Migrating the User Interface between the Digital TV and Mobile Devices

Giulio Mori, Fabio Paternò, Carmen Santoro, Sandro Sansone
ISTI-CNR
{giulio.mori, fabio.paterno, carmen.santoro, sandro.sansone}@isti.cnr.it

Abstract
In this paper we present a solution able to support user interfaces that migrate among digital TVs and mobile devices, thus useful for users freely moving about in the home and outside. The proposed environment is able to understand what tasks the users are performing and dynamically generate user interfaces for the target device, with the state updated to the point at which it was left off in the previous device, and taking into account the different interaction resources of the new device. We also introduce an example application of the migration environment in the area of entertainment.

Keywords: Ubiquitous interfaces, Multi-device interfaces, Migratory interfaces, Model-based design, Digital TV.

1. Introduction

One important aspect of pervasive environments is to provide users with the possibility to freely move about and continue the interaction with the available services through a variety of interactive devices (i.e. cell phones, PDAs, desktop computers, digital television sets, intelligent watches, and so on). However, continuous task performance implies that applications be able to follow users and adapt to the changing context of use.
In this paper, we present a solution for supporting migration of application interfaces among different types of devices, including the digital TV. This opens up the possibility of creating environments supporting users freely moving in the home and outside, which can be interesting for the entertainment domain as well.
Our solution is able to detect any user interaction performed at the client level. Then, we can get the state resulting from the different user interactions and associate it to a new user interface version that is activated in the migration target device. Solutions based on maintaining the application state on the server side have been discarded because they are not able to detect several user interactions that can modify the interface state. In particular, we present how the solution proposed has been encapsulated in a service-oriented architecture and supports interfaces with different platforms (digital TV, mobile device, desktop) and modalities (graphical, vocal, and their combination). Users can conduct their regular access to the application and then ask for a migration to any device that has already been discovered by the migration environment. Migration among devices supporting different interaction modalities has been made possible thanks to the use of a logical language for user interface description that is independent of the modalities involved, and a number of associated transformations that incorporate design rules and take into account the specific aspects of the target platforms.
The motivation for migration is that the world is becoming a multi-device experience and one big potential source of frustration is that people cannot continue to perform their tasks when they move about and change their interaction device. There are many applications that can benefit from migratory interfaces. In general, services that require time to be completed (such as games, booking reservations) or services that have some rigid deadline and thus need to be completed wherever the user is (eg: online auctions).
A general reference model for user interfaces aiming to support migration, distribution and adaptation to the platform was proposed in the EU CAMELEON project. Our system, in particular, proposes a more concrete software architecture that is able to support migration of user interfaces, associated with Web applications hosted by different application servers, among automatically discovered devices, therefore resulting in a more concrete solution to the issue of migration of multimodal user interfaces.
In the paper, we first introduce our approach, then we provide more detail on the issues addressed related to including support for the digital TV platform, the device independent languages used and the reverse and forward transformation supported in order to adapt user interfaces to the device at hand. We then show an example application of the migration approach involving PDA and Digital TV (DTv) platforms, a game application. Lastly, we draw some conclusions along indications for future work.

2. The Approach

In order to support migration involving Digital TV, we have considered previous experiences in the area of migratory interfaces (Bandelloni *et al.*, 2004) and extended them in several aspects in order to be able to support migration to the digital TV. In practise, our environment supports a number of reverse and forward transformations that are able to transform existing Web desktop applications for various interaction platforms and support task continuity.

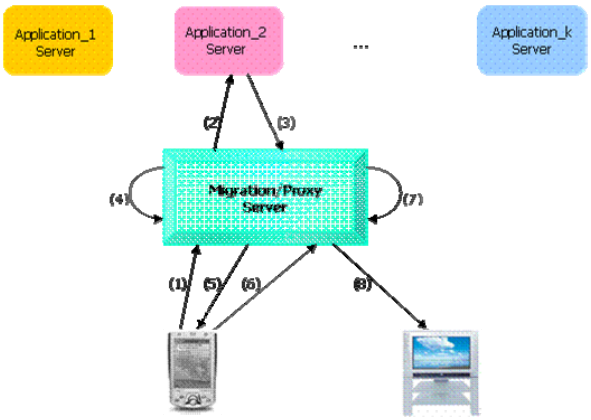

Figure 1: The architecture of the migration approach

As Figure 1 shows, the architecture is based on a proxy/migration server. Users who subscribe to the migration service can then access Web applications through such server (see arrows (1,2,3) in Figure 1). If the interaction platform used is different from the desktop, the server transforms it (arrow (4)) by building the corresponding abstract description and using it as a starting point for creating the implementation adapted for the device accessing it, which is the delivered (arrow (5)). In addition to interface adaptation, the environment also supports task continuity. To this end, when a request of migration to another device is triggered the environment detects the state of the application modified by the user input (elements selected, data entered, …) and identifies the last element accessed in the source device. The information about the user interaction state is collected into a string formatted following a XML-based syntax and submitted to the server together with the IP of the target device in the migration (arrow (6)). Then, a version of the interface for the target device is generated, the state detected in the source device is associated with the target device version so that the selections performed and the data entered are not lost (arrow (7)). Lastly, the user interface of the target device is activated at the point supporting the last basic task performed in the initial device (arrow (8)).

Migration can be triggered either by the user (through either a graphical interface or scanning RFID tags associated to the target device) or by the environment when some specific event (such as battery or connectivity very low) is detected or in a mixed solution in which the environment suggests possible migrations based on the devices available at a short distance and the user decides whether to accept them or not. In the process of creating an interface version suitable for a platform different from the desktop we use a semantic redesign module, which transforms the logical description of the desktop version into the logical description for the new platform.

In order to support transformations from an abstract platform-independent language to an implementation (for example, for the digital TV), we use an intermediate specification, the concrete description, which is a platform-dependent refinement of the abstract interface. These two types of user interface logical languages consider two types of elements: basic elements (output-only and interaction), and composition elements, aiming to indicate how to combine together two or more elements.

Our approach also supports interoperability between various implementation languages. Thus, a Web application can be transformed into a Java application for the Digital TV. In this case the generation of the user interface implementation involves the generation of a file in a Java version for digital TVs representing an Xlet, which is an application that is immediately compiled and can be interpreted and executed by the interactive digital TV decoders.

3. User Interface Logical Languages

In the research community in model-based design of user interfaces there is a general consensus on what the useful logical descriptions are (Bouillon and Vanderdonckt, 2002; Paternò, 1999):

- The task and object level, which reflects the user view of the interactive system in terms of logical activities and objects manipulated to accomplish them;
- The abstract user interface, which provides a modality independent description of the user interface;
- The concrete user interface, which provides a modality dependent but implementation language independent description of the user interface;

- The final implementation, in an implementation language for user interfaces.

Thus, for example we can consider the task "switch the light on": this implies the need for a selection object at the abstract level, which indicates nothing regarding the platform and modality in which the selection will be performed (it could be through a switch or a vocal command or a graphical interaction). When we move to the concrete description, we have to assume a specific platform, for example the graphical PDA, and indicate a specific modality-dependent interaction technique to support the interaction in question (for example, selection could be through a radio-button or a drop-down menu), but nothing is indicated in terms of a specific implementation language. When we choose an implementation language we are ready to make the last transformation from the concrete description into the syntax of a specific user interface implementation language.

In addition, at the abstract level we also describe how to compose basic elements through some operators which have been defined taking into account the type of communication effects that designers aim to achieve when they create a presentation (Mullet and Sano, 1995). They are: Grouping, indicating a set of interface elements logically connected to each other; Relation, highlighting a one-to-many relation among some elements, one element has some effects on a set of elements; Ordering, some kind of ordering among a set of elements can be highlighted; Hierarchy, different levels of importance can be defined among a set of elements. Therefore, an abstract user interface is composed of a number of presentations and connections among them. While each presentation defines a set of interaction techniques perceivable by the user at a given time, the connections define the dynamic behaviour of the user interface, by indicating what interactions trigger a change of presentation and what the next presentation is.

The concrete level is a refinement of the abstract interface: depending on the type of platform considered there are different ways to render the various interactors and composition operators of the abstract user interface. For example, in the case of a graphical desktop platform a navigator can be implemented either through a textlink, or an imagelink or a simple button, a single choice object can be implemented using either a radio button or a list box or a drop-down list and the same holds for the operators. It is worth noting that all the logical languages in our approach, for any abstraction level, are defined in terms of XML-based languages in order to allow their exporting/importing in different tools.

4. Reverse Engineering

The main purpose of the reverse engineering part is to capture the logical design (in terms of tasks, types of interaction elements, and ways to structure the user interface) underlying the interface implementation and then use it as a key element to drive the generation of the interface for the target device. Some work in this area has been carried out previously (Bouillon and Vanderdonckt, 2002; Paganelli and Paternò, 2003). The reverse engineering module of our migration environment takes a Web page and recursively analyses its DOM tree starting with the body element and going in depth, and then provide any of the three levels of logical descriptions supported (concrete, abstract and task level), as needed.

5. Semantic Redesign

In order to automatically redesign a desktop presentation for a mobile device we need to consider semantic information and the limits of the available resources. To this end, our algorithm tries to maintain interactors that are composed through some operator at the conceptual level in the same page using implementations that take into account the interaction resources available but preserve the communication goals of the designer.

In the transformation process we also take into account the cost in terms of interaction resources of the elements considered. We have defined for each mobile device class identified (large, medium or small) a maximum acceptable overall cost in terms of the interaction resources utilizable in a single presentation. Examples of elements that determine the cost of interactors are the font size (in pixels) and number of characters in a text, and image size (in pixels), if present. Example of the costs associated with composition operators is the minimum additional space (in pixels) needed to contain all its interactors in a readable layout.

6. User Interfaces for the Digital TV

Usability is a fundamental consideration for interactive services delivered through technologies such as the digital TV, which can be accessed by people with any background, often with limited computer skills. Since this is a new area there is still a lack of standards, but some guidelines have started to emerge. see for example (Fondazione Ugo Bordoni) based on a number of user tests. One

of the criteria we adopted was deciding to use a specific kind of font –Tiresias– due to its highly suitability to be visualised on TV displays. In addition, in order to guarantee the best readability of the text, quite high font dimensions were selected (range between 24-36pt), avoiding the smaller ones, which do not guarantee sufficient readability.
Also, in digital TVs the availability of an application should be clearly indicated and the access should be simple, fast and clear. In particular, it is important to indicate the availability of the application; set a key to access the interactive application, to exit, and to support zooming; indicate when the application is closing or loading state. The application structure should be clear and intuitive. In particular, it is crucial to separate content and commands; locate the most important elements in the top-left area; locate the audio/video flow in the top-right area; dedicate a portion of the screen to the commands.

Moreover, bearing in mind that there is no mouse to support navigation but just TV remote controls, designers should structure the content hierarchically; assign a consistent and unique function to the keys; keep the order and representation of the control keys consistent on the screen; allow users to access menus through directional, coloured or numerical keys associated with the functionalities. Furthermore, the user should receive feedback of the activated process before it terminates. Thus, the interactive applications should promptly communicate through messages what is happening; define graphical or textual help functions.

In addition, appropriate colours should be selected, taking into account that the DVB-MHP standard supports a limited number of colours (palette with 188 colours). Also, flickering should be reduced at maximum by avoiding drawing lines with a size of 1 or 2 pixels, and/or by strongly contrasting colours in contiguous areas. In addition, some applications require the user to enter some data: the lack of a keyboard requires using the TV control for this purpose, which is generally not so efficient from the user's point of view. Therefore, the number of fields to be filled in should be limited on this platform.
The nature of the TV is different from many other communication media providing textual information. It is thus important to pay attention in order to avoid making the reading process tedious. To this end the interactive applications should present the text in such a way that the important information is at the beginning; use simple verbal forms; use titles, subtitles and abstracts to structure the text; divide text on multiple pages or insert a scrolling bar; use the colours with semantic purposes.
As for the generation of the user interface implementation, it involves the generation of a file in a Java version for digital TVs representing a Xlet, which in real settings (which usually means if a digital phone line is available) is supposed to be downloaded on the Set-Top-Box; otherwise an emulator can be used to execute the interactive Xlet. For our examples, we used the XletView emulator (http://xletview.sourceforge.net), we will show an example of its use in next sections.

6.1 A Toolkit for Java Digital TV Interfaces

As we already said, Xlets are similar to Java applets, However, there is a big difference regarding the user interface part: Xlets do not include the AWT package, which contains several widgets for Java programs, such as radio button, check box, button, etc.

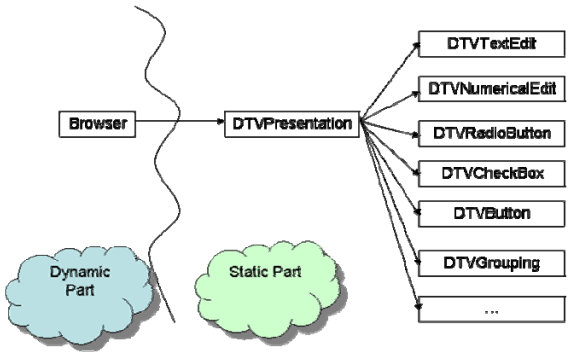

Figure 2: The structure of the toolkit for digital TV interfaces

Thus, we have also developed a toolkit that provides such techniques and simplify the target for the user interface generation in the digital TV platform. In the library there is a class (DTVPresentation) that provides the methods to include the widgets in the application without having to specify all the details every time. Then, there is a class for each interactor implementing its appearance and behaviour.

The semantics of the interactors' implementation is managed through specific event handlers that allow them to update the state and then their appearance accordingly. Thus, for example, in the case of a radio button the corresponding event handler keeps track of the currently selected element (using state variables) and will take care to update the graphical view of the interaction object.

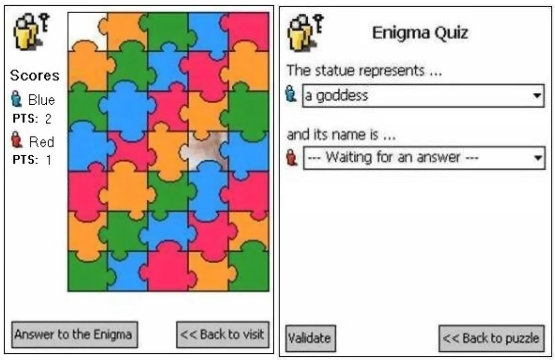

Figure 3: The user interface visualised on the PDA

7. An Example Application

The example application considered in this case is a game application in the area of e-learning associated with a museum.
Users can interact with their PDAs to access to a number of games that are associated to the artworks in the museum while on the move. They can start to play the game: access the service, identify themselves and start to access the different games. When they arrive at home, they may want to continue to play the games exploiting its larger screen. For example, it may occur that while on the move they started to solve some questions associated to an artwork, but without finishing it. When they arrive at home, they may want to continue to play the game with more powerful device. Therefore, they require to migrate the user interface onto the large screen of the digital TV available without having to re-enter the previously specified options.
After the interface migration, they can still find the options they specified before (see Figure 4) and continue to play the game by solving the remaining questions.
In Figure 3 it is displayed the user interface of an enigma as it is visualised onto the PDA. Due to the small capability of the PDA, the user interface is presented into two separate screens: one dedicated to the visualisation of the image associated to the enigma, the other one showing the questions to be answered. In Figure 4 it is visualised the user interface after the user migration request onto the digital TV. As you can see the choice the user has already carried out on the PDA (what the statue represents) is visualised as soon as the user interface is loaded on the user interface of the digital TV. Then, the user can continue his/her interaction by selecting the name of the represented artwork. As you can see, thanks to the larger screen of the digital TV, the user interface shows in a single presentation both the image of the enigma and the associated questions.

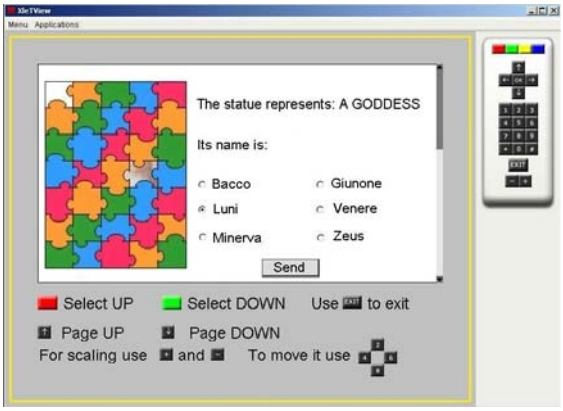

Figure 4: The user interface after the migration on the digital TV

8. Conclusions and Future Work

We have presented an environment able to support migration of user interfaces for various platforms including digital TV, mobile devices and desktop systems.
This solution can be useful for supporting various types of interactive services able to follow and support the user moving through different devices in the home and outside for common tasks such as games, shopping, bids for auction on line, making reservations.

References

Bandelloni, R., Berti, S., Paternò, F. (2004). Mixed-Initiative, Trans-Modal Interface Migration. In Proceedings of *Mobile HCI'04*, (pp 216-227)., LNCS 3160. Springer-Verlag, 216-227.
Bouillon, L., and Vanderdonckt, J. (2002). Retargeting Web Pages to other Computing Platforms. *IEEE 9th Working Conference on Reverse Engineering WCRE'2002* (pp. 339-348). IEEE Computer Society Press.
Fondazione Ugo Bordoni – Recommendations for Interactives services interfaces in the Digital TV. From http://www.fub.it/ambientedigitale/repository/Generale/Raccomandazioni.pdf
Mullet, K. & Sano, D. (1995), *Designing Visual Interfaces*. Prentice Hall.
Paganelli, L., and Paternò, F. (2003). A Tool for Creating Design Models from Web Site Code. *International Journal of Software Engineering and Knowledge Engineering*, World Scientific Publishing *13*(2), 169-189.
Paternò, F. (1999). *Model-based Design and Evaluation of Interactive Applications*. Springer Verlag, ISBN 1-85233-155-0.