User Interface Migration between Mobile Devices and Digital TV

Fabio Paternò[1], Carmen Santoro[1], and Antonio Scorcia[1]

[1] ISTI-CNR, Via G. Moruzzi, 1
56124 Pisa, Italy
{Fabio.Paterno, Carmen.Santoro, Antonio.Scorcia}@isti.cnr.it

Abstract. In this paper we present a demonstration of the Migrantes environment for supporting user interface migration through different devices, including mobile ones and digital TV. The goal of the system is to furnish user interfaces that are able to migrate across different devices, in such a way as to support task continuity for the mobile user. This is obtained through a number of transformations that exploit logical descriptions of the user interfaces to be handled. The migration environment supports the automatic discovery of client devices and its architecture is based on the composition of a number of software services required to perform a migration request.

Keywords: User Interface Migration, Adaptation to the Interaction Platform, Ubiquitous Environments.

1 Introduction

One important aspect of pervasive environments is the possibility for users to freely move about and continue interacting with the services available through a variety of interactive devices (i.e. cell phones, PDAs, desktop computers, digital television sets, intelligent watches, and so on). In this area, one important goal is to support continuous task performance, which implies that applications be able to follow users and adapt to the changing context of users and the environment itself. In practice, it is sufficient that only the part of an application that is interacting with the user migrates to different devices.

In recent years, research on issues related to user interfaces in ubiquitous environments has started (see for example [1] [2] [3]). For instance, a discussion of some high-level requirements for software architectures in multi-device environments is proposed in [1], although it is done without presenting a software architecture and implementation solution for these issues. In our work, we propose a specific architectural solution, based on a migration/proxy server, able to support migration of user interfaces associated with applications hosted by different content servers.

More in detail, in this demo, we show a solution for supporting migration of application interfaces among different types of devices. Such 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. In particular, the solution proposed has been encapsulated in a service-oriented architecture and supports user interfaces with different platforms (fixed and mobile) and modalities (graphical, vocal, and their combination). The new solution also includes a discovery module, which is able to detect the devices that are present in the environment and collect information on their features. Users can therefore conduct their regular access to the Web application and then ask for a migration to any device that has already been discovered by the migration server. The discovery module also monitors the state of the discovered devices, automatically collecting their state-change information in order to understand if there is any need for a server-initiated migration. Moreover, we show how the approach is able to support migration across devices that support various implementation languages. This has been made possible thanks to the use of a logical language for user interface descriptions at different abstraction levels [4], which is independent of the implementation languages involved, and a number of transformations that incorporate design rules and take into account the specific aspects of the target platforms.

In the paper we first describe a scenario supported by our demo, next we briefly describe the underlying architecture, and lastly we discuss an example session showing the corresponding user interfaces provided to the users.

2 A Scenario Supported by the Demo

The demo regards a user returning home from work, who starts to prepare the shopping list through a mobile device (while s/he is on the bus or train) and then when s/he gets at home, s/he may look at what is actually available and realise that some items are still missing. Then, s/he completes the list by interacting with the digital TV with large screen while sitting comfortably on the couch.

Thus, using the PDA, the users can access the page dedicated to the products and specify the category they are interested in (for example “meat”). Depending on the selected category, the application allows a further refinement of the selection. In our scenario, the users are allowed to select which kind of meat they want to buy by means of choosing among beef, poultry and pork. Then, a number of options are visualised together with the associated amounts, and the user can start to select what s/he wants to buy. When the user enters home, the smart environment suggests the user the possibility to migrate the user interface to other devices which have been recognised as available in the new environment, since the agent-based architecture has recognised a situation where more comfortable interactions might take place (e.g.: the user could interact with the desktop PC which has a larger screen, or s/he can interact with the TV while comfortably sitting on the couch). Then, if the user decides to migrate the user interface to the digital TV, s/he can continue editing the shopping list through a larger screen without having to save their selections from the PDA and login again the application from the new device. After the interface migration, the user can find the items that were specified before, through the PDA (e.g. the request for three beef steaks, which was specified using the handheld device) and edit them or

add new ones until lastly they send the request. The text can be entered by selecting a specific button on the TV controller, which activates a virtual keyboard on the screen.

3 The Migration between Mobile Device and Digital TV

The main characteristics of migration are: device change, adaptation, and continuity. The basic idea is that people would like to freely move and still be able to continue to perform their tasks and thus the interactive part of an applications should be able to follow them and adapting to the changing context of use.
Our migration environment is based on a service-oriented architecture involving multiple clients and servers: the architecture is aimed at providing interoperability between the different services, which can be also combined for delivering composite services, as it happens in the migration support. We assume that the desktop version of the considered applications already exists in the application servers. In addition, we have a migration platform, which is composed of a proxy service and a number of specific services and can be hosted by either the same or different systems.
The main services that have been identified to compose the migration platform:

- the *Discovery Manager*, which includes the functionalities for discovering the available devices and update the device list accordingly;
- the *Migration Manager/Proxy* is the core of the system: it handles the communication with the other modules, also including proxy functionalities.
- the *Reverse Engineering*, is in charge of reversing the desktop implementation into a logical user interface description;
- the *Semantic Redesign* module, which transforms the logical description of the user interface designed for the source platform into a logical description of the user interface for the target migration platform;
- the *State Mapper*, which updates the final user interface with the values of the current state, which have been saved at the time the request of migration occurred;
- The *UIGenerator*, which reifies the logical concrete description into an implementation language for the target platform.

The process starts with the source and target devices notifying their presence to the Discovery Manager, which is in charge of discovering the available devices and updating the list of devices accordingly, also showing their characteristics. Indeed, in order to allow for a good choice of the target device, information about the devices that are automatically discovered in the environment is displayed and saved. Such information mainly concerns device identification and interaction capabilities and, on the one hand, it enables users to choose a target migration device with more accurate and coherent information on the available targets and, on the other hand, it enables the system to suggest or automatically trigger migrations when the conditions for one arise. Thus, both the system and the user have the possibility to trigger the migration process, depending on the surrounding context conditions.
Users have two different ways of issuing migration requests. The first one is to graphically select the desired target device in their migration client. Users only have the possibility of choosing those devices that they are allowed to use and are currently

available for migration. The second possibility for issuing migration requests occurs when the user is interacting with the system through a mobile device equipped with an RFID reader. In this case, users could move their device near a tagged migration target and keep it close from it for a number of seconds in order to trigger a migration to that device. In this case, in addition to a spatial threshold used to indicate when the user is sufficiently close to trigger a migration, a time threshold has been defined in order to avoid accidental migration, for example when the user is just passing by a tagged device. This second choice offers users a chance to naturally interact with the system, requesting a migration just by moving their personal device close to the desired migration target, in an easy manner. Migration can also be initiated by the system, skipping explicit user intervention in critical situations when the user session could accidentally be interrupted by external factors. Alternatively, the server can provide users with migration suggestions to improve the overall user experience.
The migration clients are supposed to access the various applications through the proxy available within the Migration Manager. Indeed this module works as a proxy since it is in charge of intercepting the clients' request of accessing a page, retrieving such a page from Internet and saving it locally together with the referred entities (images, CSS files, etc.). Afterwards, the Migration Manager receives from the source device the request for migration (which specifies the source device, the target device, and the page that has to be migrated), and it triggers the sequence of actions needed for fulfilling such a request. It is worth noting that the application that triggers the migration – the so-called 'Migration Client'- can be contained in an application which is separated from the web browser. For instance, in the current implementation, the migration request is activated through a separate C# program which allows the user to select the devices available for migration (see Figure 1, Left).

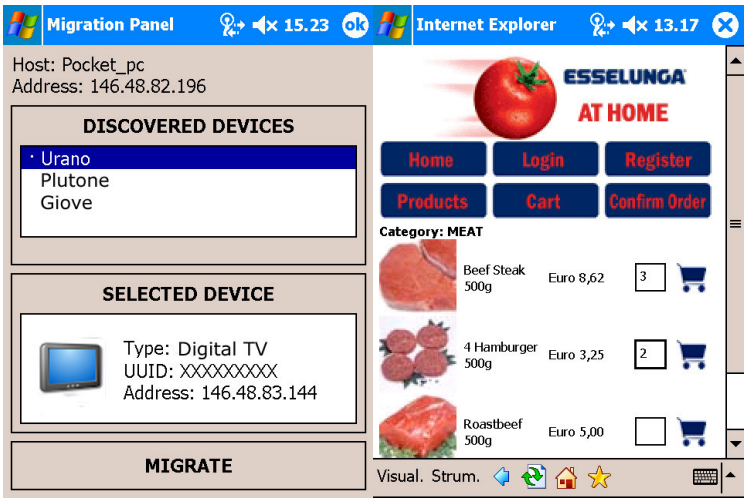

Figure 1: Left, The Migration Client Interface, Right, the Application User Interface.

Once the Proxy receives the Web page from the concerned Application Server, the Proxy modifies it by including JavaScript functions that are aimed at collecting information about the state of the migrating page, and afterwards it sends to the Web

browser of the source device (PDA). The JavaScript functions that are automatically inserted by the proxy server are in charge of collecting the information that describes the state of the migrating page by accessing its DOM. The information is collected into a string formatted following a XML-based syntax and submitted to the server together with the IP of the target device. This information is sent to the server through an AJAX script. The reason for this is that only the application running on the client device can access the DOM and the AJAX callback can transmit the data without requiring any additional explicit action from the user.

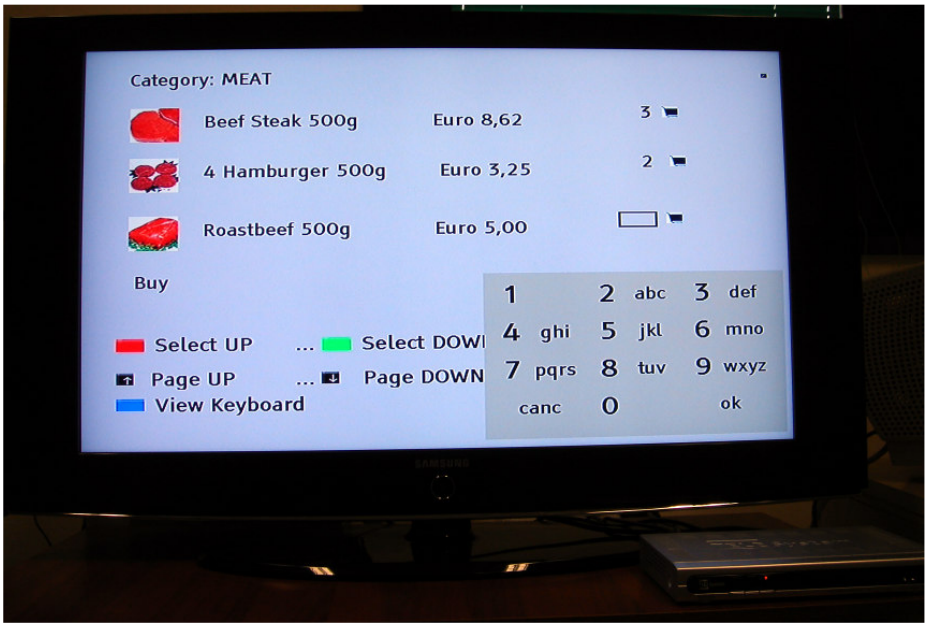

Figure 2: The Application User Interface Migrated into the Digital TV.

The Migration Server, after receiving the request of migration by the source device, interrogates the Migration Client of the target device asking about its availability/willingness for accepting a migrating UI: if the migration is accepted, 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. Then, the Migration Manager gets information about the source device and, depending on such information it builds the corresponding logical descriptions, at a different abstraction level, by invoking the Reverse Engineering service of our system. At this point, the Migration Manager asks the Discovery Manager information about the target device in order to understand for which platform the redesign process has to be carried out. Indeed, the result of the reverse engineering process, together with information about source and target platforms is used as input for the Semantic Redesign service, in order to perform a redesign of the user interface for the target platform. This part of the migration environment transforms the logical description of the desktop version into the logical description for the new platform.

This solution allows the environment to exploit semantic information contained in the logical description and obtain more meaningful results than transformations based only on the analysis of the specific implementation language used for the final UI. Once the application presentation to activate on the target device is identified, the Migration Manager asks the State Mapper to adapt the state of the concrete user interface with the values that have been saved previously. Then, once the concrete user interface adapted with the new values has been obtained, the reification of such a logical description into the final user interface for the target platform is performed by the UIGenerator module and lastly, the resulting page is sent to the browser of the target device in order to be loaded and rendered. Figure 2 shows the UI migrated into the Digital TV. It is possible to see that the values entered in the source device (see Figure 1) have been preserved in the user interface generated for the target device, and the users can continue from the point they left off. As for the implementation for the digital TV, it involves the generation of a file in a Java version for digital TVs representing a Xlet, which is downloaded on the Set-Top-Box. In our demo we use a Set Top Box Telesystem TS7.2 DT, which supports Multimedia Home Platform (MHP 1.0.2), an open middleware system standard for interactive digital television, enabling the execution of interactive, Java-based applications on a TV-set. It is worth pointing out that in this example we considered migration from PDA and the Digital TV, but the approach can be extended for any platform, providing that exists a Web desktop application version and the opportune software modules (migration client, UI generator,..) are provided taking into account the characteristics of the considered devices (available interaction resources, implementation languages supported, …).

4 Conclusions

In this paper we describe a system for enabling user interface migration through different devices: UI logical descriptions (with associated transformations) have been exploited for supporting the migration mechanisms, together with various technologies (e.g. AJAX scripts) for saving the current state of the user interface. Ongoing work is dedicated to further enrich the data associated with the current state of the user interface in order to support continuity in a wider set of user's interactions.

References

1. Balme, L. Demeure, A., Barralon, N., Coutaz, J., Calvary, G.: CAMELEON-RT: a Software Architecture Reference Model for Distributed, Migratable and Plastic User Interfaces. In: Proceedings EUSAI '04, LNCS 3295, Springer-Verlag, 2004, 291-302.
2. Bandelloni R., Mori G., Paternò F., Dynamic Generation of Migratory Interfaces, Proceedings Mobile HCI 2005, ACM Press, pp.83-90, Salzburg, September 2005.
3. Luyten, K., Coninx, K. Distributed User Interface Elements to support Smart Interaction Spaces. In: IEEE Symposium on multimedia. Irvine, USA, December 12-14, (2005).
4. Mori G., Paternò F., Santoro C.: Design and Development of Multi-device User Interfaces through Multiple Logical Descriptions. In: IEEE Transactions on Software Engineering August (2004), Vol 30, No 8, IEEE Press, 507-520.