following table.

User Interface

The UIs that are included in our case of study correspond to two different applications: an *Internet Radio Player* and an *Instant Messaging Client*. The radio player includes controls to search radio stations, filter the search results and select the desired station, play, pause and stop, as well as volume controls. The latter includes allow the user enter text for new messages, and follow up the online conversation. Both inter-faces are specified in USIXML language using the models provided. In order to make this task easier, different tools have been developed, helping the UI designer to visually create these models. Thus, IDEALXML allows the designer to create the abstract model in a diagrammatic way [17], and GRAFIXML supports the visual creation of the CUI model (Figure 6). From the specified models, a final UI is generated as a VRML97 or X3D file, based on VUIToolkit (Figure 5).

Migrating from one Platform to Another

In this section, it will be described and illustrated how MIGRIXML environment can be used to perform the migration of an application, running in a given platform, to another target platform.

Migrating from one laptop to the other. In this first example, the user selects the Internet Radio Player, currently running in the Acer laptop, and takes it to the Dell laptop. The sequence of images illustrates the following process (Figure 7):

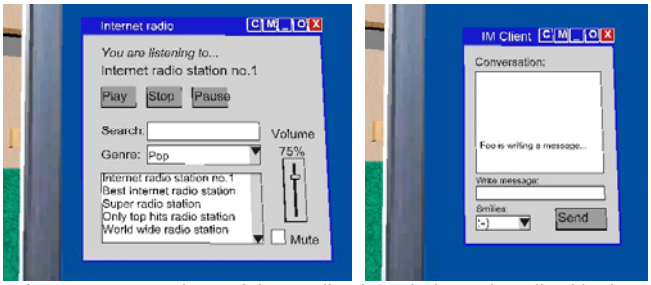

Figure 5. Screenshots of the application windows described in the case study: Internet Radio Player (left), Instant Messaging Client (right).

1. First of all, the users 'grabs' the application windows by pressing the button (M) located in its button bar. Then, the virtualized user interface communicates the user action to its corresponding virtual window manager, which forwards the information to the migration manager.
2. The user moves the cursor around the screen and the application window follows that movement. By then, the migration manager changed the state of other plat-forms, which are awaiting the next user action: selection of the target platform.
3. As the user moves the cursor around the environment towards the Dell laptop, the user points at the screen where a desktop image is being projected by the Panasonic device. The TouchSensor of that platform captures the user action and the migration manager is informed of that event. Then, the migration manager proceeds to nest the virtualized interface in the transformation hierarchy of that platform. As a result, the application window is rendered on that screen according to the resolution and definition of the image projector.
4. The user goes on moving the cursor and the window follows it until the Dell laptop is reached. Then, the user selects that laptop as the target platform with a mouse single click. However, the migration process is not finished until the migration manager re-generate the given application for the new platform, and once re-generated is hosted in the laptop, adapting itself to the new characteristics of the screen and the desktop.

Migrating from a laptop to the Pocket PC. In this second example, the user wants to 'grab' the Instant Messaging Client from the same starting point as in the previous example, but this time to achieve its adaptation to the Pocket PC. This time, the desktop characteristics of the laptop are quite dissimilar to those of the Pocket PC. Therefore, the application window must be adapted to the reduced size of the new display, and its appearance modified according to the Pocket PC desktop theme (Figure 8).

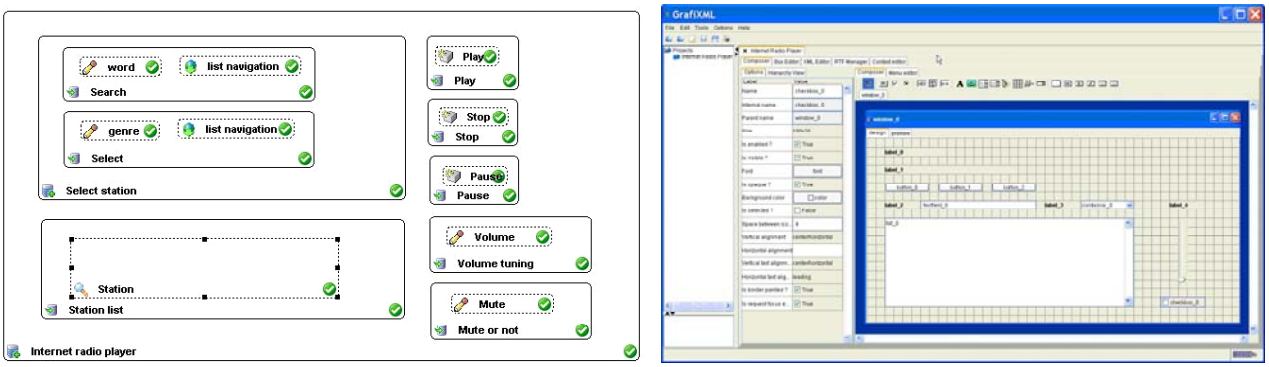

Figure 6. Abstract model of the user interface corresponding to the Internet Radio Player application, done with IDEALXML (left), and the concrete model of the same interface done with GRAFIXML (right).

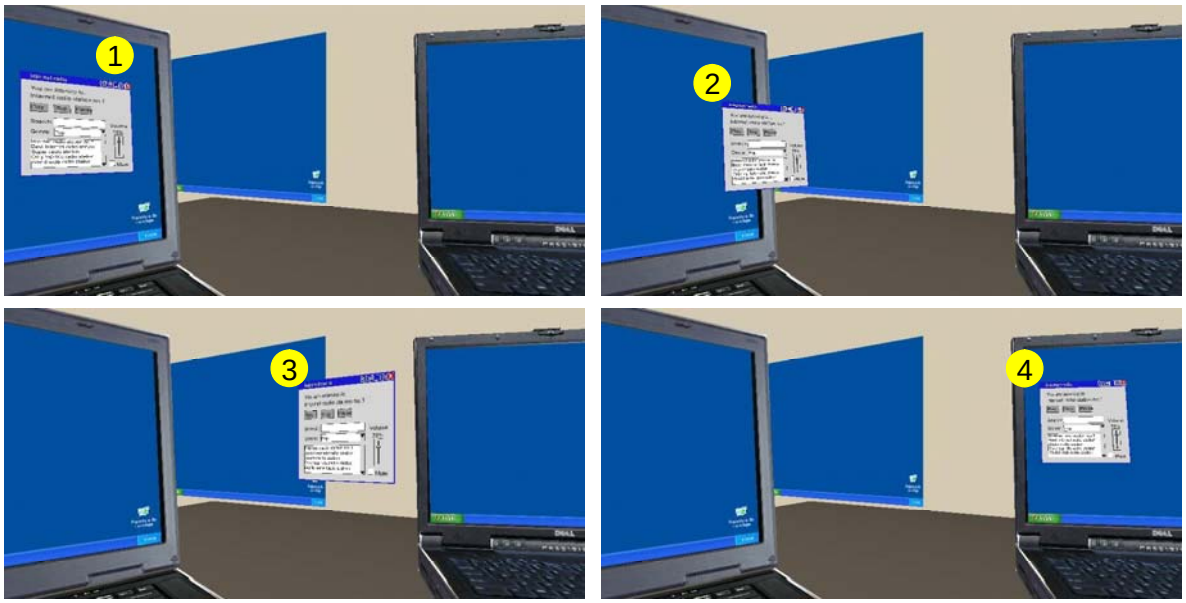

Figure 7. Sequence of screenshots for the first example, from the Acer laptop to the Dell one.

Figure 8. Sequence of images for the second example, from the Acer laptop to the Pocket PC.

CONCLUSION

In this paper, we have presented a virtual reality environment that reproduces the user's real world in which UI migration may occur by dynamically generating a virtual scene in which the user may initiate any migration (total or partial, control- or presentation-oriented) by direct manipulation. Once the manipulation occurred, the UI model with the values are transferred through the network. This virtual environment probably constitutes the best possible meta-UI for controlling the UI migration since all involved elements are graphically rendered (therefore, functional) and the migration is continuously depicted during the process. This system, MIGRIXML, is developed on top of VUIToolkit, a Virtual UI toolkit developed for this purpose exploiting USIXML models of the UI to be migrated. This appproach is superior to existing migration systems from the representation viewpoint as it supports direct manipulation of migration (windows are augmented by a <M> icon for allowing a window to be migrated), partial or total migration, contiguous or non-contiguous portions could be migrated (which is difficult to do

in a non-graphical environment), adaptation or no adaptation when the target computing platform is more constrained than the source targer. The long term goal of this research is to provide end users with a complete environment for multi-user, multi-platform, and multi-context environment fro migration, therefore allowing them to exchange and share portions of UIs depending on the way they work. Videos of MigriXML interactive sessions are accessible at http://www.usixml.org/index.php? view=page&idpage=40.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the SIMILAR network of excellence (http://www.similar.cc), the European research task force creating human-machine interfaces similar to human-human communication. J.P. Molina Massó was also supported by the Spanish CICYT project TIN2004-08000-C03-01 when visiting BCHI. See http://www.usixml.org.

REFERENCES

[1] Balme, L., Demeure, A., Barralon, N., Coutaz, J., and Calvary, G. Ethylene: a Software Architecture Reference Model for Distributed, Migrable, Plastic User Interfaces. In *Proc. of Conf. on Ambient Intelligence EUSAI'04*. Springer-Verlag, Berlin, 2004, 291–302.

[2] Bandelloni, R., Berti, S., and Paternò, F. Mixed-Initiative, Trans-modal Interface Migration. In *Proc. of MobileHCI'2004*. Springer-Verlag, Berlin, 2004, 216–227.

[3] Bandelloni, R. and Paternò, F. Migratory User Interfaces Able to Adapt to Various Interaction Platforms. *International Journal of Human Computer Studies 60* (2004), 621–639.

[4] Baudish, P., Cutrell, E., Czerwinski, M., Tandler, P., Bederson, B., and Zierlinger, A. Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-Operated Systems. In *Proc. of Interact'2003*. IOS Press, Amsterdam, 2003, 57–64.

[5] Bharat, K.A. and Cardelli, L. Migratory Applications. In *Proc. of UIST'95*. ACM Press, New York, 1995, 133–142.

[6] Biehl, J.T. and Bailey, B.P. ARIS: An Interface for Application Relocation in an Interactive Space. In *Proc. of Graphics Interface*, 2004, pp. 107-116.

[7] Coutaz, J., Lachenal, C., Calvary, G., and Thevenin, D. Software Architecture Adaptivity for Multi-surface Interaction and Plasticity. In *Proc. of IFIP WG2.7 Workshop on Software Architecture Requirements for CSCW–CSCW'2000*, ACM Press, 2000.

[8] Coutaz, J., Lachenal, C., and Dupuy-Chessa, S. Ontology for Multisurface Interaction. In *Proc. of Interact'2003*, IOS Press, Amsterdam, 2003, 447–453.

[9] de Sousa, J. and Garlan, D. AURA: An Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *Proc. of IEEE-IFIP Conf. 140 on Software Architecture* (Montreal), IEEE Computer Society Press, Los Alamitos, 2002.

[10] Grolaux, D., Van Roy, P., Vanderdonckt, J. Migratable User Interfaces: Beyond Migratory User Interfaces. In *Proc. of MOBIQUITOUS'04*. IEEE Computer Society Press, Los Alamitos, 2004, 422–430.

[11] Grudin, J. Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use. In *Proc. of CHI'2001*, ACM Press, New York, 2001, 458–465.

[12] Grundy, J.C. and Hosking, J.G. Developing Adaptable User Interfaces for Component-based Systems. *Interacting with Computers 14* (2002), 175–194.

[13] Guimbretière, F., Stone, M., and Winograd, T. Fluid Interaction with High-resolution Wall-size Displays. In *Proc. of 14th ACM Conf. on User Interface Software Technology UIST'2001* (Orlando, November 11-14, 2001). ACM Press, New York, 2001, 21–30.

[14] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Lopez, V., UsiXML: a Language Supporting Multi-Path Development of User Interfaces, Proc. of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004 (Hamburg, July 11-13, 2004). LNCS, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 200-220.

[15] Milojicic, D.S., Douglis, F., Paindaveine, Y., Wheeler, R., and Zhou, S. Process Migration. *ACM Computing Surveys* 32, 3 (2000), pp. 241–299.

[16] Molina, J.P., Vanderdonckt, J., Montero, F., and Gonzalez, P. Towards Virtualization of User Interfaces. In *Proc. of 10th ACM Int. Conf. on 3D Web Technology Web3D'2005* (Bangor, March 29-April 1, 2005), ACM Press, New York, 2005, 169–178.

[17] Montero, F., López-Jaquero, V., Vanderdonckt, J., Gonzalez, P., Lozano, M.D., and Limbourg, Q. Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML. In *Proc. of 12th Int. Work-shop on Design, Specification, and Verification of Interactive Systems DSV-IS'2005* (Newcastle upon Tyne, July 13–15, 2005), Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2005.

[18] Puerta, A.R. A Model-Based Interface Development Environment. *IEEE Software 14,4* (July/August 1997) 41–47.

[19] Rekimoto, J. and Masanori, S. Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. In *Proc. of CHI'99*. ACM Press, New York, 1999, 378–385.

[20] Rekimoto, J. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In *Proc. of UIST'97*. ACM Press, New York, 1997, 31–39.

[21] Rekimoto, J., Ayatsuka, Y., Kohno, M., and Oba, H. Proximal Interactions: A Direct Manipulation Technique for Wireless Networking. In *Proc. of Interact'2003*. IOS Press, Amsterdam, 2003, 511–518.

[22] Schäfer, K., Brauer, V. and Bruns, W. A New Approach to Human-Computer-Interaction Synchronous Modelling in Real and Virtual Spaces. In *Proc. of DIS'97*, ACM Press, New York, 1997, 335–344.

[23] Schmalstieg, D. and Hesina, G. Application Migration for Virtual Work Environments. Vienna Univ. for Virtual Work Environments, Vienna, 2001.

[24] Song, H., Chu, H., and Kurakake, S. Browser Session Preservation and Migration. In *Poster Session of WWW'2002*.

[25] Tan, D.S. and Czerwinski, M. Effects of Visual Separation and Physical Discontinuities when Distributing Information across Multiple Displays. In *Proc. of Interact'2003*. IOS Press, Amsterdam, 2003, 252–255.

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

[27] Vanderdonckt, J. A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In *Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05* (Porto, 13-17 June 2005). LNCS, Vol. 3520, Springer-Verlag, Berlin, 2005, 16–31.

[28] VRML Widgets Working Group Website. URL: http://zing.ncsl.nist.gov/~gseidman/vrml/wwg/

[29] Web3D Consortium Website. URL: http://www.web 3D.org