Automatic Semantic Platform-dependent Redesign

Giulio Mori & Fabio Paternò

ISTI – C.N.R.
Via Moruzzi, 1 - 56124 Pisa, Italia
{giulio.mori, fabio.paterno}@isti.cnr.it

Abstract

Nowadays, many devices provide access to Web pages: desktops, mobile phones, PDAs, etc.. Often desktop user interfaces need to be redesigned for mobile devices in order to support nomadic access. The problem of adapting the interface to different platforms can be addressed in many ways. Low-level syntactical transcoding or just resizing elements do not seem able to provide general solutions: they often generate poor results in terms of usability because they follow rigid rules and mainly try to fit the same design into different devices. This paper presents our solution, which is based on platform-dependent semantic redesign. Semantic redesign means that transformation from one platform to another is based on the use of semantic information and not only on the analysis of the low-level implementation. In our case, such semantic information is contained in logical descriptions of the user interfaces that also capture the possible tasks users intend to accomplish.

1. Introduction

In recent years there has been an increasing availability in the mass market of various types of interactive devices, in particular mobile devices (UMTS phones, tablet PCs, intelligent watches, just to mention a few). Thus, in order to improve user experience, it is important that many applications be accessible through such a variety of devices. This means that the interface must be able to adapt to the interaction resources actually supported by each device. However, separate development of the user interface software for each potential support is quite expensive. The problem of adapting the interface to different platforms can be addressed in many ways. There are automatic tools that mainly translate from one implementation language to another. This type of low-level syntactical transcoding often provides poor results in terms of usability because it follows rigid rules and mainly tries to fit the same design into different devices. We do not believe that just resizing elements is sufficient for obtaining general solutions. The more the environment is able to ascend hierarchy of interface abstraction levels, the more substantial will be the possible modification that can be performed taking into account the characteristics of the other target devices. If the environment is able to identify the concrete object associated with the current user interface element, then it is possible to represent that specific object in a way tailored to the new device, whereas if it is able to identify the corresponding abstract object (a modality-independent description), then the environment can change the choice of the interaction technique implementation depending on the characteristics of the new device. However, if the environment is also able to understand the corresponding task then it can reason at this level as well and make a wider set of decisions: the task at hand can still be performed, but with different modalities (different user interface elements or domain elements or even a different task structure with different subtasks associated with the same main task) or it can decide that in the new context of use the original task no longer has importance. It is difficult to perform this level of reasoning completely automatically. This has raised interest in research results in model-based approaches for interactive applications [2], [5], [6], [10], which provide declarative descriptions of the user interface and tool support to generate the corresponding implementation taking into account the features of the target devices. This approach has also been adopted in new W3C standards (such as XForms where the same logical interaction can be rendered differently according to the platform). Thus, providing pervasive usable services calls for tools able to exploit such possibilities. One example is TERESA [8], which is an authoring tool that provides support for various model-based methods following a top-down approach. In this framework, herein we present a new solution in order to support automatic semantic platform-dependent redesign. This is useful, for example, when a desktop version exists or is the first to be implemented and then designers would like to obtain a version for mobile devices, which is able to adapt to their features while reusing elements already generated for the desktop version. By platform-dependent redesign we mean the possibility of changing the design for an interactive platform in order to adapt to a new one. A platform is a set of devices that have similar interaction resources (for example, the desktop, the PDA, the vocal device). Semantic redesign means that this transformation is based on the use of semantic information and not only on the analysis of the low-level implementation. In our case, the semantic information is in the logical descriptions of the user interface at various abstraction levels. Different approaches to transcoding are possible [7]. Some work aiming at supporting redesign of desktop user interfaces has started to appear, such as Vaquita [4] and its evolutions, but we aim to provide a more general solution able to consider a wider set of semantic information, including user tasks. To this end we have developed a solution that is able to support all such possible abstraction levels differently from other approaches, such as UIML [1], which are mainly limited to the concrete level.

In the paper we first introduce the type of logical user interface descriptions we use to analyse user interfaces and their potential redesign, without having to manage many low-level implementation details. Then, we introduce semantic redesign and show how it can be used in various design processes. These elements are exploited in our transformation, whose main rules are first introduced in general terms and then discussed through a specific example and a further case study. Lastly, we draw some conclusions and provide indications for future work.

2. Logical description of the user interface

The environment that we propose is able to support two device-independent languages: one (ConcurTaskTrees [9]) is used to describe the tasks that users intend to accomplish and the objects manipulated for this purpose, the other one shifts the focus to the user interface, but still in an abstract, modality-independent manner. Then, for each platform there is a concrete language. The concrete description is mainly a platform-dependent refinement of the abstract interface. Its purpose is to create a link between the abstract description and the implementation languages for a given platform. We can generate such logical descriptions and manipulate them with the support of semantic information in order to obtain the versions for mobile devices in two manners:

- through a reverse engineering transformation of desktop Web pages, which can be performed by a *reverse/redesign proxy server* [3]; the created logical descriptions can be transformed in order to obtain the logical descriptions first and then the implementation for mobile devices.
- through the TERESA tool; starting with a task model for a desktop platform, the tool supports the transformation for redesigning desktop logical descriptions for mobile devices. This feature is useful for developers who want to quickly generate a mobile interface version.

An abstract user interface is structured into presentations and connections indicating how it is possible to move from one presentation to another. Each presentation is structured into interactors (logical interaction objects) and composition operators. We have defined a number of composition operators, which aim to capture communication effects that often designers want to achieve when they structure their user interfaces. The purpose of the composition operators is to indicate how to put together interactors. Each composition operator is associated with a communication goal. Depending on such goals, different implementation techniques will be used to support the composition operator. Figure 1 shows an example of a Web page taken from a frequently accessed Web site. We can note how the designer used various techniques to highlight groups of related interface elements. On the top there are elements that are ordered according to the potential user interest. Some elements are grouped using implementation techniques such as same background, same structure, bullets and so on. There are elements that are related to the rest of the Web site, such as the search element. Other elements are highlighted using large image and fonts because they are considered important.
In general, the composition operators can involve one or two expressions, each of which can be composed of one or several interactors or even compositions of interactors. In addition, their definition is modality-independent. They are:

- Grouping (G): indicates a set of interface elements logically connected to each other;
- Relation (R): highlights a relation (usually one-to-many) among some elements; one element has some effects on a set of elements;
- Ordering (O): some kind of ordering among a set of elements can be highlighted;
- Hierarchy (H): different levels of importance can be defined among a set of elements.

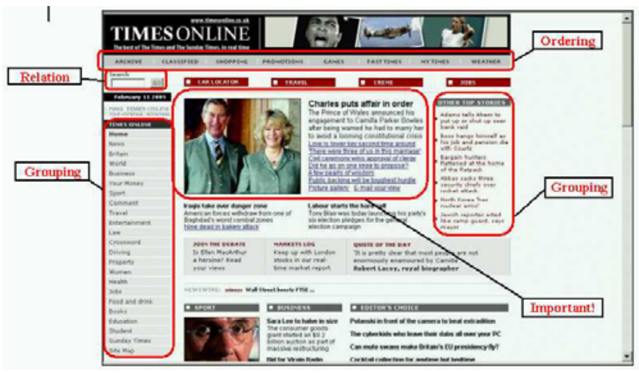


Figure 1: Web page with indication of some associated communication goals.

3. Semantic redesign

In our environment semantic redesign can be applied in three design processes. In one case the designer uses our environment to create the abstract user interfaces first (8 of Figure 2), then the corresponding concrete one and lastly the final interface for the desktop system (step 2 and 1). Then, the abstract and concrete descriptions are again considered as input for the redesign module, which can produce the new interface and corresponding logical descriptions.

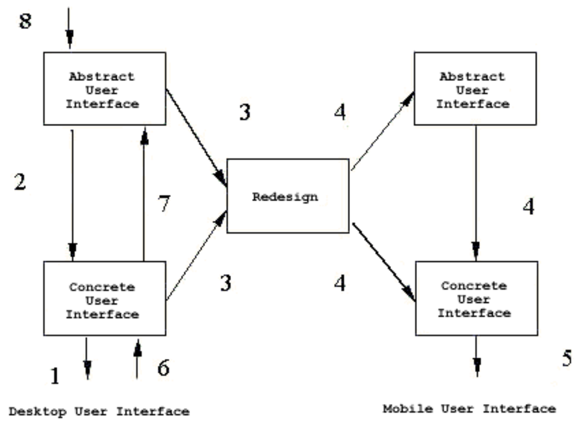


Figure 2: Semantic redesign with forward and reverse engineering.

This can also be obtained through a process whereby there is an existing desktop version and a reverse engineering transformation is applied (step 6 and 7) to derive the corresponding concrete and abstract user interfaces. Subsequently, these are input (step 3) to the redesign module, which generates (step 4) the abstract and concrete descriptions (and their mappings) for the mobile interface. These are then used for generating the final corresponding user interface.

A variant of this solution is represented in Figure 3. The difference in this case is that the redesign module also receives information from the nomadic task model as input. A nomadic task model is a model that indicates the platforms suitable for supporting each task. Thus, filtering only those tasks suitable for a given platform on a nomadic task model generates the task model for that specific platform. This means that this variant of the redesign module also receives information regarding which platforms are suitable to support a given task. Thus, if a task is considered appropriate for a desktop system but not for a mobile device, because of its more limited interaction resources, then the interactors corresponding to such task will not be created in the logical description of the mobile interface and, consequently, will be lacking in its implementation.

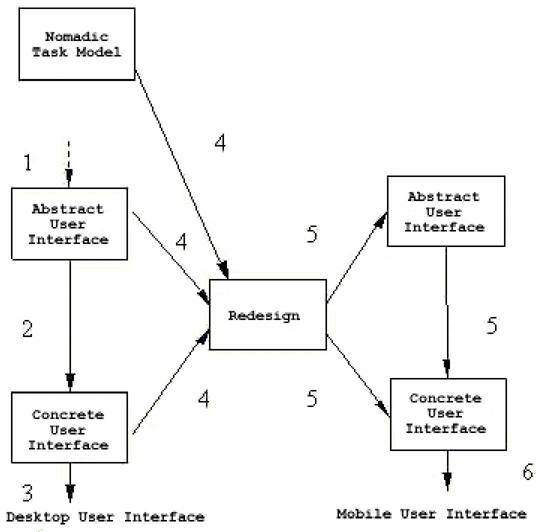


Figure 3: Semantic redesign with nomadic task model support.

4. The transformation supporting semantic platform redesign

In this section we discuss how our platform-dependent redesign transformation works considering the concrete /abstract description of the user interface. Given the limited resources in screen size of mobile devices, desktop presentations generally must be split into a number of different presentations for the mobile devices. The logical levels provide us with some semantic information that can be useful for identifying meaningful ways to split the desktop presentations along with the user interface state information (the actual implemented elements, such as labels, images, …). The redesign module analyses the input from the logical descriptions and generates an abstract and concrete description for the mobile device from which it is possible to automatically obtain the corresponding user interfaces. The redesign module also decides how abstract interactors and composition operators should be implemented in the target mobile platform.

In order to automatically redesign a desktop presentation for a mobile presentation, we need to consider semantic information and the limits of the available resources. If we only consider the physical limitations we may end up dividing large pages into small pages that are not meaningful. To avoid this, we also consider the composition operators indicated in the logical descriptions. To this end, our algorithm tries to maintain interactors that are composed through some operator at the conceptual level in the same page, thus preserving the designer's communication goals. However, this is not always possible because of the limitations of the target platform. In this case, the algorithm aims (when possible) to equally distribute the interactors into mobile device presentations. In addition, splitting the pages requires a change in the navigation structure with the need for additional navigator interactors that allow access to the newly created pages. More specifically, the transformation follows these main criteria:

- The presentation split from desktop to mobile takes into account the composition operators because they indicate semantic relations among the elements that should be preserved in the resulting mobile interface. Another aspect considered is the number and cost of interactors. The cost is related to the interaction resources consumed, so it depends on pixels required, size of the fonts and similar aspects.
- The implementation of the logical interactor may change according to the interaction resources available in the target platform.
- The connections of the resulting interface should include the original ones and add those derived from the presentation split.
- The images should be resized according to the screen size of the target devices keeping the same aspect ratio. In some cases they may not be rendered at all because the result is too small or the mobile device does not support them.
- Text and labels can be transformed as well, because they may be too long for the mobile devices. In converting labels we use tables able to identify shorter synonyms.

In particular, regarding the creation of new connections the following rules are applied:

- original connections of desktop presentations are associated to the mobile presentations that contain the interactor triggering the transition. The destination for each of these connections is the first mobile presentation obtained by splitting the original desktop destination presentation;

- composition operators that are allocated to a new mobile presentation are substituted in the original presentation by a link to the new presentation containing the first interactor associated with the composition operators.
- when a set of interactors composed through a specific operator has been split into multiple presentations because they do not fit into a single mobile presentation, then we need to introduce new connections to navigate through the new series of mobile presentations.

In the transformation process we take into account semantic aspects and the cost in terms of interaction resources of the elements considered. In an early version we attempted to define the maximum number of interactors that can be used in a mobile presentation. However, this proved to be too rigid as different interactors have varying screen space and interaction resource requirements. So, we decided to define for each mobile device class identified (large, medium or small) a maximum acceptable overall cost in terms of the interaction resources utilizable in a single presentation. So in this approach, each interactor and (even each composition operator) has a different cost in terms of interaction resources.

The algorithm inserts interactors into a mobile presentation until the sum of individual interactor and composition operator costs reaches the maximum global cost supported. Examples of elements that determine the cost of interactors are the font size (in pixels) and number of characters in a text, image size (in pixels), if present. One example of the costs associated with composition operators is the minimum additional space (in pixels) needed to contain all its interactors in a readable layout. This additional value depends on the way the composition operator is implemented (for example, if a grouping is implemented with a fieldset or with bullets). Another example is the minimum and maximum interspace (in pixels) between the composed interactors;
After such considerations, it is easy to understand that each mobile presentation could contain a varying number of interactors depending on their interaction resources consumption.

5. A small example

In order to explain the transformation, we can consider a specific example of a desktop Web site and see how one of its pages (see Figure 5) can be transformed using our method.

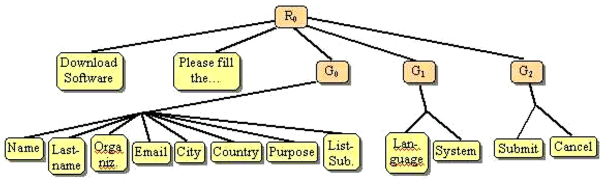
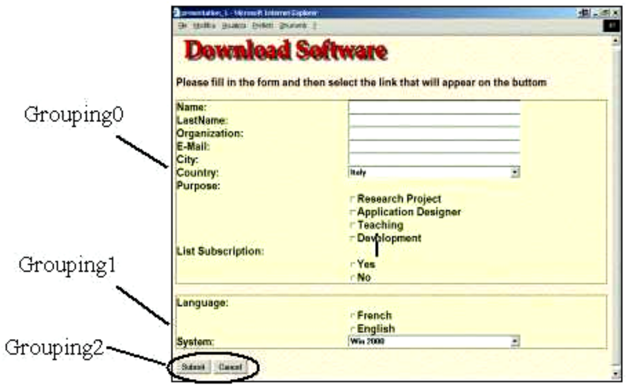


Figure 4: CUI DOM tree-structure of interface in fig 5.

The automatic transformation starts with the XML specification of the Concrete Desktop User Interface and creates the corresponding DOM tree-structure (Figure 4). The concrete user interface description contains interactors (such as text, image, text_edit, single_choice, multiple_choice, control, etc) and composition operators (grouping, ordering, hierarchy or relation) which define how to structure them. A composition operator can contain other interactors as well as other composition operators.
In the example, there is a *relation* operator, which involves all the elements of the page: the elementary description interactor "Download Software", the elementary text interactor "Please fill in the form…" and the elements made up of three grouping operators. In general, the relation operator identifies an association between the last element and all the other elements involved in the operator. In this case, the last element is represented by the composition operator G_2, which groups the "Submit" and "Cancel" buttons. Indeed, they are related to all the remaining content of the Web page because they allow to transmit such information to the server. There are also two *grouping* operators (G_0 and G_1) implemented by the two fieldsets in the user interface in Figure 5.

Overall, this desktop presentation contains 14 interactors, which require a large amount of interaction resources; too great to be contained in a single mobile phone presentation, even a large one (such as a smartphone). Our transformation divides the "desktop_Download" presentation of the example into four presentations for mobile devices. Considering the tree structure of the XML specification of the Concrete User Interface, the algorithm makes a depth first visit starting with the root, and generates the mobile presentations by inserting the elements in each level as long as the sum of the cost of each interactor (and related composition operators) is lower than the maximum value supported by a mobile presentation.



Download Software

Please fill in the form and then select the link that will appear on the buttom

Name:
LastName:
Organization:
E-Mail:
City:
Country: Italy
Purpose:
Research Project
Application Designer
Teaching
Devlopment
List Subscription:
Yes
No

Language:
French
English
System: Win 2000

Submit Cancel

Figure 5: Example of desktop Web user interface.

Each composition operator (with its associated elements) that cannot fit in the presentation (in the example G_0 and G_1) is substituted by a link pointing to a mobile presentation containing their first elements. In this case, the two new links point to the mobile presentations containing the first element of G_0 (i.e., "Name") and the first element of G_1 (i.e., "Language"). So, looking at the example, the algorithm begins to insert elements in the first mobile presentation, and when it finds a composition operator (such as G_0), it starts to generate a new mobile presentation with its elements. Continuing the visit, the algorithm explores the composition operator G_0, which has 8 elements, all of which cannot fit in a single new presentation. Thus, two mobile presentations are created and the algorithm distributes the elements equally between them. The depth first visit of the tree continues and reaches G_1. The algorithm inserts in the main mobile presentation a corresponding link, pointing to the newly generated mobile presentation where the elements of G_1 are inserted.
The relation operator captures a many-to-one association. The latter element of such a Relation must be contained in its entirety in the same presentation as the other elements of the same Relation because it is the fundamental element defining the association. Even when the latter element is another composition of elements (such as G_2), it is completely inserted into the presentation (cost permitting).
The XML specifications of the concrete and abstract interfaces also contain tags for connections (elementary connections or complex connections). An elementary connection permits moving from one presentation to another and is triggered by a single interactor. A complex connection is triggered when a boolean condition related to multiple interactors is satisfied. The transformation creates new connections among the presentations for the mobile phone following the rules introduced in the previous section. One rule indicates that composition operators that are substituted by a link introduce new connections to presentations containing the first interactor associated with the composition operators. In the example (Figure 6), we have two new links "Form – Part 1" and "Form – Part2", which support access to the pages associated with the first interactor of G_0 and the first interactor of G_1 respectively. Another rule indicates that when a set of interactors composed through a specific operator has been

split into multiple presentations we need to introduce new connections to navigate through the new mobile presentations. In the example "*previous*" and "*next*" links have been introduced automatically by the environment. These connections are useful to navigate among presentations that are derived from the splitting of the G_0 elements. There is also the need for identifying the connections for going back from the new generated presentations to those containing the links to them. In the example, we have the "Form – Part1" and the "Form – Part2" links contained in the first mobile presentation. Thus, we need two corresponding "*home*" links that allow going back to it (Figure 6).

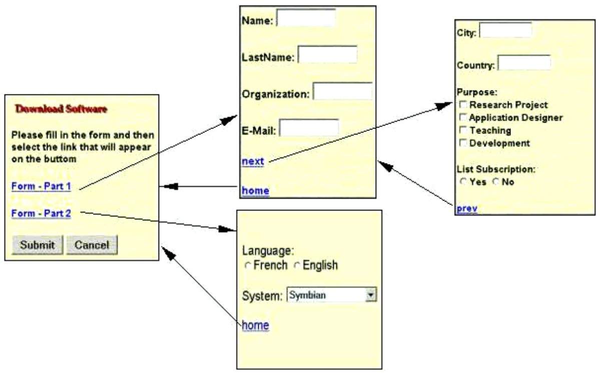


Figure 6: Mobile pages resulting from the transformation of the example desktop page.

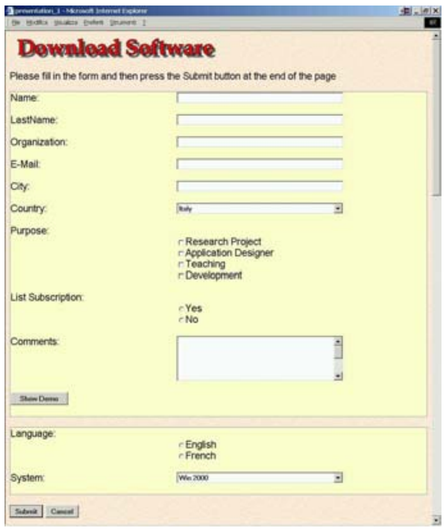
6. Task-based semantic redesign

We can now analyse a variant of semantic redesign exploiting knowledge of the tasks involved. This variant is useful when there are interactors supporting tasks considered suitable for a desktop system but not for a mobile device, which should therefore be removed from mobile presentations because of its limited interaction resources. This variant also calls for platform task information for successful completion.

Let us consider a desktop user interface (Figure 7) for subscribing to a service; it is composed of two presentations. In this case, five tasks: "Select Country", "Choice Purpose", "Select List Subscription", "Enter Comments" and "Show multimedia Demo" are supported by the desktop presentation but may not be suitable for a mobile platform.

Thus, when mobile presentations are produced via redesign of desktop presentations by analyzing the corresponding tasks, the interactors associated with these five tasks will not be inserted in the mobile Concrete User Interface generated. Conceptually, these five tasks are unsuitable for the mobile platform (filling in a form through a mobile device should require a minimal amount of input) and, in addition, tasks such as "Enter Comments" and "Show multimedia Demo" need a lot of space and multimedia resources for presentation (not present in most mobile phones).

This type of analysis can produce more suitable and simpler mobile presentations and is triggered when the "Task Semantic Redesign" transformation is selected. To this end, the transformation considers the type of target mobile device (classified into the three aforementioned categories) and the associated task model to be able to identify which tasks are to be supported. Figure 8 shows the results of task-based semantic redesign transformation for this example.


Download Software

Please fill in the form and then press the Submit button at the end of the page

Name:
LastName:
Organization:
E-Mail:
City:
Country: Italy
Purpose:
Research Project
Application Designer
Teaching
Development
List Subscription:
Yes
No
Comments:
Show Demo
Language:
English
French
System: Win 2000
Submit Cancel

Figure 7: Example of desktop presentation.

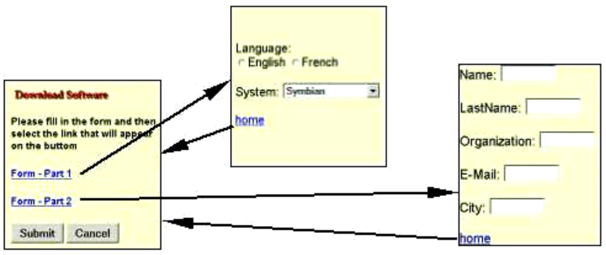


Figure 8: Mobile presentations after task–based semantic redesign.

7. A case study

Now we address another, more complex example of redesign of a desktop interface from the tourism Web site of the French towns of Sedan and Bouillon, which has been considered in the CAMELEON project. Managed by the Tourist bureaus, this Web site promotes tourism in the area. In order to help tourists prepare their visits, the Web site aims to gather all the necessary information about Hotels, Restaurants, Camping sites, Lodgings, Museums and other interesting initiatives held during different periods of the year such as "the castles treasure hunt", "ballet of the raptors" and night visits. Documentation (books, maps, etc.) and detailed information about visits (individual or in groups) is also available for download by filling out the contact form (Figure 9). Without going into the details of the DOM structure of the XML-based logical description, Figure 9 shows four parts (corresponding to four grouping 0,1,2 and 3), contained and grouped together in the whole desktop page (corresponding to a top level grouping). In the transformation, the algorithm produces five mobile presentations (Figure 10) in which we have the home page *mobile_presentation1* containing three links (*Sections*, *Form* and *Contact*) pointing to mobile presentations containing respectively elements of groupings 0, 2 and 3. In *mobile_presentation1* the three images of *Fort de Sedan castle*, *Map of Sedan – Bouillon* and *Guide of Pays Sedanais*, have been removed because the mobile devices considered support checkbox with only text choices. Interactors belonging to grouping 2 occupy too much space to be contained in a single mobile presentation, so they are distributed in *mobile_presentation3* and *mobile_presentation4*. *Mobile_presentation2* thus not contain the image of *Pass Muraille* (corresponding to a graphical link interactor in the desktop platform), because we have addressed mobile devices supporting only text or button links.

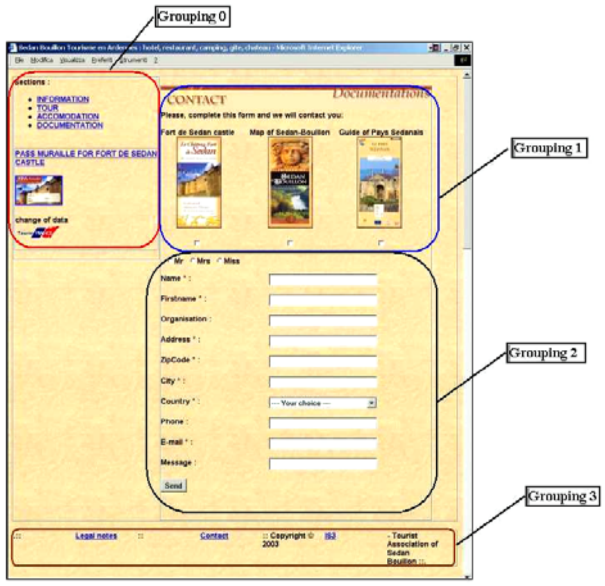


Figure 9: A desktop presentation in the case study.

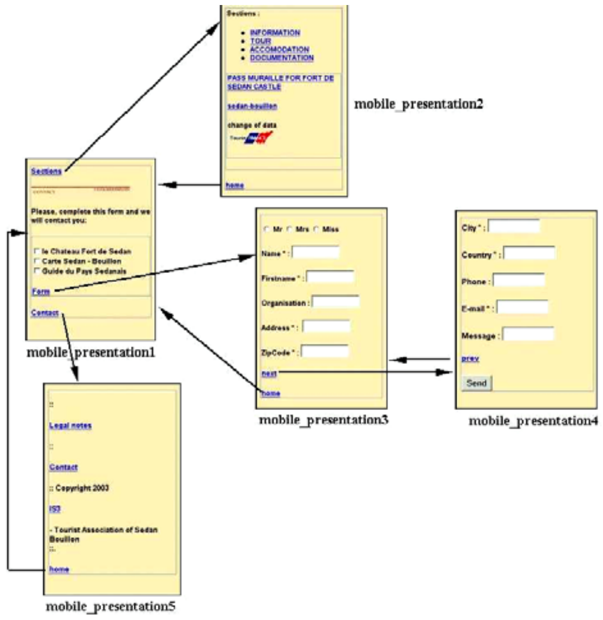


Figure 10 : The resulting mobile presentations.

8. Conclusions

We have discussed how to support semantic platform redesign. Examples of application of the approach proposed have been reported. Such transformation can be used at both design and run-time. In the case of use at design time, we have indicated three different design processes that can benefit from its main features. This is particularly useful for developers who need a version for mobile devices and would like to have some support in order to facilitate the process and still obtain meaningful results. At run-time, the transformation is integrated in a server, which recognizes the type of platform accessing the Web site and, in the event it is a mobile device, transforms the page to adapt it to the new platform.

Currently, our approach works for Web pages that have been obtained using model-based environments able to first create their logical descriptions and then the corresponding implementations or with a server able to take the desktop interface and generate the corresponding version for mobile devices exploiting logical descriptions automatically created using reverse engineering techniques.

In the implementation of our transformations there are still some issues to be resolved when reversing and redesigning Web pages containing particular dynamic elements (such as Flash animations, banners, etc.) or in some cases where complex layout of pages are dynamically generated (i.e.: using combination of servlets, jsp, etc.) or when the page layout is poorly managed (i.e. with nested tables used for layout or extensive use of frames).

References

[1] Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S. and Shuster, J. (1999) *UIML: An Appliance-Independent XML User Interface Language*, in *Proc. EighthWWWConf.*

[2] Banavar, G., Bergman L., D., Gaeremynck, Y., Soroker, D. and Sussman, J. (2004) *Tooling and system support for authoring multi-device applications*, in *The Journal of Systems and Software 69*. p.227–242.

[3] Bandelloni, R., Mori G., and Paternò, F. (2005) *Dynamic Generation of Migratory Interfaces*, in *Proceedings MobileHCI05*, Springer Verlag, Salzburg.

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

[5] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. (2003) *A Unifying Reference Framework for Multi-target User interfaces*, in *Interacting with Computers* Vol. 15/3, p. 289-308, Elsevier.

[6] Calvary, G., Coutaz, J. and Thevenin, D. (2001) *A Unifying Reference Framework for the Development of Plastic User Interfaces*, in *IFIP WG2.7 (13.2) Working Conference, EHCI01*, May, Springer Verlag Publ., LNCS 2254, M. Reed Little, L. Nigay Eds, Toronto pp.173-192.

[7] Menkhaus, G. and Fischmeister, S. (2003) *Evaluation of User Interface Transcoding Systems*, in *Proc. Seventh World Multiconf. Systemics, Cybernetics and Informatics*.

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

[9] Paternò, F. (1999) *Model-Based Design and Evaluation of Interactive Applications*, in *Springer Verlag, ISBN* 1-85233-155-0.

[10] Szekely, P. (1996) *Retrospective and Challenges for Model-Based Interface Development*, in *2nd International Workshop on Computer-Aided Design of User Interfaces*, Namur University Press, Namur.