Flexible Interface Migration

Renata Bandelloni, Fabio Paternò
ISTI-CNR, via G.Moruzzi 1,
56100 Pisa, Italy
{renata.bandelloni, fabio.paterno}@isti.cnr.it

ABSTRACT
The goal of this work is to provide users immersed in a multi-platform environment with the possibility of interacting with an application while freely moving from one device to another. We describe the solution that we have developed for a service to support platform-aware runtime migration for Web applications. This allows users interacting with an application to change device and continue their interaction from the same point. The service performs the migration of the application taking into account its runtime state and adapting the application interface to the features of the target platforms. The service is optimized for applications developed through a model-based, multiple-level approach. The intelligence of the adaptive interfaces resides in the migration server, which adapts data collected at runtime from their original format to the format best fitting the features of the target platform. We also indicate how it is possible to extend this result in order to support partial migration and synergistic access, by which a part of the user interface is kept on one device during runtime and the remaining part is moved to another with different characteristics.

Categories and Subject Descriptors
H.5 INFORMATION INTERFACES AND PRESENTATION – I2.2 Automatic Programming: Program Transformation.

General Terms
Design, Experimentation, Security, Human Factors.

Keywords
Migratory interfaces, Adaptive interfaces, Multi-platform applications, Remote control.

IUI'04, January 13–16, 2004, Madeira, Funchal, Portugal.

INTRODUCTION
Nowadays, everyday life is becoming a multi-platform environment where people are surrounded by different types of devices through which they can connect to the Internet in different ways. Most of them are mobile and personal devices carried by the user who can move around different environments populated by various other platforms. This kind of scenario raises the need to find a way for the user to deploy his personal devices to get connected and exchange information with the other platforms that may be in the surroundings. These considerations lead to the idea of migrating interfaces among different platforms. 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 personal device. Another interesting case is when the user is interacting with a multimedia application, say a PDA, browsing through images and videos. Both power consumption and the reduced screen size of the device would make it hard to fully enjoy the visualisation of such content. It would be much more appealing to keep interacting with the PDA for the control operations and watch the videos displayed on a big wall-sized screen by just pressing a button.
The short scenarios introduced exemplify the need for a multi-platform migration service, allowing a user to interact with an application through different devices. There are two main issues concerning this kind of service. Firstly, the diversity in features of the platforms involved in migration, such as different screen size, interaction facilities, processing and power 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 needs an intelligent engine in order to adapt its interface to the different features of the target platform. 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 [13].

Two main kinds of information are relevant in performing migration: static information referring to the device features, 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 [5], 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 [10] from which we generate the actual application by means of the TERESA tool [6]. 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 concerns elements selected and data entered. It is transmitted to the server in order to recreate the corresponding state in the application for the target device. In this paper, we moreover describe the main features of the prototype service implementing the framework which was introduced in [1]. The prototype was first designed to support total migration, and has been extended to support partial migration, whereby the user can keep the control part of the application on the original device while migrating the display content part onto another.

We first discuss related work and introduce the TERESA approach to design and development of 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. Finally, we discuss the development of the partial migration module, considering the main issues that have to be tackled for this purpose. We outline the most critical aspects concerning migration, focusing on the steps required to add partial migration capabilities to our system. We describe how to split the migrating application into several parts, basically, the control and visualisation parts, performing a semantic analysis of the application model in order to attain partial migration of the application. In essence, partial migration allows one platform to be the controller for what happens on another. The prototype confirmed the potential of the model-based approach and provides insight into the possibility of building a more enhanced service.

RELATED WORK

In recent years there has been an increasing interest in application migration in the area of ubiquitous computing that considers users immersed in multi-platform environments. A good example of multiple device interaction is shown in [3], where a system allowing a Web page to be seen contemporarily by more than one display is described. Different kinds of clients are defined, able to download, upload or both, the page shown on a screen. The system addresses issues concerning the movement of Web content across several types of displays, but no adaptation to the different features of the client platforms and displays involved is performed.

A more general framework for migratory applications is presented in [2]. In this work the main issue is platform adaptivity. Migration is intended both as total migration of the application interface and as splitting of the interface into several parts to be spread over different platforms. The framework does not pose any limitation on the kind of platform, screen, operating system that can be addressed. The generality of the cases considered causes the resulting architecture to be extremely complex.

The display features of heterogeneous devices may be very different because their interaction facilities may vary as well, influencing the usability of the applications. Work at Nokia Research [4] has shown how different features of the interaction platforms can influence Web applications usability. Results proved that designing usable applications for mobile phones requires different criteria than the ones needed by wider screen device. One of the main goal of our migration service is to keep usability features of applications. Thus, the 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. Techniques used in adapting user interfaces to different devices, in particular to small screens rely on size reduction and data summarization as it is shown in [5]. Such an approach raises the risk to make the application unusable because objects on the page become difficult to recognize. We want to overcome this kind of problem adapting interfaces to different platforms, taking into account the effects on the usability of the application.

While in the past model-based approaches have mainly focused on the design of desktop applications (see for example [12] [14]) in this work we want to exploit their potential to address multi-device migratory interfaces.

A semantic analysis of the abstract description of the application [11] allows us to associate elements in the actual interface with their meaning, hence elaborating the best way to present them depending on the current platform. Beside interface adaptivity, interaction continuity has to be addressed too. The main aspects concerning this problem are discussed in [13]. The system described allows

"freezing" Internet browser sessions and retrieve them in a second time, from another device. The runtime state of the application is preserved and retrieved when the user asks to access the application previously stopped. This sort of migration is based on a stop and go paradigm and is not performed on the fly. In any case, the capture of the session data is an important issue in our work too, what we want to add is platform-dependent adaptivity that is not addressed in the work.

All previously cited works concern the total migration approach. In our vision of a future user interacting with an application through multiple devices, a partial migration framework can be useful as well. This will allow the user to spread the application interface onto several devices, one of which will be used to control all the others. Our approach differs from the remote control style of the Pebbles Slideshow commander [7] that allows a user to control a power point presentation running on a desktop PC from a PDA. The application is very specific and the system cannot be applied to any other kind of migration.

Our approach also differs from the one presented in [8]. In this case a PDA is used as an actual remote controller for various types of home devices, such as video recorders, stereo systems, and so on. We are addressing applications that can run on many devices and allow them to migrate from one platform to another supporting adaptation, while in [8] the application can run only on the PDA that adapts its user interface to the controlled device.

DESIGN OF MULTI-DEVICE 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 analyze their design at different abstraction levels and consequently generates 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, composed of interaction objects [11] classified in terms of their semantics still independent from the actual implementation; the concrete user interface, and the actual corresponding code. The main transformations supported in TERESA are:

- *Presentation Task Sets and Transitions Generation.* From the specification of a task model [9] 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 characterized in terms of the tasks 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 starting with the currently loaded (single-platform) task model, and using a number of default configuration settings.
- *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 customization of the concrete user interface are made available to the designer in order to obtain the concrete 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 XHTML, XHTML Mobile Profile and VoiceXML.

RUNTIME MIGRATION ANALYSIS

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

- *Total Migration*: the client interface migrates totally from a device to the other.
- *Partial Migration*: the client interface is divided into two parts, one for user interaction (control part) and one for information presentation (visualization part). The control part remains on one device, while the presentation one migrates to the other device.
- *Mixed Migration*: the client interface is split into several parts, concerning both control and presentation

and different parts are distributed over two or more devices.

The first version of the service we have developed focuses on *Total Migration*, with the goal to support a runtime migration that takes into account the differences between the two platforms involved. When we migrate an interface from a platform to another one, the runtime support first retrieves the corresponding presentation, hence identifies the closest presentation in the target platform and the associated target page. 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 mapping of a presentation from source onto target platform:

- *One to One.* The source 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.
- *One to Many*. The source 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.
- *Many to One.* Multiple presentations for 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.

The abstract user interface produced by TERESA also includes information that can be deployed to design the partial migration module. This will be discussed in the section dedicated to partial migration.

TOTAL MIGRATION SERVICE

Supporting many platforms means making use of a wide quantity of static data concerning the application features of each platform. The implementation of a migration engine residing only on client side, in a peer-to-peer style, causes a set of data and processing too heavy for very small devices. For this reason we have chosen a solution based on the service provided by a server machine. The server works both as a Web server making accessible the platform-specific application implementations and as a migration server managing context information to support migration requests. The platform that wants to access the migration service, only has to load the migration client directly from the server. The migration client allows the user 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.

The user interface for the control service makes it possible to access the list of migratory applications available and the list of target systems that are currently enabled to the migratory service. It is also possible to request a dynamic update of such information and trigger the migration of the current application. The service offers support for interfaces developed by the TERESA tool and residing on the server machine. When a platform asks for migration, the request sent by the migration client running on the source platform reaches the migration server, which will exploit both runtime and static context data to perform the presentation mapping process as described ahead in this section. 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 1).

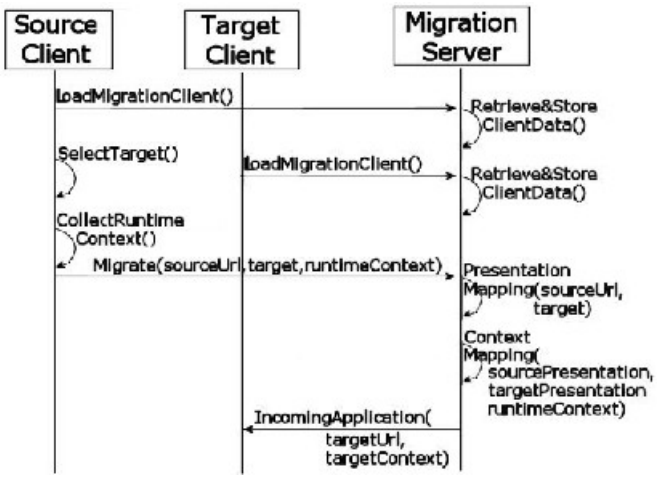

Figure 1: The Migration Process.

Our migration service is designed to meet three main requirements, device awareness, interaction continuity and support of usability criteria. 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. To this aim, the run-time state of the application, consisting of the visualised page and the state of its objects, is deployed.
First, the abstract presentation corresponding to the migrating page is retrieved. At this point the corresponding task set is retrieved too. The page to be visualised on the target device will be identified using the inverse process: from the task set to support, the tool identifies the most *similar* abstract presentation and then the corresponding page in the application version for the target platform (see Figure 2).

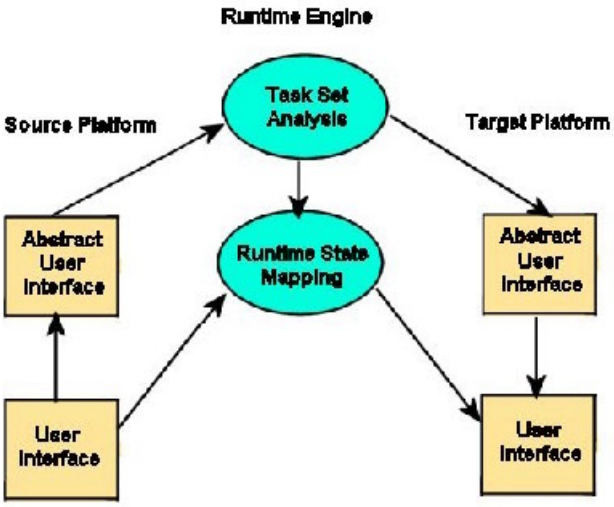

Figure 2: Runtime Presentation Mapping.

Similarity is calculated in terms of supported tasks, 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, while data concerning tasks not supported by the target device, will be ignored. The objects on the target page, supporting tasks that where not available on the source one, will assume their default state.
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 considered because the

tool takes into account the tasks that the application should support and for each of them only the interaction techniques suitable for their support through the current platform are used for the implementation.

PARTIAL MIGRATION

The total migration service basically allows the user to change the device deployed to interact with an application. In this operation the intelligent core of the system is in charge of keeping interaction continuity and supports interface adaptivity to different platforms. What we are addressing with partial migration is the ability to move from interacting with an application through a single platform, to controlling one platform from another. This allows users to comfortably control, for example, videos displayed on a wall-sized screen from their handheld PDA, or projecting a presentation stored in a personal device like a PDA to the desktop-controlled maxi video screen in a conference hall, while maintaining control on the personal device. Partial migration requires more complex processing than total migration.

The interaction continuity and platform awareness criteria that we adopted in the prototype version of the total migration service have been deployed in partial migration as well. The novel and most difficult issue to consider is the splitting of the application into its visualisation and control parts.

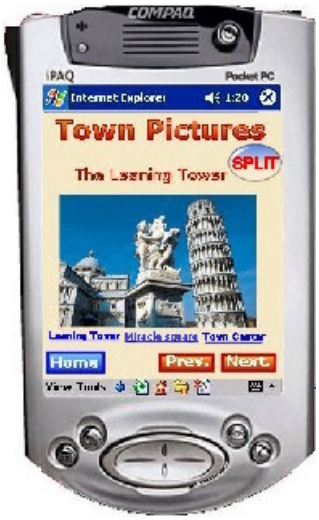

Figure 3: Sample Application.

Figure 4: Control Part on PDA.

Figure 3 shows a sample PDA application, to which partial migration towards a desktop PC is applied. Figure 4 illustrates the result on the PDA, where only the control part remains and the objects on the page have been rearranged in order to provide a pleasant and usable interface. The user can now select images using the handheld device and look at them on the desktop PC as shown in Figure 5. In this case, we have partial migration from the device maintaining the control part to the one that will visualise the result of the user interactions; the service is designed to support the inverse case too.

As for total migration, applying partial migration to every possible Web page is not simple. This necessarily calls for restricting the range of applications to take into consideration. Based on our total migration prototype experience, we have decided to continue considering applications developed with the TERESA tool.

Even with this approach, which provides a useful logical description of pages, the range of cases to be considered is still wide. In particular, it is not always possible to split the application into exactly two parts. Problems arise when the presentation resulting from a control action contains some additional control objects, and partial migration could thus lead to an unusable and confusing split interface, with control objects on both the source and target platform.

Analysing the different possibilities that may arise, we have identified two fundamental cases we consider well-suited for partial migration:

- *Case1(page splitting)***:** the migratable page contains both control objects and the result of control actions. Partial migration will be performed by loading a page containing only the control objects of the migrating page onto the source platform. The target platform, on the other hand, will show the content part, which will disappear from the source, where only control objects are to remain. Any time the user selects a control object to modify the visualization part, then the result of the action will be shown on the target and no longer together with the controls on the source.
- *Case2(remote content control):* the page contains control objects that, when selected, cause the loading of a new page on the device. When partial migration is performed, the source still shows the page with control objects and when the user performs an interaction, the referred page is loaded on the target in a suitably adapted version. The user can continue selecting new pages from the source device and the result of the actions will be shown on the target.

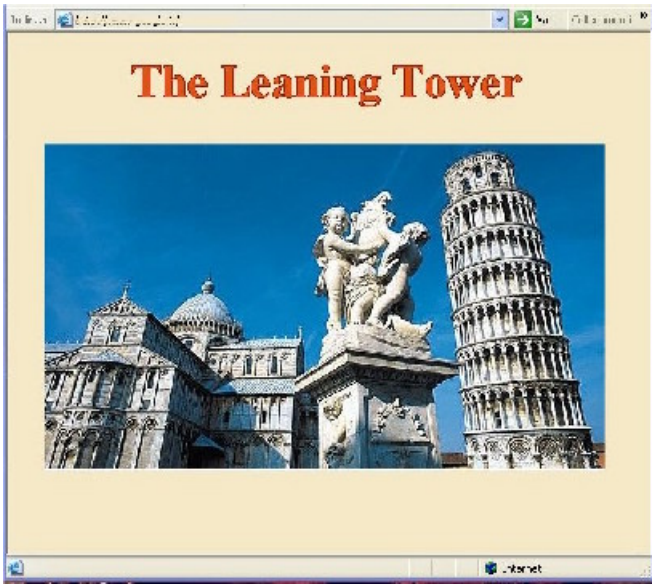

Figure 5: Visualisation Part on Desktop PC.

PARTIAL MIGRATION SOLUTION

The logical description of the application produced by TERESA provides more information than that used for total migration. Such information is fundamental to help decide which objects in the page must be considered the control part and which make up the visualisation part. In the first case, which is the splitting of the interface, a first analysis is based on the types of objects contained in the page. The description of a presentation is made in terms of two types of interactors: *Interaction* objects, for example objects supporting selection, editing, control, ; and *OnlyOutput*, for example text and graphic presentation objects. This first simple classification is not enough to decide how to actually split the interface for partial migration. Thus, control objects, which belong to the *Interaction* interactors, are classified into two types: activators and navigators. Activators are control objects used to generate some events that cause changes in the page containing them, while navigators are control objects that cause a new page to be loaded. Secondly, *OnlyOutput* and *Interaction* interactors can be strictly connected and dividing them may affect the logic behind the whole application. For example, an *OnlyOutput* interactor could contain the description of some *Interaction* interactor and they should thus be kept together.

Such relations are also described in the abstract user interface by means of the composition operators Ordering, Grouping, Hierarchy and Relation, already discussed in the section dedicated to the logical description of user interfaces. The strongest relation is the one concerning objects affected by Grouping, this operator applies to objects intended for tightly related tasks, hence they must not be separated. Relation is a candidate for splitting because it is often composed of a control part and controlled ones. The Hierarchy operator (H) is an indicator of the importance of the elements contained in the page. It identifies the tasks that must be highlighted in the actual interface, in effect giving them a sort of higher visual priority over the other elements of the page. In the process of page splitting, the element having the highest level for the H operator is considered a possible candidate for display in the larger screen. Hence, the H operator will be split in the event that the set of arguments can be divided into two parts: the first one contains the main visualization part and the second only interaction objects.

The Ordering operator (O) can also identify a potential splitting point. It relates to elements that are correlated by some kind of order, such as temporal, and can apply to both visualization and control elements. This is the case in which performing some operations like making a selection, causes some other data to be visualised. In such a situation a set of interaction objects has to be manipulated in order to change the content of some other application objects.

As the O operator arguments are mostly interaction objects, they will usually precede all application objects. From this consideration we will consider O operator splittable only in case the set of its argument can be divided into two parts, the first of which contains only interaction objects and the second one containing only application objects.

Figure 6 outlines the logical interactions involved in a sample application and how operators compose them. In particular, we have *PickImage1, PickImage2, PickImage3, BackHome, PreviousImage, NextImage* as *Interaction* interactors *and ShowMainInfo*, being an *OnlyOuput* interactor.

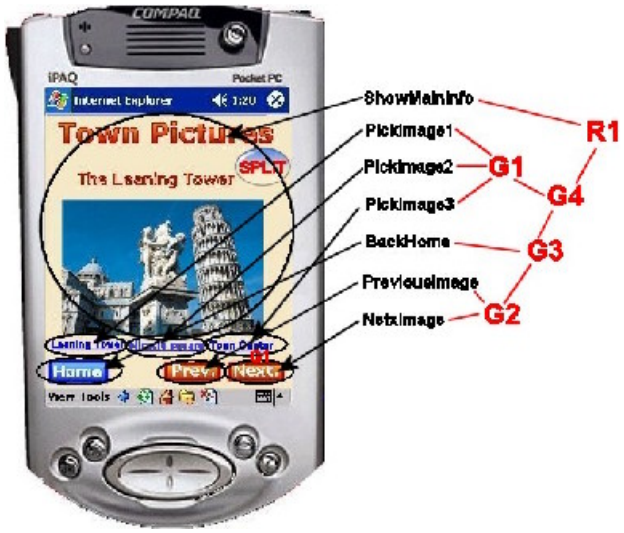

Figure 6: Operators and Tasks.

Using **G** to identify Grouping and **R** for Relation, the compositions of operators applied to the abstract interactors of the sample is:

R(ShowMainInfo,**G**(**G**(BackHome,**G**(PreviousImage, NextImage)), **G**(PickImage1, PickImage2, PickImage3)).

Indeed, in the interface we can see that *NextImage* and *PreviousImage* are grouped through lateral adjacency, then they are grouped with *Home*, as they are kept in the same line. The result is grouped by vertical adjacency with the three image selections (which are once again grouped through adjacency). All of them are related to *ShowMainInfo* because if one of them is selected then the main information presented changes. The automatic identification of this potential split point allows the migration service to define how to split the user interface into two parts presented simultaneously on two different devices.

When considering Case2, the page loaded on the current control platform remains as is, and the new pages selected are loaded one by one on the target platform. The issue here is how to automatically distinguish the two cases. To this end, we use information in the abstract interface where the control interactors are classified depending on whether they correspond to elements that generate new content on the current page or cause the loading of a different page. The two cases are distinguished on the basis of the type of control interactors contained. When all the control interactors on a page support navigation, we are dealing with Case2.

If the source page contains control interactors for accessing content and navigation, we use a mixed approach. The splitting of the page is performed, and if the user interacts with a navigation interactor on the control platform, then the visualisation page is sent to the target device.

CONCLUSION AND FUTURE WORK

We have presented a solution to support total and partial interface migration for applications developed using a model-based approach. The semantic information produced in the development process is used for building an efficient migration service. The total migration framework deploys only part of the information contained in the abstract description of the user interface. Careful analysis of the description of control objects can be used in order to support partial migration able to split a presentation into a control part and a presentation part.

Web interfaces not developed with TERESA can also migrate, but they do not benefit from the adaptation to the new platform, since it is not possible to retrieve the logical description required by the migration server. A possible approach to overcome the limitation could be to apply a reverse engineering tool calculating the logical description of the page at runtime. Future work will also be dedicated to extending the partial migration cases considered and to engineer the implementation of run-time support. We also plan to consider a further solution where the abstractions are used at run-time to track and support run-time operations for the migration service and other functionalities such as context-dependent help.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the IST EU R&D CAMELEON project (http://giove.cnuce.cnr.it/cameleon.html) and would like to thank our colleagues in the project for useful discussions.

REFERENCES

1. Bandelloni, R., and Paternò, F. *Platform Awareness in Dynamic Web User Interfaces Migration*. Proceedings Mobile HCI 2003, LNCS 2795, Springer Verlag, 2003, pp.440-445.
2. Coutaz, J., Lachenal, C., Dupuy-Chessa, S. *Ontology for Multisurface Interaction.* Proceedings INTERACT 2003. pp.447-453, IOS Press. Zurich, September 2003.
3. Johanson, B., Ponnekanti, S., Sengupta, C., and Fox, A. *Multibrowsing: Moving Web Content Across Multiple Displays*. In proceedings of UBICOMP 2001. LNCS 2201, Springer Verlag. pp 346-353.
4. Kaikkonen, A., and Roto, V. *Navigating in a Mobile XHTML application*. In Proceedings ACM CHI 2003. Ft. Lauderdale, Florida, 2003. Vol.5, pp. 329-336.
5. MacKey, B. *The gateway: A Navigation Technique for Migrating to Small Screens*. Doctoral Consortium, CHI 2003. Ft. Lauderdale, Florida, 2003. pp. 684-685.

6. Mori, G., Paternò, F., and Santoro, C. Tool support for designing nomadic applications. In Proceedings of IUI 2003. ACM Press, 2003. pp. 141–148.
7. Myers, B.A. *Using Hand-Held Devices and PCs Together*. Communications of the ACM. Volume 44, Issue 11. November, 2001. pp. 34 - 41
8. Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R., Pignol, M. *Generating remote control interfaces for complex appliances. Proceedings* ACM UIST'02. Paris. Vol.4, pp.161-170.
9. Paternò, F. *Model-Based Design and Evaluation of Interactive Application.* Springer Verlag, ISBN 1-85233-155-0, 1999.
10. Paternò, F., Santoro, C. *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.
11. Paternò, F., Leonardi, A. *A Semantics-based Approach to the Design and Implementation of Interaction Objects.* Computer Graphics Forum, Blackwell Publisher, Vol.13, N.3, pp.195-204, 1994.
12. Puerta, A., Eisenstein, J. *Towards a General Computational Framework for Model-based Interface Development Systems.* Proceedings ACM IUI'99, pp.171-178.
13. Song, H., Chu, H., Kurakake, S. *Browser Session Preservation and Migration.* In Poster Session of WWW 2002, Hawai, USA. 7-11. May, 2002. pp. 2.
14. Vanderdonckt, J., Bodart, F. *Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection,*. In ACM Proc. of INTERCHI'93, ACM Press, New York, 1993, pp. 424-429.