Dynamic Generation of Web Migratory Interfaces

Renata Bandelloni, Giulio Mori, Fabio Paternò

ISTI-CNR

Via G.Moruzzi 1, 56124, Pisa, Italy

ABSTRACT
In this paper, we present a solution for dynamic generation of Web user interfaces that can dynamically migrate among different platforms. The solution is based on a migration/proxy server able to automatically convert a desktop service into a service accessible from a different platform, such as a mobile one. This solution can support new environments where users can freely move about and change interaction device while still continuing task performance and accessing the application in a usable manner.

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

General Terms
Design, Human Factors, Languages.

Keywords
Migratory Interfaces, Ubiquitous environments, Model-based design, Automatic transformations.

1. INTRODUCTION
Recent technological evolution is characterized by the increasing availability of a wide variety of interaction devices, in particular to support the mobile user. This poses a number of challenges to designers and developers of interactive services that can be accessed through various devices. Developing a version for each type of platform separately can be very expensive in terms of time and can also generate inconsistent results. Style sheets can provide some support by rendering the same element in different manners according to the type of platform but they are still limited since they cannot change the structure of the interface, which is sometimes necessary to better take into account the features of the device at hand.

One more general solution can be obtained with the use of logical descriptions able to indicate the tasks that the system aims to support and the type of communication effects that should be achieved. This type of description can then be analysed to generate the corresponding user interface according to design criteria specific for the target platform. By platform we mean a set of devices that share similar interaction resources, such as the desktop, the PDA, the vocal interface. To further complicate the issue we have to take into account that sometimes the user would like to change device and carry on the task started with the previous one. This issue has stimulated a good deal of attention to migratory interfaces, which are interfaces that can transfer among different devices, allowing the users to continue their task. They are useful in environments where people can move, change context, while still continuing their activities. For example, the user can be playing at home with a desktop system, then realises that it is getting late and has to leave but still wants to continue the game. Thus, he takes a PDA and interacts with it until he reaches the car where the game is finished through a vocal interface while driving. Supporting similar scenarios implies having an infrastructure able to detect requests for migration, identify possible new target devices and therein activate an interface able to adapt to their features, still maintaining the state resulting from the user interactions performed in the first device.

In Aura [7] the solution proposed is mainly to change applications supporting the same service through different devices. Thus, for example, if users have to edit text and have a PC, then they can use MS-Word, whereas if they have a mobile phone then they can use a NotePad-like application, because it requires less resources for execution and interaction. We aim to provide a more flexible solution where the user can still access the same application through different devices but with user interfaces able to adapt to the interaction resources at hand.

Dygimes [6] is another approach proposed for dynamically generating multi-device user interfaces. The authors use an annotated version of the task level logical description. In Dygimes migratory interfaces were not addressed. The same group proposed in [5] a solution based on logical descriptions for supporting distributed user interfaces. Migratory interfaces are different from distributed user interfaces, where the interface runs in one device and is allocated to multiple interaction resources connected to that device (for example two screens). In dynamic distributed user interfaces the allocation of the user interface parts to the interaction resources is dynamic (for example, moving one window from one screen to another or changing from graphical to vocal modality) but they are not migratory interfaces because the interface is always executed in the same device.

CAMELEON-RT [1] is a proposed general reference model to be compliant with when designing a user interface aiming to support migration, distribution and adaptvity to the platform (termed

MobileHCI'05, September 19–22, 2005, Salzburg, Austria.

plasticity in their article). In our work we propose a concrete architecture focused on supporting migration based on a client/server infrastructure. In addition, we can support migration of user interfaces associated with applications hosted by different servers thanks to the proxy capability of our system.

Different techniques for redesigning Web applications for small devices have been proposed in [4] and [8], both oriented to obtain a thumbnailed view of the original page. These approaches are mainly based on the layout analysis of the page and the small screen pages accessed through the thumbnails are built by extracting parts from the original page. Besides the layout, we also consider redesigning the task support, and we change the associated implementing interaction objects in the newly generated page by selecting those best suited for presentation in the small device.

A solution for supporting migratory interfaces was presented in [2]. While that solution contains various interesting elements it has a strong limitation: it uses pre-computed interfaces developed with a specific tool, TERESA [9]. This tool works in a top-down manner. It starts with logical descriptions and then generates the corresponding user interfaces. At run-time the state resulting from the previous user interactions with the source device is adapted to the new user interfaces and associated with it. The system also identifies the point in which the user interface should be activated in the target device. Since TERESA is only a research tool, requiring its use poses a severe limitation to this approach.

In this paper, we present a new solution, which overcomes such limitations. Indeed, in this solution we only require that a desktop version of an interactive service exist. It does not matter how it has been developed or which authoring environment has been used for this purpose. Thus, when a request to migrate to one platform different from the desktop arrives, it automatically generates a new user interface version for the new device associating the current state resulting from the user interactions to it. This is obtained by first performing a reverse engineering transformation able to reconstruct the logical description underlying the current desktop version, and then applying a semantic redesign transformation able to generate a version for the target platform taking into account the tasks to support and the features of the platform. The first version of our migratory server works with desktop Web applications and dynamically generates the possible versions for various types of mobile devices.

In the paper we first discuss the architecture of our migration environment, next we discuss the rules that we have designed for reverse engineering the logical description of the user interface and those for performing semantic redesign. Then, we show an example application of our tool. Lastly, some conclusions and indications for future work are provided.

2. THE ARCHITECTURE OF THE MIGRATION ENVIRONMENT

We have designed an architecture able to support migratory interfaces based on a migration/proxy server. The architecture works currently for Web applications but can be generalised with little effort to other implementation environments.

One basic assumption is that the desktop version for the application has been developed but access from various platforms can occur. This solution seems reasonable because the majority of existing applications have been designed with this type of platform in mind. This also implies that we have available content for the most powerful platform, which can be transformed or minimised for the other platforms.

The client devices that want to access the service, first have to load the migration client. This will allow the server to identify them and recognise their platform, and will enable the devices to issue migration requests. The migration/proxy server behaviour can be described analysing three basic situations:

- The device used to access the service belongs to the same platform type for which the pages where created for (desktop). The migration/proxy server retrieves the required pages from the Web server and passes them on to the client.
- The device used to access the service belongs to a platform different from that for which the pages where created. As an example we can think of a PDA client accessing the desktop designed pages. Here the migration/proxy server retrieves the required page from the Web server, then the page is redesigned for the PDA and the result is sent to the client.
- The device accessing the Web is the same platform type for which the pages where created and at a certain point migration towards a different platform is required. This is the most complex and interesting case, in which all the functionalities of our migration/proxy server are involved. First, the target device of the migration has to be identified. In the example in Figure 1 we consider a desktop to PDA migration. Thus, the page that was accessed through the source device in the migration is stored and automatically redesigned for the PDA. This is enough to support platform adaptation, but there is still one step to be performed in order to preserve interaction continuity. The runtime state of the migrating page must be adapted to the redesigned page. In addition, the server has to identify the first page to activate in the target PDA. At the end of the process the adapted selected page is sent to the PDA from which the user can continue the activity which was left off on the desktop.

Figure 1 shows what happens in the scenario of use within the proposed architecture more in detail. A request for an access to a desktop page is first sent (1). This goes through the migration and proxy server (2 ,3, 4). When a request for migration is sent (5) the server identifies the possible target device based on an analysis of the currently available device position and features (6). In this case a PDA has been identified. Thus, the desktop page is redesigned for the PDA platform (7). For this purpose the server first performs a reverse engineering able to identify the logical description of the page, then exploiting this information a semantic redesign transformation is applied. The result is the generation of PDA pages corresponding to the desktop page. In addition, through an analysis of the tasks supported by the source page, the server is able to identify the PDA page corresponding to the last task performed on the source device. Another functionality performed by the server is to take the state resulting from the user interactions on the source device (such as text

entered, elements selected) and to associate it to the pages created for the PDA platform. In this way the user still has available the results of the interactions through the previous device. Thus, the PDA page corresponding to the portion of the desktop page which was last used on the source device is presented to the PDA (8) and the user can carry on the task. When a new page is selected then the process is applied again (following steps 9, 10, 11, 12, 13). The selected desktop page is transformed in the same manner in order to be presented to the PDA in such a way as to be suitable to its interaction resources.

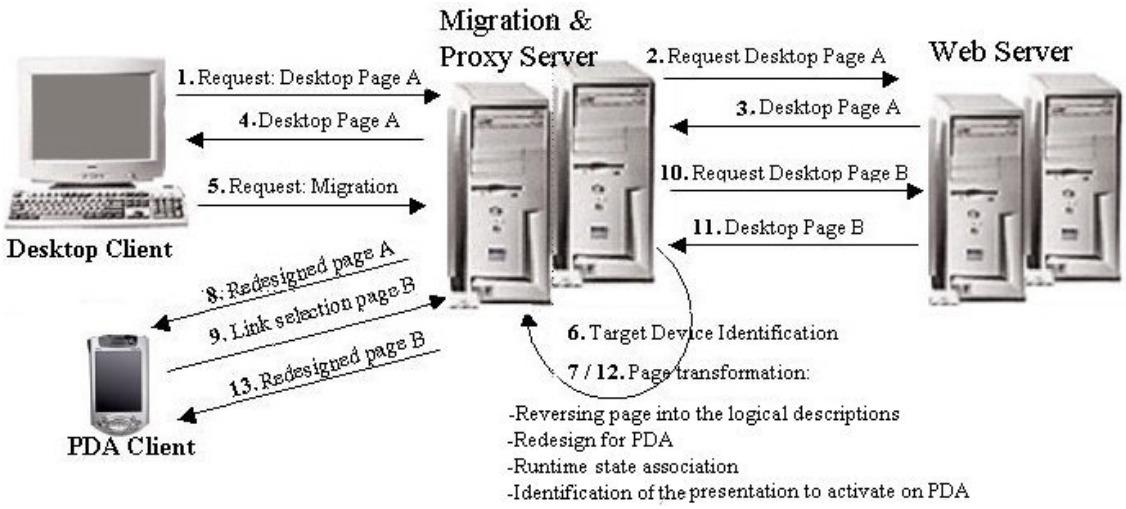

Figure 1: The architecture of the migration environment

3. THE REVERSE ENGINEERING TRANSFORMATION

This work is based on the assumption that there can be different logical views on an interactive application:

- The task level, where the logical activities are considered;
- The abstract interface level, consisting in a modality-independent description of the user interface;
- The concrete interface level, consisting in a modality-dependent description of the user interface but independent of the implementation language;
- The user interface, the actual implemented user interface.

There have already been proposals aiming to provide some support for reverse engineering user interfaces. For example, WebRevEnge [10] automatically builds the task model associated with a Web application, whereas Vaquita [3] and its evolutions build the concrete description associated with a Web page. In our case, we have developed new transformations able to take Web pages and then provide any of the three possible logical descriptions (task, abstract interface, concrete interface). In particular, in order to support the automatic redesign for migration purposes, we need to reconstruct concrete and abstract description and the task description.

The information regarding the abstract description is also integrated in the concrete description. In fact, the concrete description is a refinement of the abstract description obtained by adding information regarding concrete attributes to the structure provided by the abstract description. The abstract description level represents platform-independent semantics of the user interface and it is responsible for how interactors are arranged and composed together (this will also influence the structure of the final presentations). The concrete description represents platform-dependent descriptions of the user interface and is responsible for how interactors and composition operators are implemented in the chosen platform with their related information content (text, labels, etc.).

The abstract description is used in the redesign phase in order to drive the changes in the choice of some interaction object implementations and their features and rearrange their distribution into the redesigned pages. Both task and logical interface descriptions are used in order to find associations between task support implemented in the original interface and in the redesigned one and associate the runtime state of the migrating application.

3.1 From Web Pages to Their Logical Descriptions

The logical description of the user interface is organised in presentation(s) interconnected by connection elements. Presentations are made up of logical descriptions of interaction objects called interactor elements. The interactor objects can be combined by composition operators. Connections are defined by indicating the source and target presentation, and the interactor in the source presentation triggering the activation of the target presentation.

The reverse engineering tool takes a whole Web site or single page designed for a desktop platform and generates the corresponding logical descriptions. Each page is reversed into a

presentation and each interaction object into an interactor. Interactors are composed by means of composition operators. The goal of such composition operators is to identify the designers' communication goals, which determine how the interactor should be arranged in the presentation. Thus, we have a grouping operator indicating that there is a set of elements logically connected to each other, a relation operator indicating that there is one element controlling another set of elements, a hierarchy operator indicating that different elements have different importance for users, and an ordering operator indicating some ordinal relation (such as a temporal relation) among some elements.

The reversing algorithm processes the DOM tree of each page. In order to acquire it, we need to have well formed X/HTML files. Since many of the pages available on the Web do not satisfy such requirement, before starting the reversing phase, each page is parsed using the W3C Tidy parser, which corrects features, such as missing and mismatching tags, and returns the DOM tree of the corrected page. Each page is mapped onto a presentation. The reversing algorithm recursively analyses the DOM tree of the X/HTML page starting with the body element and going in depth. For each tag that can be directly mapped onto an interactor a specific function analyses the corresponding node and extracts information to generate the proper interactor or composition operator. In the following table we show how X/HTML and logical elements are associated. Given that the semantic distance between the implementation and the logical user interface description is not great, associations usually provide meaningful results.

Table 1: Associations used in the reverse engineering process.

X/HTML element	Abstract element / operator
Ordered List	Ordering
Unordered List	Ordering
Table	Grouping
Table Row	Grouping
Table Data	Grouping
Select	Selection
Textarea	Textfield
Form	Relation
Input text	Textfield
Input checkbox	Selection
Input radio	Selection
Input reset	Activator
Input submit	Activator
Input button	Navigator
Div	Grouping
Fieldset	Grouping
Anchors	Navigator
Text	Description
Img	Description

After the first generation step, the logical description is optimised by eliminating some unnecessary grouping operators (mainly groupings composed of one single element) that may result from the first phase. Then, according to the X/HTML DOM node analysed by the recursive function, we have three basic cases:

- The X/HTML element is mapped into a concrete interactor. This is a recursion endpoint. The appropriate interactor element is built and inserted into the XML-based logical description.
- The X/HTML node corresponds to a composition operator. The proper composition element is built and the function is called recursively on the X/HTML node subtrees. The subtree analysis can return both interactor and interactor composition elements. Whichever they are, the resulting concrete nodes are appended to the composition element from which the recursive analysis started.
- The X/HTML node has no direct mapping to any concrete element. If the element has no child nodes, no action is taken and we have a recursion endpoint, otherwise recursion is applied to the element subtrees and each child subtree is reversed and the resulting nodes are collected into a grouping composition.

3.2 From the Logical User Interface to the Task Model

Each logical presentation can contain both elements that are the description of single interactor objects and composition operator elements. The composition operators can contain both simple interactors and multiple composition operators. Our reverse engineering transformation builds a task model represented through the ConcurTaskTrees (CTT) notation [11]. For each presentation a CTT abstraction task node is built, to which the subtrees obtained by reversing the elements contained in the presentation are connected through the appropriate temporal operator.

Each composition operator in the logical user interface is reversed into an abstract task node, whose children are the tasks obtained by reversing the elements to which the operator applies. The reversed children are connected through CTT temporal operators as shown in Table 2. The relation operator is usually associated with cases where there is one control element that can trigger some other activity while disabling other interactions which were available concurrently.

Table 2: Associations used in logical interface to CTT reverse engineering.

Composition operator	CTT Temporal Operator
Grouping	Interleaving
Ordering	SequentialEnabling
Hierarchy	Interleaving
Relation	Interleaving among interaction elements Disabling with control element

Each interactor is reversed into the corresponding CTT task.

A CTT task element is characterised by its "category" and "type". The category indicates how the task performance is allocated and can take the following values:

- abstraction: for higher level tasks with subtasks allocated differently. This category of task is associated with composition operator elements and the overall access to one presentation.
- interaction: for tasks obtained by reversing interaction interactor elements.
- application: for tasks obtained by reversing only-output interactor elements.

Once each single presentation has been reversed, the corresponding CTT subtrees must be composed to build up the whole application task model tree. The root node of the model is an abstraction task representing access to the whole application. The task sub-models associated with single presentations are inserted, directly or grouped through a further abstraction task, as children of the root task. The order in which tasks associated with each presentation are inserted in the overall model, the temporal operators connecting them and their possible groupings depend on the connections among presentations in the user interface logical description.

4. SEMANTIC REDESIGN

Semantic redesign adapts desktop presentations to the limited resources of mobile devices. Generally, desktop presentations must be split into a number of different presentations for the mobile devices. To avoid division of large pages into small ones which are not meaningful, this transformation considers both abstract and concrete descriptions of presentations. The abstract description (based on semantic of interactors and composition operators) is important because it identifies the original communication goals of the designer that should be preserved in the newly created mobile presentations. Concrete descriptions are important as well because they allow for assessing how many interactors can be inserted in a newly created mobile presentation on the basis of their dimensions in pixels and their implementation. For example, a single selection can be represented as a list box in a desktop presentation, but this (depending also on the number of choices) may not be suitable for a PDA presentation and so it has to be transformed into a pull down menu.

Dividing presentations requires a change in the navigation structure, with the consequent need for additional navigator interactors. Our transformation works exploiting semantic information, such as that provided by the composition operators, which indicate semantic relation among elements that should be preserved in the target device as well. In particular, the semantic redesign algorithm follows these main criteria:

- Splitting desktop presentations for mobile devices while keeping in the same presentations interactors composed through the same composition operator in order to maintain semantic relations among interactors as in the desktop presentations;
- selection of interactors in a mobile presentation by assessing their screen consumption cost in terms of required pixels, size of fonts, number of characters for text, dimensions of images and similar aspects. Cost of implementation of composition operators are considered as well. The algorithm inserts interactors into a mobile presentation until the total cost of individual interactors and composition operators reaches the maximum global cost supported in a mobile presentation;
- implementation of interactors may change according to new mobile devices resources;
- images are resized maintaining their aspect ratio, when are supported in the target mobile device;
- text is transformed (in case it is too long). Labels are converted with the help of a synonyms database;
- the algorithm aims to predict important regions of a desktop presentation in terms of information content. This issue is addressed considering some attributes associated to the composition operators identified during the reverse engineering process. To this end, we use a database of commonly used Web terms, consider image file names and analyse page tags. Predicting the region most likely to contain important content is useful in order to identify the order in which the mobile presentations should be made available to the users.

The following rules are applied for creating the new connections:

- original connections of desktop presentations are associated to mobile presentations containing the triggering interactor; destination presentation for each of these connections is the first mobile presentation obtained after division of the desktop destination presentation;
- composition operators that are allocated to new mobile presentations, are substituted in the mobile presentation that could not contain them by a link to the new presentation containing the first interactor associated with the composition operator.
- when interactors of a composition operator cannot be contained in only one mobile presentation, then they are distributed in multiple mobile presentations and new connections are generated to navigate through the new series of mobile presentations.

To better understand how semantic redesign works we can consider the example in Figure 2, which shows how a Web page for desktop is adapted to a PDA through semantic redesign transformation.

During the reconstruction of the logical description of the current desktop page, the server (using a database of common used terms in the Web, considering also image file names and analysing tags) aims to understand most likely important parts in terms of page content. Considering the three main groupings of the desktop page in Figure 2, the server identifies grouping 3 as the most likely composition operator to contain more important information.

In fact, the page part represented by *Desc. Title* (highlighted with a dotted line) contains tags such as <h1>, ..,<hn> and an image file, named "logo", denoting that this section could be used to represent the page title. Regions represented by grouping 2 and by grouping 4 contain respectively many links and many image links to internal and external Web resources, so we can deduce that these page parts should not contain important information content for this Web site, but only navigation elements. The region represented by grouping 3 is the only page part containing text interactors, so with a good probability this grouping can be considered the most important content section of page. This hypothesis is also supported by the fact that this text contains some images named "new", thus indicating possible access to the Web site news.

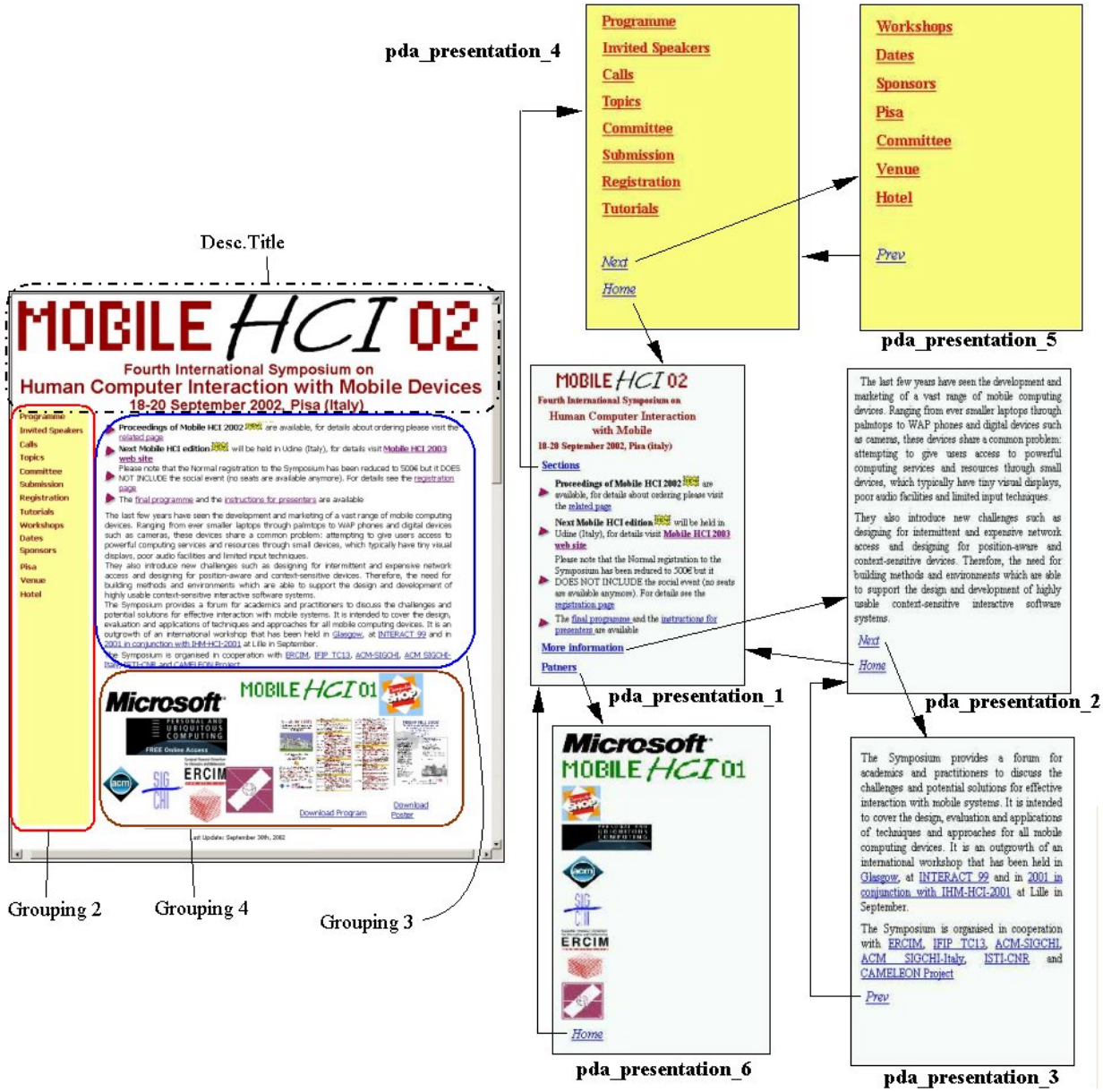

Figure 2: Example of desktop home page and its corresponding PDA pages obtained through semantic redesign algorithm.

5. EXAMPLE OF DYNAMICALLY GENERATED MIGRATORY INTERFACE

In this section we present a usage sample of our system by considering the following scenario. Robert has an article accepted to "Mobile HCI 2002" and he needs to register to the forthcoming conference. He turns on his desktop, accesses the conference site and starts filling in the registration form. After having filled the "telephone" field, an alert on the screen advises him that it is very late and he has to join a meeting regarding budget allocation for the new year. Robert asks for migration and the registration page is transferred to the PDA from which he can continue to fill in the fields exactly from where he left on the desktop, while moving to the meeting room.

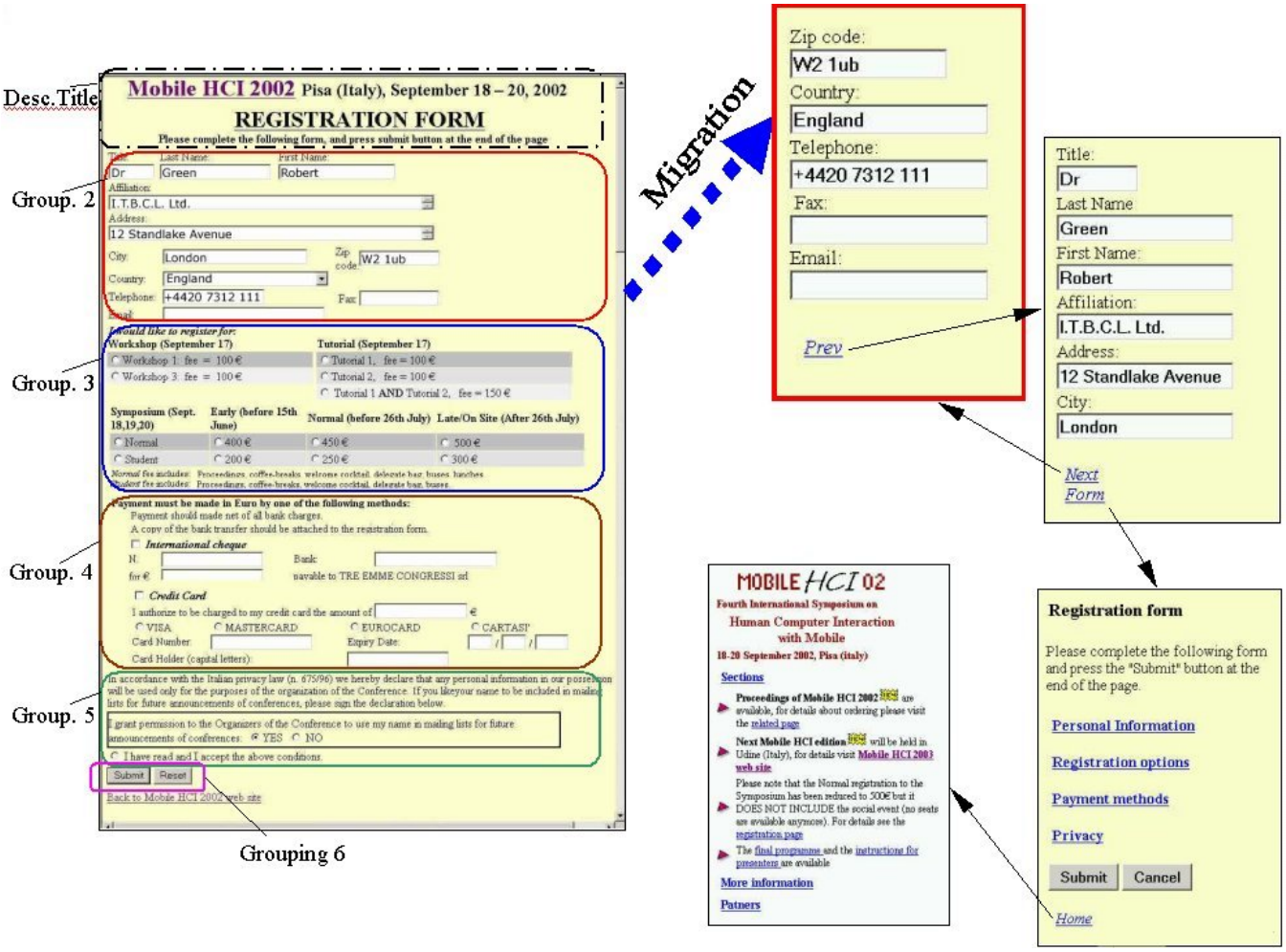

Figure 3: An Example of Dynamically Generated Migratory Interface.

Having completed the registration, Robert comes back to the conference home page to check the programme of the event, to see when his talk is scheduled for. Selecting the "Sections" link he gets the first part of the site menu from which he can finally access the conference programme and see that he will be the first speaker of the second conference day.

The conference site has been designed only for desktop platform. When Robert asks for migration, the migration server recognises that the most suitable target device is a PDA, thus the registration page in the proxy server is stored and reversed into a logical description that is used to split and redesign the page for the PDA. The PDA screen is too small for containing all the interaction objects of the original page, hence the redesign algorithm properly divides it into smaller pages adapting them to the PDA features. Once the redesign phase is completed, the runtime state of the desktop original page is transformed and applied to the new pages. The redesigned page, containing the interaction object corresponding to the last task performed by the user, that is the "Telephone" field, is finally loaded onto the PDA (see Figure 3). The redesigned pages still contain links of unvisited pages to the original desktop version pages of the site. When Robert selects the link for accessing the home page, the proxy server retrieves the desktop version. Then, the reverse engineering and the semantic redesign process are applied again so that, lastly, Robert sees the different sections of the home page on his PDA divided into multiple parts and properly fitting the PDA screen.

6. CONCLUSIONS and FUTURE WORK

We have presented a system able to support user interface migration, where no constraints are given on how the application was originally designed and developed. Migration involves platforms different from the one for which the user interface was originally designed (the desktop), thanks to a reverse engineering/semantic redesign process. Interaction continuity is supported by using logical descriptions that help to associate interactors in the source device with interactors on the target device. A prototype supporting the approach has been implemented and we plan to extend it in such a way as to also consider migration to multi-modal interfaces (for example, with combined use of graphic and vocal interactions).

Future work will be dedicated to extending this approach to support distributing migration where users can move from one device to multiple devices for carrying on their tasks.

7. REFERENCES

[1] Balme, L. Demeure, A., Barralon, N., Coutaz, J., and Calvary, G. CAMELEON-RT: a Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces. In *Proceedings the Second European Symposium on Ambient Intelligence (EUSAI* '04), LNCS 3295, Markopoulos et al. Springer-Verlag, Berlin Heidelberg, 2004, 291-302

[2] Bandelloni, R., Berti, S., and Paternò, F. Mixed-Initiative, Trans-Modal Interface Migration. In *Proceedings of Sixth International Conference on Human Computer Interaction withj Mobile Devices and Services (Mobile HCI'04)* (Glasgow, September 2004), LNCS 3160. Springer-Verlag, 216-227.

[3] Bouillon, L., and Vanderdonckt, J. Retargeting Web Pages to other Computing Platforms. In *Proceedings of IEEE 9th Working Conference on Reverse Engineering* (WCRE'2002) (Richmond, Virginia, 29 October-1 November 2002), IEEE Computer Society Press, Los Alamitos, 2002, 339-348.

[4] Chen, Y., Ma, W.-Y., and Zhang, H.-J. Detecting Web page structure for adaptive viewing on small form factor devices. In *Proceedings **of** the twelfth international conference on World Wide Web* **(***WWW'03)(*May 20-24, 2003, Budapest, Hungary), ACM 1-58113-680-3/03/0005, 225-233.

[5] Coninx, K., and Vandervelpen, C. Towards Model-based Design Support for Distributed User Interfaces. In *Proceedings of the Third Nordic Conference on Human Computer Interaction (NordiCHI'04)*(October23-27, 2004, Tampere, Finland), ISBN:1-58113-857-1, 61-70.

[6] Coninx K., Luyten K., Vandervelpen C., Van den Bergh J., and Creemers B. Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems. In *Proceedings of Fifth International Conference on Human Computer Interaction with Mobile Devices and services (MobileHCI'03)(September, 8-11,2003,Udine, Italy)*, LNCS 2795, ISBN 3-540-40821-5.

[7] Garlan, D., Siewiorek, D., Smailagic, A., and Steenkiste, P. Project Aura: Toward Distraction-Free Pervasive Computing. IEEE Pervasive Computing, Vol 21, No 2 (April-June 2002), 22-31.

[8] MacKay, B., Watters, C. R. and Duffy, J. Web Page Transformation When Switching Devices. In *Proceedings of Sixth International Conference on Human Computer Interaction with Mobile Devices and Services (Mobile HCI'04)* (Glasgow, September 2004), LNCS 3160. Springer-Verlag, 228-239.

[9] Mori, G., Paternò, F., and Santoro, C. Design and Development of Multi-Device User Interfaces through Multiple Logical Descriptions. IEEE Transactions on Software Engineering, August 2004, Vol.30, N.8, IEEE Press, 507-520.

[10] Paganelli, L., and Paternò, F. A Tool for Creating Design Models from Web Site Code. International Journal of Software Engineering and Knowledge Engineering, World Scientific Publishing 13(2), (2003), 169-189.

[11] Paternò, F. Model-based Design and Evaluation of Interactive Applications. Springer Verlag, ISBN 1-85233-155-0, 1999.