Migratory User Interfaces Able to Adapt to Various Interaction Platforms

Renata Bandelloni, Fabio Paternò

ISTI-CNR,Via G.Moruzzi 1, 56100 Pisa, Italy.
{r.bandelloni , f.paterno}@isti.cnr.it

Abstract. The goal of this work is the design of an environment for supporting run time migration of Web applications among different platforms. This allows users interacting with a Web application to change device and continue their interaction from the same point. The migration takes into account the runtime state of the interactive application and the different features of the devices involved. We consider Web applications developed through a multiple-level approach using: the definition of the tasks to support, the abstract description of the user interface, and the actual code. The runtime migration engine exploits information regarding the application runtime state and higher-level information on the available target platforms. Runtime application data are used to achieve interaction continuity and preserve usability, while information on the different platforms is considered to adapt the application's appearance and behaviour to the specific device. The paper also discusses a sample application in order to provide concrete examples of the results that can be achieved through our approach.

Keywords: Migratory interfaces, Run-time architectural support, Multi-platform applications

1 Introduction

Today we are witnessing a proliferation of new mobile devices offering the possibility of accessing the Internet through several types of devices. Everyday life is becoming a multiplatform environment where people are surrounded by devices through which they can get connected in different ways.

Many efforts are currently aimed at allowing users to interact through multiple devices. For example, a user browsing the net with a PDA touch screen or a mobile phone keypad would be more comfortable using the mouse and keyboard of a stationary PC. Conversely, a user may be entering private data through a stationary PC and wish for the greater privacy afforded by a PDA. In both cases, a multi-platform migration service would be necessary, by which the user could interact with applications while changing devices and still maintaining interaction continuity. There are two main issues concerning this kind of service. Firstly, the diversity in features of the

platforms involved in migration, like different screen size, interaction facilities, processing power and energy supply, can make an application developed for a desktop, unsuitable for a PDA and vice versa. Thus, an application cannot migrate *as is* from one device to another, and must be adapted at runtime, taking into account the diversity of the devices involved (see for example, Kaikkonen and Roto, 2003). The second issue concerns interaction continuity. Users who want the application to migrate, do not want to have to restart the application on the new device; they want to continue their interaction from the same point where they left off, without having to re-enter the same data and go through the same long series of links to get to the page they were visiting on the previous device (these issue are introduced in Song et al. 2003). Two main kinds of information are relevant in performing migration: static information referring to the features of the devices, and runtime information that refers to the state of the migrating application, which can be summarised by the history of user interactions with the application, including visited pages, submitted data and results of previous data processing. There are several techniques for migrating user interfaces to different devices, in particular to small screens, and most of them rely on size reduction and data summarisation (MacKey, 2003), with the risk of making the application unusable because objects on the page are difficult to recognise. Herein we focus on interaction continuity and device adaptation at runtime that takes into account usability principles. We consider different platform-specific versions of the same application, starting with a general task model (Paternò and Santoro, 2003) from which we generate the actual application by means of the TERESA tool (Moriet al. 2003). We take into account the migration of TERESA-generated applications. This tool produces a description of the pages and the interactions that they support at different abstraction levels. Runtime data on the state of the application for which migration is required will be collected locally from the platform requesting migration. This information is transmitted to the server in order to recreate the corresponding state in the application for the target device.

We have introduced a basic framework to support a device-aware runtime migration for Web applications in (Bandelloni and Paternò, 2003*)*. In this paper we provide a more systematic solution on how to use information created following a model-based design approach in order to have a migration service offering user interaction continuity, platform awareness and runtime adaptability. In particular, we first discuss related work and introduce the TERESA approach to design and develop multi-platform user interfaces. Then, we move on to discuss the possible cases that should be considered when migrating interfaces between different interaction platforms and present our solution to these issues. The final part of the paper is dedicated to discussing an example application of the migration support.

2 Related Work

Application migration is a field undertaken by several research projects, which address it from different points of view. The growing interest in migration in the ubiqui-

tous computing field aims to make an application follow a user from one device to another.

An interesting paper (Krishna and Cardelli, 1995*)* provides an overview of the main matters concerning application migration. They present a programming model and an implementation of a tool for developing migratory applications, placing no restriction on the kind of application that can be built. The approach is agent-based: agents carrying pieces of code and the state of the migratory application are sent from a host to another where a server allows the agent to rebuild the migrating application. This approach can also be used to build migratory interfaces into the agent, including interface code and state. Such an approach is not suitable for our goals, our intent is to build a migration service supporting several kind of platforms, from powerful stationary PCs to PDAs to cell phones. Most of them are mobile platforms, having to cope with power consumption, low storage and elaboration capabilities. The processing load involved with using agents that migrate to a platform hosting an agent server, where the application is rebuilt at runtime, would be too heavy for most of the platforms we have targeted in our migration service.

In (Coutaz et al., 2003*)* a framework to support migratory applications among different surfaces is presented. A complex architecture is used to allow an application both totally and partially migrate over different platforms adapting itself to the target host feature. The continuity of the user interaction and the adaptability of the application to the host platform are subjects we are addressing too. Our approach differs from this one in being specific for Web applications, addressing their specific features and taking into account their usability requirements. We are not addressing the splitting of the application in many parts. This allows us to build a less complex and computationally lighter framework for supporting migration. In (Coutaz at al, 2003) there is no indication about the efficiency of the framework in terms of time required to perform migration. It is a fact that users are disappointed to have to wait long time for operations to be performed. We want to offer a migration service that can be perceived as instantaneous, by developing a less general, yet lighter framework.

Kaikkonen and Roto, 2003 prove how the different features of the interaction platforms can influence Web applications usability. Different screen size, interaction facilities, processing power and energy supply, can make a Web application developed for a desktop, unsuitable for a PDA and vice versa. Thus, an application cannot migrate as it is from one device to another, and must be adapted at runtime, taking into account the diversity of the involved platforms.

An overview of the techniques used in adapting user interfaces to different devices, in particular to small screens rely on size reduction and data summarisation as it is shown in (MacKey, 2003*)*. This approach raises the risk to make the application unusable because objects on the page become difficult to recognise. We want to overcome this kind of problem adapting interfaces to different platforms, taking into account the effects on the usability of the application.

In (Song et al. 2002*)* aspects concerning interaction continuity are shown. The runtime state of the migrating application must be preserved on the target device, to allow the user to continue the interaction from the exact point where it left. Interaction continuity is one key factor in our migration service, what is new in our approach is that when target and source platforms involved in migration are different, the run-

time state will not be sent to the target platform as it is, data are transformed and adapted to the features of the target platform.

3 Model-base Development of Platform-Aware User Interfaces

TERESA is a transformation-based environment (http://giove.cnuce.cnr.it/teresa.html) designed and developed at the HCI Group of I.S.T.I.-C.N.R. It is intended to provide a complete semi-automatic environment supporting a number of transformations useful for designers to build and analyse their design at different abstraction levels and consequently generate the concrete user interface for a specific type of platform.

The abstraction levels considered are: the task model, where the logical activities to support are identified; the abstract user interface, in this case the interaction objects (but classified in terms of their semantics, still independent from the actual implementation) are considered, and the concrete user interface (the actual corresponding code). The main transformations supported in TERESA are:

- *Presentation Task Sets and Transitions Generation.* From the specification of a ConcurTaskTrees (Paternò, 1999) task model it is possible to obtain the set of tasks, which are enabled over the same period of time according to the constraints indicated in the model. Such sets, depending on the designer's application of a number of heuristics supported by the tool, might be grouped together into a number of *Presentation Task Sets (PTSs)* and related *Transitions* among the various PTSs.
- *From Task Model-related Information to the Abstract User Interface.* Both the task model specification and PTSs are the input for the transformation generating the associated abstract user interface, which will be described in terms of both its static structure (the "presentation" part, which is the set of interaction techniques perceivable by the user at a given time) and dynamic behaviour (the "dialogue" part, which indicates what interactions trigger a change of presentation and what the next presentation is). The structure of the presentation is defined by elementary interactors characterised in terms of the task they support, and their composition operators. Such operators are classified according to the communication goals to achieve: a) *Grouping*: indicates a set of interface elements logically connected to each other; b) *Relation*: highlights a one-to-many relation among some elements, one element has some effects on a set of elements; c) *Ordering*: some kind of ordering among a set of elements can be highlighted; d) *Hierarchy*: different levels of importance can be defined among a set of elements. There is also the option to automatically generate the abstract UI for the target platform (instead of going through the two transformations mentioned before), starting with the currently loaded (single-platform) task model, and using a number of default configuration settings related to the user interface generation.
- *From the Abstract User Interface to the User Interface for the specific platform.* This transformation starts with the abstract user interface, it is possible to move into the related concrete user interface for the specific interaction platform selected. A number of parameters related to the customisation of the concrete user interface are

made available to the designer in order to obtain the concrete user interface. Lastly, the tool generates the code according to the type of platform selected from the concrete user interface description. Currently it generates code in HTML, XHTML Mobile Profile and VoiceXML.

4 Runtime Migration Cases

Different types of runtime migration can be identified, along with different levels of complexity for each one of them:

- *Total Migration*: the client application migrates totally from a device to the other.
- *Control Migration*: the client application is divided into two parts, one for user interaction (control part) and one for information presentation (presentation part). The control part remains on one device, while the presentation one migrates to the other device, or vice versa (an example is discussed in Nichols et al. 2002).
- *Mixed Migration*: the client application is split into several parts, concerning both control and presentation and different parts are distributed over two or more devices.

In our work, we focus on *Total Migration*, with the goal to support a runtime migration that takes into account the differences between the two platforms involved. TERESA structures user interfaces into presentations and transitions among them. When we migrate a presentation from a platform to another one the runtime support first identifies the closest presentation in the target platform. The difference between presentations in different platforms is calculated in terms of the number of logical tasks supported. A task can be supported through different interaction techniques. However, the logical meaning of the task is still the same. Taking into account interactive applications developed by means of TERESA we can identify the following situations concerning the runtime migration of a presentation between two platforms (see laso Figure 1):

- The migrating presentation corresponds to one target presentation. In this case the target page to be loaded on target device can be immediately identified through a one by one mapping. The two presentations can differ in the number of supported tasks, in particular the target presentation can support:
    - Same tasks. Even if the same tasks are supported by the two presentations, the runtime state data may need to be modified too, because tasks can be implemented by means of different kind of concrete objects.
    - Lower number of tasks. As in previous case, there can be a mapping of data concerning corresponding tasks. Data concerning tasks not supported by the target presentation are ignored.
    - Higher number of tasks. Source tasks are treated as in previous cases, while target tasks for which information cannot be retrieved

from source runtime state data, are untouched and loaded with their default values.

- The migrating presentation corresponds to multiple target presentations, among which the tasks set of the source presentation are spread. In this case the target presentation is identified by computing the one having the highest number of tasks in common with the source one. In case that more than one target presentation has the same similarity degree according to this criterion, it is chosen the one supporting the task associated with the last concrete object through which the user interacted with the application on the source side.
- Multiple presentations in the source platform correspond to one presentation in the target platform. In this case the migrating task set will correspond to part of the task set supported by one of the target presentations.

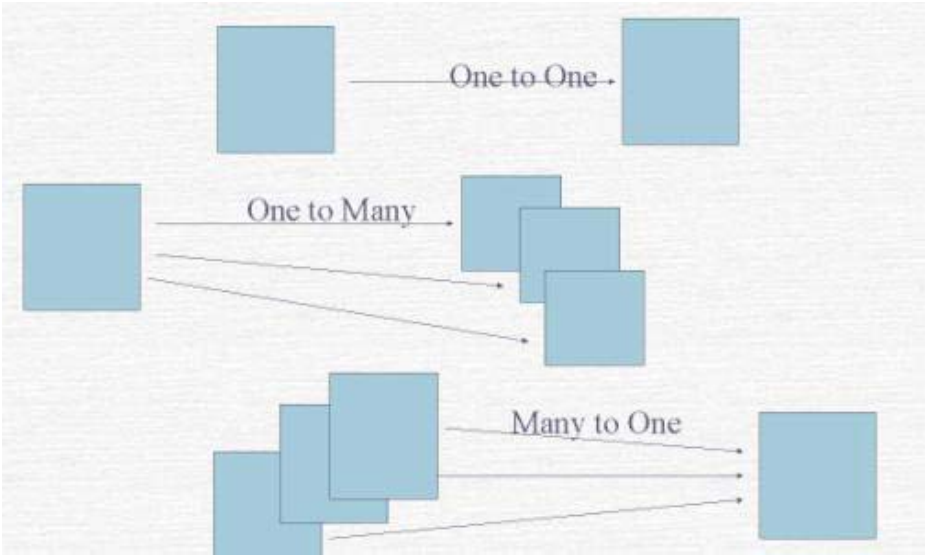

Figure 1: The possible relationships between source and target presentations.

5 Our Migration Solution

Our migration service is designed to meet two main requirements, device awareness and interaction continuity. To keep interaction continuity it is necessary to collect information concerning the runtime state of the migrating application, to activate the application on the target device, from the same point where it left. Since migration will involve different types of platforms, runtime state will not be migrated as it is. Data concerning the platform type will be used to adapt the runtime data collected on the source platform to the target one. Hence we have implemented a mapping algorithm that make use of both runtime state and involved platform data, to load on the

target the application version fitting its features, and keeping state consistency with the state the application had at migration time.

Information concerning the platform requesting migration and the state of the application running on it is collected and processed in order to activate the application on the target platform without losing interaction continuity. Since the number of presentations and the tasks supported by the various platforms may be different, it is not possible to create a one-to-one correspondence between presentations for different platforms. Source and target platform versions are generated by TERESA separately, using the information contained in the two corresponding task models. One important issue is how to identify the presentation for the target platform corresponding to the one active on the platform requesting migration, while maintaining the state of its interaction objects. The run-time state, consisting of the visualized page and the state of its objects, is mapped first onto the corresponding abstract presentation and then onto the corresponding set of tasks. The page to be visualized on the target device will be identified using the inverse process: from the set of tasks to support, the tool identifies the most similar abstract presentation and then the corresponding page in the application version for the target platform.
Similarity is calculated in terms of tasks supported, the more similar the tasks associated to the two presentations are, the more similar the presentations will be. Presentation similarity is the basic criterion to be considered, but under particular conditions it may not be enough. When the migrating presentation supports a task set that is associated with multiple presentations in the target version, each of which supports the same number of tasks, then the similarity will be the same for each potential target presentation. Thus, a further criterion is used to decide which target presentation to activate. To this end, we identify the target presentation supporting the task associated with the interaction object last modified by the user, since the user is most likely to continue interaction from that point.
Once the target presentation has been identified, it is necessary to calculate the state of the objects contained in the corresponding page, which will be communicated to the target device along with its URL. For this purpose, data referring to the runtime state of the application will be associated to the corresponding tasks and adapted to the object implementation for the target device.
One potential issue for migrating interfaces to a target device where the same task is supported by means of different interaction objects is whether the change of user interface can disorient the users. Since our migration service is designed to address TERESA-generated interfaces, this potential problem is taken into account because the tool takes into account the tasks that the application should support and for each of them only the concrete objects suitable for their support are used for implementation. The actual interaction object to be used in generating the user interface will also depend on the kind of platform it is intended for. Hence, interaction objects that can disorient the user will not be proposed to support the task performance.

6 Migration Service Architecture

We aim to support user interface migration for a wide variety of devices such as desktops, laptops, PDAs, cellular phones and generally any device able to access the Internet through a browser. Our migration service relies on a server machine working as a Web server making accessible the platform-specific application implementations as well as a migration server managing context information to support migration requests. Client platforms use the migration client loaded from the server in order to enable or disable the possibility of receiving incoming applications and migrating Web applications. References to all platforms, which enable the reception of incoming applications, are stored in the server.
Figure 2 shows the user interface for the control service: it is possible to access the list of migratory applications available and the list of target systems that are currently enabled to the migratory service, request a dynamic update of such information and trigger the migration of the current application.
When a platform asks for migration, the request sent by the locally running migration client, reaches the migration server, which will exploit both runtime and static context data to perform the presentation mapping process as described in Section 5.

Figure 2: The interface for the migration service in the PDA and desktop environments.

The corresponding page and its runtime context for the target device will be finally sent to the migration client in the target platform that will open a local browser window allowing users to continue their interaction (the sequence of functionalities to perform is indicated in Figure 3).

Figure 3: The Migration Process.

A number of modules and algorithms are used both for starting up the migration server and performing migration requests.

6.1 Web Server Startup. No specific algorithms have been implemented for the web server.

6.2 Migration Server Startup. The migration server must be started in order to support migration requests. In this phase an internal data structure is filled in with static data concerning all applications supported by the migration service. For each application and for each platform, information regarding associations between different abstraction levels is built to allow the matching of the presentations to supported tasks and the abstract user interface elements to the corresponding user interface elements. Hence, for each platform-specific version of the application, the XML file defining the corresponding abstract user interface is analysed to retrieve the set of presentations making up the specific version of the application and the tasks supported by each presentation. Finally, for each presentation the static data are completed by adding the concrete type, name and identifier of the concrete elements implementing the supported tasks. The concrete data are retrieved by analysing the file implementing each presentation.

6.3 Migration Service Loading. Users who want to access the migration service have to request it by loading the client service manager, which depends on the actual platform. In any case, the client migration service will store information on the plat-

form accessed by the user communicating them to the server, and a graphic interface is activated. A further client module is started in order to allow incoming application migration acceptance.

6.4 Client Migration Request Sending. On the source client side, JavaScript functions collect runtime data concerning the URL of the loaded application page and the state of the interaction objects contained in the page when the user requests migration. A migration request, including the IP address of the target device and all runtime data collected into a single string, is sent to the server by submitting a form, which causes the migration server run-time module activation.

6.5 Server Migration Request Elaboration. Once the migration request is received, the server checks whether the target IP address matches a currently connected platform, and the corresponding platform type is retrieved, hence the URL string of the migrating page is analysed in order to extract the name of the migrating application. Also the set of corresponding tasks are retrieved and matched against the whole set of presentation data corresponding to the target version of the application in order to retrieve the most similar presentation. The most similar target presentation is the one sharing the highest number of supported tasks with the source presentation. In case this criterion identifies multiple presentations, the runtime state data is used to identify the one containing the task corresponding to the last interaction performed by the user. At this point it is possible to build the URL string that must be loaded on the target device. Once the target URL has been built, the runtime state string must be modified, in order to map the state of source interaction objects to target ones. Information related to tasks that are not supported by the target presentation is eliminated from the string. Tasks implemented by the same kind of interaction objects maintain the runtime state, and their concrete name and ID must be updated. Tasks implemented by means of different interaction objects also need information concerning such objects to be adapted.
Once the mapping of both the URL and runtime state is completed, a connection is opened with the target platform, and the target URL string is sent followed by target runtime state data string.

6.6 Client Migration Request Acceptance. On the target client side, the migration application keeps listening for connection requests coming from the server. Once a connection is received it is accepted and the URL of the page to be loaded and runtime state data are read from it. The incoming URL is loaded in a new browser window in case of a desktop system, and into the browser window that is already opened in case of PDAs. Hence, runtime state data are applied to the concrete object of the loaded page. The application of state information to the interaction objects is performed by means of Javascript functions implemented on client side.

7 A Case Study

In this section we discuss a case study that provides concrete examples of the results that can be achieved through our approach. We first introduce a couple of scenarios that outline the main features covered by the migration service. On the basis of the scenarios, we will show the user interface of a sample migrating, multi-platform application, focusing on the operations that the migration service must perform in order to keep interaction continuity and to adapt the migrating application to the different kind of platforms involved.

7.1 The scenarios

Louis is walking on the street going to his office. He is thinking of vacations and decides to check the state of his bank account, to control how much he can afford to spend for a journey. Louis turns on his own PDA and accesses the bank web site. He had previously registered to the bank web service, hence the bank application automatically identifies the PDA as the Louis personal device. From the main page Louis choose to access his bank account data. He does not need to enter any personal information, neither bank account number, the application has already retrieved all data after having identified the PDA and the actual amount of money in Louis bank account is shown. Louis cannot remember what the previous situation was and would like to access information concerning the last operation performed over his account. He asks the application to see the 10 last money movements, and data appears on the PDA. Only the amounts of money added or subtracted at each operation are shown, to access more details concerning operations, a new page for each operation must be loaded. Moreover operation data are split into two pages, and are not displayed together. Too boring staff, it would be much comfortable to have a full overview of operations and their details and such a feature is not supported by the PDA application version, it would be useless to try and show a large comparative table on a small sized PDA screen. Meantime Louis has reached his office and turned on the Desktop PC. Interacting with the migration application on the PDA, he asks for the bank application to migrate on the Desktop PC. The application is automatically shut down on the PDA and is activated on the Desktop in the exact runtime state it had when migration was required. Bank account related operations are immediately shown on the Desktop without Louis having to re-enter any data or make any request, has he would have had to do accessing directly from the desktop. On the Desktop screen a table showing detailed information for each of the latest 10 operations performed on the bank account is displayed. Louis can easily see that the last movement is a credit versed by the company he works for, and it is a travel expenses refunding, concerning the conference he attended four months ago, conversely his salary has not been credited yet.

Louis decides to go personally to the bank office, in order to withdraw money he thinks to spend during his vacations and interacts with the bank application to check how many people are waiting. He sees that the estimated waiting time is one hour and a quarter, and he decides to pick up a reservation ticket. He gets ticket number 40 and interacts with the migration service to ask for bank application migrating back on the PDA where he can see his reservation number and the real-time bank office situation.

On the PDA he also can turn on the alarm feature, that will alert him when he has to hurry up because his turn is getting closer.

In the second scenario Louis is at home and accesses to the Bank application through his Desktop PC. After having entered identification data, he is allowed to access the bank services. He needs some money transfer in order to pay the fee for a conference registration and accesses the page reserved to on-line operations. He starts filling in the form to perform the transfer, meantime he realises that it is late and it is time to go to work. Through the migrating service, he can migrate the page he was interacting with on his own PDA. On the PDA screen, only the part of the form Louis was filling in when asking for migration is shown and he can complete the form adding the missing data through his PDA, while he catches the metro to go to his office. Before submitting data, he wants to be sure of their correctness and accessing the previous page, he can control the data previously inserted through the Desktop PC. The form is correct and he can submit it. The registration to the conference is now completed.

7.2 The Migrating Web Bank Application.

In this section we introduce the *Web Bank Application*, a sample application built on the basis of the scenarios described in section 7.1. The application is a typical bank application that allows registered users to access their own private data and perform on-line operations. We are presenting here only the most relevant application features outlining how they can benefit of the migration service .

When a user accesses the *Web Bank Application* from a Desktop PC, he is always inquired about identification data, in order to be recognised by the application and being allowed to access the Bank Application Services. A user accessing the application for the first time, can complete a registration form in order to get a login and a password. Once the registration is completed, a user accessing from a PDA will automatically gain access to the application without having to enter identification data. Because of security matters, such a task is not enabled on the Desktop version.

Figure 4. The Desktop Main Page

Figure 5. The PDA Main Page

In Figure 4, it is shown how the application appears to a user loading it from a Desktop PC. If migration to PDA is required at this point, and the user previously registered to the Bank service, the PDA version will not show the same accessing form, the automatic access task will be activated and the user will be identified (Figure 5). Selecting access to information on his personal bank account, the user, Louis in this example, will see immediately the information required, without having to enter any identification data.

In this migration case, after the identification of the target page, the migration service will recognise that the automatic access from PDA is possible, since the user previously registered to the bank service. In addition, there is a set of tasks supported by both the corresponding presentations, such tasks are: *BankAccoutAccess, OperationsAccess, CumulatedIterestsAccess, InvestmentsAccess* and *CreditCardAccess*. On Desktop version, the above mentioned tasks are implemented through clickable button images, while the PDA version implements them by means of simple links. The migration service will have to recognise the different implementation of the tasks and adapt the runtime state retrieved from the page loaded on the Desktop to the one to be loaded on the PDA.

Let us suppose that Louis has accessed his bank account data and selected to have information on the last 10 money movements performed. To see all the operations selected he will have to access two pages and one more page has to be accessed to have more detailed information concerning each operation (Figure 6).

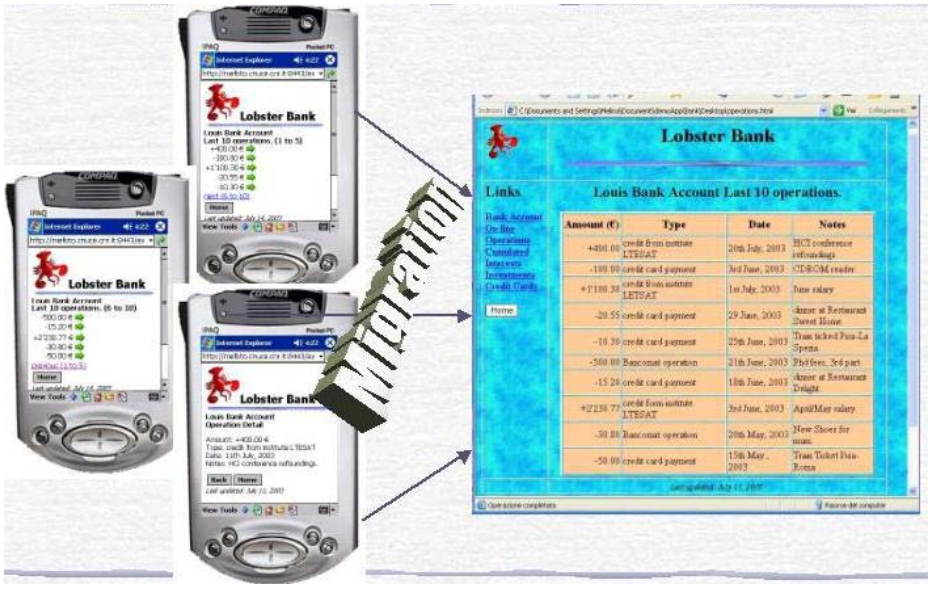

Figure 6: The PDA interface for checking the bank account.

To have a complete single view of the operation selected and their detail, Louis decide to migrate to the Desktop PC. Following the criterion of the most similar page, migration from any of the PDA page presented, the resulting page on the Desktop is the one showed in Figure 6.

Another significant migration case, is when a task performed on the migrating page enable a further task on the target platform, that was not supported by the source platform. The *Web Bank Application* allows a user to reserve a ticket for accessing the real bank office and also to the monitoring of the real-time office situation. Only on the PDA version, the user can ask the application to keep checking the real-time situation and alert the user when his turn is getting closer. Such a task is not present on the Desktop version, because it is supposed to be useful only in case the user is close to the bank office and waiting for his turn. The alerting service is enabled only after a ticket has been reserved. Figure 7 and 8 show the steps Louis has to perform in order to check the office situation and reserve his ticket.

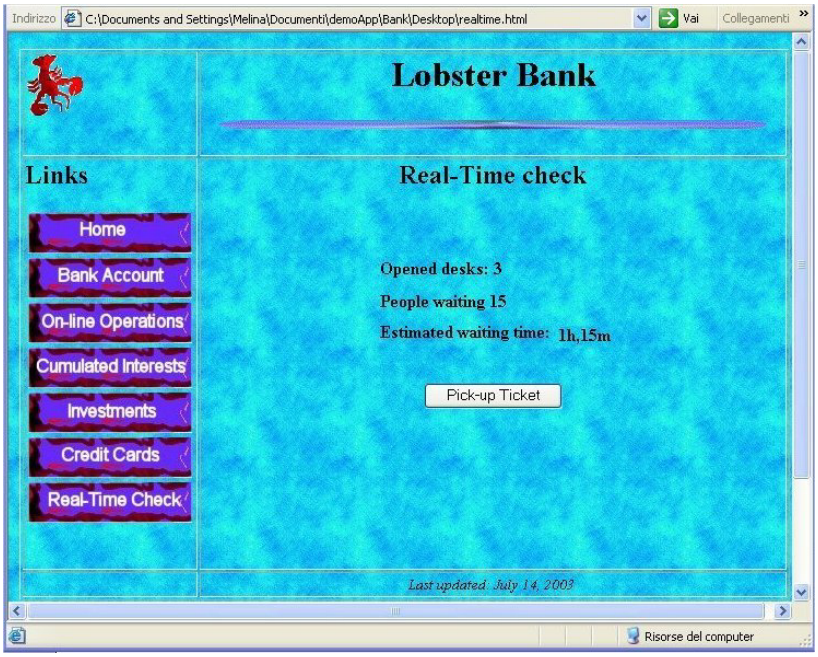

Figure 7. Desktop Real-time Check.

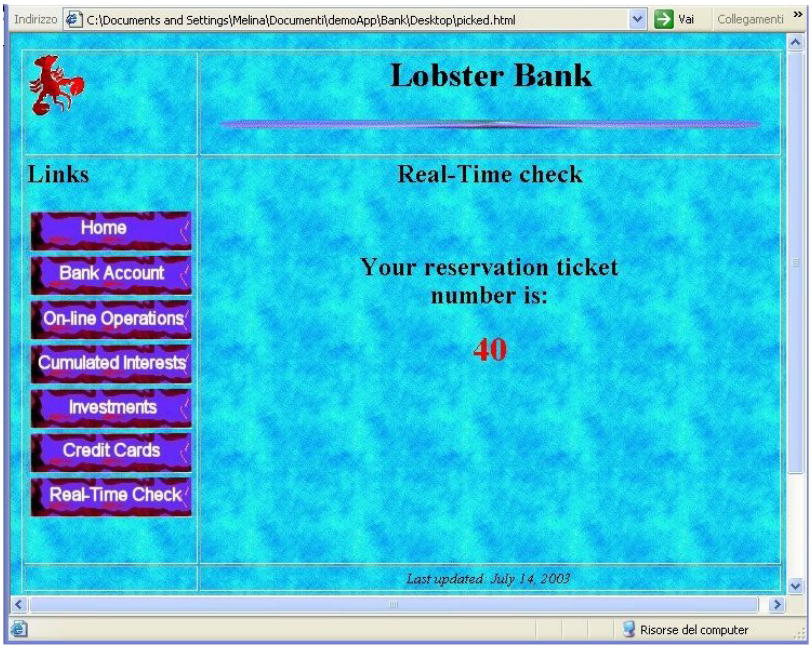

Figure 8. Desktop Ticket Reserved.

Migration required for one of both Desktop pages shown in Figure 7 and 8 will identify the PDA page shown in Figure 9, where the user can decide to activate the alert service.

Figure 9. Real-Time Monitoring on PDA platform.

The above mentioned migration case does not show only the activation of a task from one platform to the other, it is also a good example of a migrating page containing a task performed accessing to a different set of domain objects. On the desktop real-time monitoring page (Figure 7), the task *ShowRealTimeState* is performed accessing the information composed of the objects: opened gates, people waiting, estimated waiting time, while set of information accessed by the corresponding task is the PDA version (Figure 9) is composed of the objects: opened gates, people before you, estimated time and your ticket number.
The following migration case is based on scenario 2 (section 7.1). In Figure 10 we can see the page Louis is interacting with before asking for migration. Let us think he has filled in the first five fields and is now asking for migration. The first step performed by the migration service in order to retrieve the most similar PDA page, identifies two presentations containing the same number of tasks (Figure 10). The second step makes the migration server to select the page in that contains the object implementing the task last performed by the user on the target platform. As a result, page shown in Figure 13 will be loaded on the sent to the PDA. Data previously inserted in the form by the user are not lost, and can be visualised accessing the previous page on the PDA.

Figure 10. Desktop Bank Transfer Form.

7 Conclusions and Future Work.

We have discussed how to support migratory interfaces able to support interaction continuity and usability. The user interfaces are developed following a model-based

approach. A first prototype for total migration addressing applications obtained through the TERESA tool has been implemented and we show an example of application. This approach opens up the possibility of intelligent environments able to support the user through various platforms and allowing them to move from one platform to another one without having to restart from scratch the session. We are now improving the collection of run-time state data in order to make it more complete and improve the support of interaction continuity. Future work will be dedicated to extending this approach in order to support other types of user interface migrations and peer-to-peer architectures.

This work has been supported by the IST EU R&D CAMELEON project (http://giove.cnuce.cnr.it/cameleon.html). We thank colleagues for useful discussions.

References

R. Bandelloni and F.Paternò, Platform Awareness in Dynamic Web User Interfaces Migration , Proceedings Mobile HCI 2003, LNCS, Springer Verlag, 2003

K. A. Bharat and L. Cardelli. Migratory *Applications*. In proceedings of User Inteface Software and Technology (UIST '95). Pittsburgh PA USA. November 15-17, 1995. pp. 133-142.

J. Coutaz, C. Lachenal, S. Dupuy-Chessa. *Ontology for Multisurface Interaction*. Proceedings INTERACT 2003. IOS Press. Zurich, September 2003.

A. Kaikkonen and V.Roto. *Navigating in a Mobile XHTML application*. In Proceedings ACM CHI 2003. Ft. Lauderdale, Florida, April 5-10, 2003. Vol.5, pp. 329-336.

B. MacKey. *The gateway: A Navigation Technique for Migrating to Small Screens.* Doctoral Consortium, CHI 2003. Ft. Lauderdale, Florida, April 5-10, 2003. pp. 684-685.

G. Mori, F. Paternò, and C. Santoro. *Tool support for designing nomadic applications*. In Proceedings of IUI 2003 . ACM Press, 2003. pp. 141–148.

J. Nichols, B. A. Myers, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld, M. Pignol. *Generating remote control interfaces for complex appliances*. Proceedings ACM UIST'02. October 27 – 30. Paris, France. Vol.4, pp.161-170.

F. Paternò, Model-Based *Design and Evaluation of Interactive Application*. Springer Verlag, ISBN 1-85233-155-0, 1999.

F. Paternò, C.Santoro, *A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms, Interacting with Computers*, Vol.15, N.3, pp 347-364, Elsevier, 2003.

H. Song, H. Chu, S. Kurakake. *Browser Session Preservation and Migration*. In Poster Session of WWW 2002, Hawai, USA. 7-11. May, 2002. pp. 2.