Supporting Flexible Development of Multi-Device Interfaces

Francesco Correani, Giulio Mori, Fabio Paternò

ISTI-CNR
56124 Pisa, Italy
{francesco.correani, giulio.mori, fabio.paterno}@isti.cnr.it
http://giove.isti.cnr.it

Abstract. Tools based on the use of multiple abstraction levels have shown to be a useful solution for developing multi-device interfaces. To obtain general solutions in this area it is important to provide flexible environments with multiple entry points and support for redesigning existing interfaces for different platforms. In general, a one-shot approach can be too limiting. This paper shows how it is possible to support a flexible development cycle with entry points at various abstraction levels and the ability to change the underlying design at intermediate stages. It also shows how redesign from desktop to mobile platforms can be obtained. Such features have recently been implemented in a new version of the TERESA tool.

1 Introduction

Model-based approaches [10, 13] have long been considered for providing support to user interface design and development. Recently, such approaches have received further attention because of the challenges raised by multi-device environments [1, 4, 6, 13]. The use of tools based on logical abstractions enables adapting the interfaces under development to the characteristics of the target devices. This can simplify the work of designers who do not have to address a proliferation of devices and related implementation details.

The potential logical descriptions to consider are well identified, and their distinctions are clear [3]: task models represent the logical activities to perform in order to reach users' goals; object models describe the objects that should be manipulated during task performance; abstract user interfaces provide a modality independent description of the user interface in terms of main components and logical interactors; concrete user interfaces provide a platform-dependent description identifying the concrete interaction techniques adopted, and lastly the user interface implements all the foregoing.

Various approaches have benefited from this logical framework, and tools supporting it have started to appear. In particular, there are tools that implement a forward engineering approach, which take an abstract description and generate more

refined ones until the implementation is obtained; or tools supporting reverse engineering approaches, which instead take an implementation and aim to obtain a corresponding logical description. Examples of forward engineering tools are Mobi-D [13] and TERESA [6]. They both start with task models and are able to support user interface generation, though by applying different rules and additional models. TERESA is the tool for the design of multi-device interfaces developed in the EU IST CAMELEON project. It introduces the additional possibility of adapting the transformation process to the platform considered. A platform is a set of devices that share a similar set of interaction resources. Another example of tool for forward engineering is ARTstudio [4], which also starts with the task model and supports the editing of abstract and concrete user interface, but, contrary to TERESA, it generates Java code instead of Web pages and is not publicly available. Examples of different support for reverse engineering are Vaquita [2] and WebRevEnge [8]. The first one provides the possibility of rebuilding the concrete description of Web pages, whereas the latter reconstructs the task model corresponding to the Web site considered. In both cases one limitation is the lack of support for the reverse engineering of Web sites implemented using dynamic pages.

The needs and background of software developers and designers can vary considerably, and there is a need for more flexible tools able to support various transformations in the logical framework mentioned. To this end, we have designed and implemented a new version of the TERESA tool, aiming to provide new possibilities with respect to the original version [6]. In particular, the new version that is presented in this paper supports multiple entry points in the development process and the redesign of a user interface for a different platform.

In the paper we first recall the basic design criteria of the original version of the TERESA tool and then we dedicate one section to describing how multiple entry points can be supported and one for the transformation for redesign from desktop to mobile. We then show examples of applications of such new features and, lastly, we draw some conclusions and indications for future work.

2 The initial TERESA environment

The TERESA tool was originally designed to support the development of multi-device interfaces starting with the description of the corresponding task model. In order to facilitate such a development process the main functionality of the CTTE tool [7], supporting editing, analysis, and interactive simulation of task models, have been integrated into the new tool. So, once designers have obtained a satisfying task model, they can immediately change mode and use it to start the generation process. The tool provides automatic transformation of the task model into an abstract user interface structured into presentations. For each presentation, the tool identifies the associated logical interactors [11] and provides declarative indications of how such interactors should be composed. This is obtained through composition operators that have been defined taking into account the type of communication effects that designers aim to achieve when they create a presentation [8].

The composition operators identified are:

- Grouping (G): indicates a set of interface elements logically connected to each other;
- Relation (R): highlights a one-to-many relation 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.

In addition, navigation through the presentations is defined taking into account the temporal relations specified among tasks. The abstract user interface description can then be refined into a concrete user interface description, whereby a specific implementation technique and a set of attributes are identified for each interactor and composition operator, after which the user interface implementation can be generated. Currently, the tool supports implementations in XHTML, XHTML mobile device, and VoiceXML (one version for multimodal user interfaces in X+V and one version for graphical direct manipulation interfaces are under development).

3 Support for Flexible Forward Engineering

Interface design is complex. Often, as designers go through the various steps in order to develop suitable solutions for the current abstraction level, they would like to reconsider some of the choices made earlier in an iterative process. Furthermore, the actual results of automatic transformations may not be precisely those expected and thus would need to be refined. Lastly, the need to provide relevant support to a flexible methodology requires the ability to offer different entry points.

The original version of the TERESA tool provided a concrete solution to the issue of supporting development of multi-device interfaces through various levels of automation. However, when designers selected the completely automatic solution sometimes it happened that what they get was rather different from what they wanted (Figure 1 shows an example [12]). Thus, there was a need for providing designers with better support for tailoring the transformations to their needs.

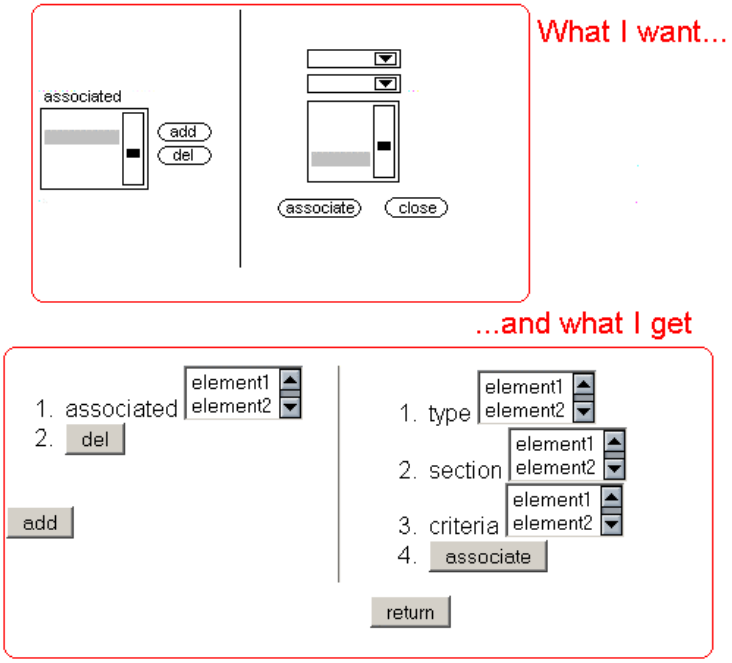

Fig. 1. Example of mismatch between designer's goals and result of automatic generation.

Once a suitable description of the abstract user interface has been obtained from a given task model, it is important that its properties be adjusted to increase usability for the generated presentations. Designers may also decide to start defining the abstract interface from scratch, bypassing the task modelling phase.

In order to deal with all these issues we decided to extend TERESA functionalities by adding new features, in particular, enabling changes, even radical ones, in the properties of abstract user interface elements and the ability to develop an abstract user interface from scratch.

Once an abstract user interface has been created, there are various levels of modifications that can be possible:

- *Modifying the structure of a presentation without changing the associated interactors*. This can be performed in different ways: moving the orders of the interactors within a composition operator, changing, adding or removing composition operators;
- *Modifying the association between interactors and presentations without changing existing interactors*. This can be performed by merging or splitting existing presentations or moving one interactor from one presentation to another.
- *Modifying the set of available interactors*, this means changing the type of interactors, adding or removing interactors (this can be done by either working on single interactors or adding or removing groups of interactors or entire presentations).

In order to avoid confusing designers the editing features have to be explicitly enabled. Then, to ease the use of these functionalities, a number of features have been introduced. The type of an interactor is explicitly represented through an icon (as are the task categories in the task model) and modifications to the interactors order within a presentation can be performed through a drag and drop function. The result of a completely automatic transformation from the task model to the abstract user interface is a set of presentations (which are listed on the left side of the control panel, see Figure 2) and the related connections defining navigation through them. When one presentation is selected then its logical structure in terms of interactors and composition operators is shown in the central part. Designers can select either composition operators or interactors and the corresponding attributes are shown in the bottom part. The position of an interactor in the presentation can be moved through drag and drop interactions. If editing has been enabled it is also possible to change the type of operators and interactors. For example, in Figure 2 there is a change of a Grouping operator.

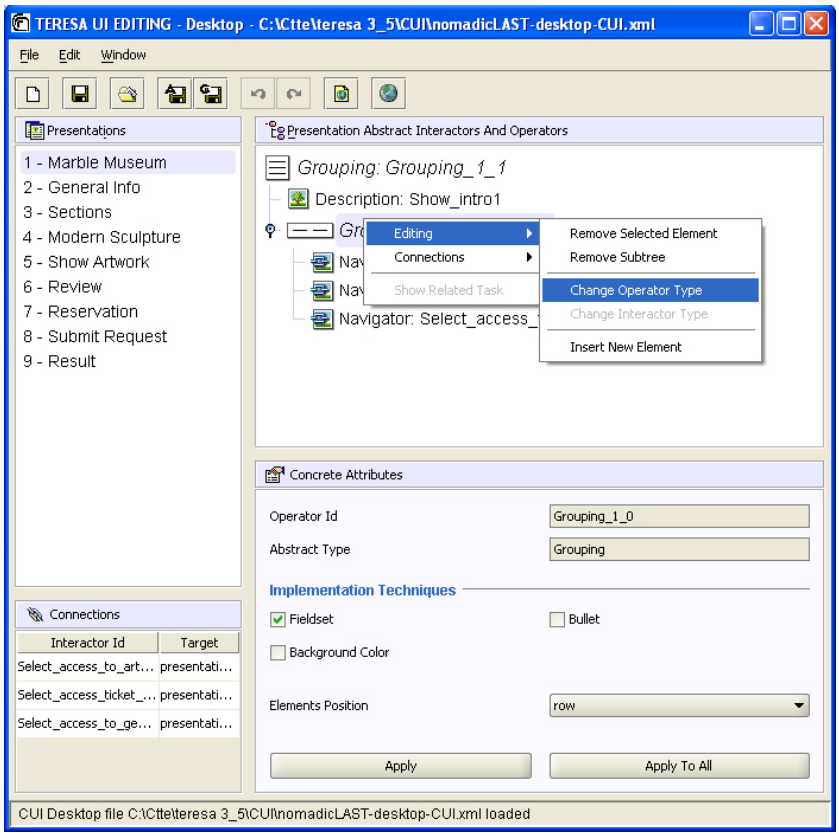

Fig. 2. Example of change of composition operator.

The editor of the abstract user interface (see Figure 3) provides designers with a view on various aspects that can be modified. One panel indicates the list of presentations defined so far. The logical structure of the currently selected presentation is shown as well. It can be represented either showing the logical structure in a tree-like manner or through the list of the elements composing it. The concrete aspects of the currently selected interactor are displayed in a separate panel. For example, in the figure a navigator interactor has been selected and its identifier, type, concrete implementation (in this case through a graphical link) and related attributes (in this case the image) are shown in the associated panel. Even the navigation through the various presentations is represented and can be edited: it is defined by a list of connections, each one defined by the interactor that triggers the change and the target presentation. The tool also provides the possibility of showing the corresponding XML-based specification and the logs of the designer interactions with the tool.

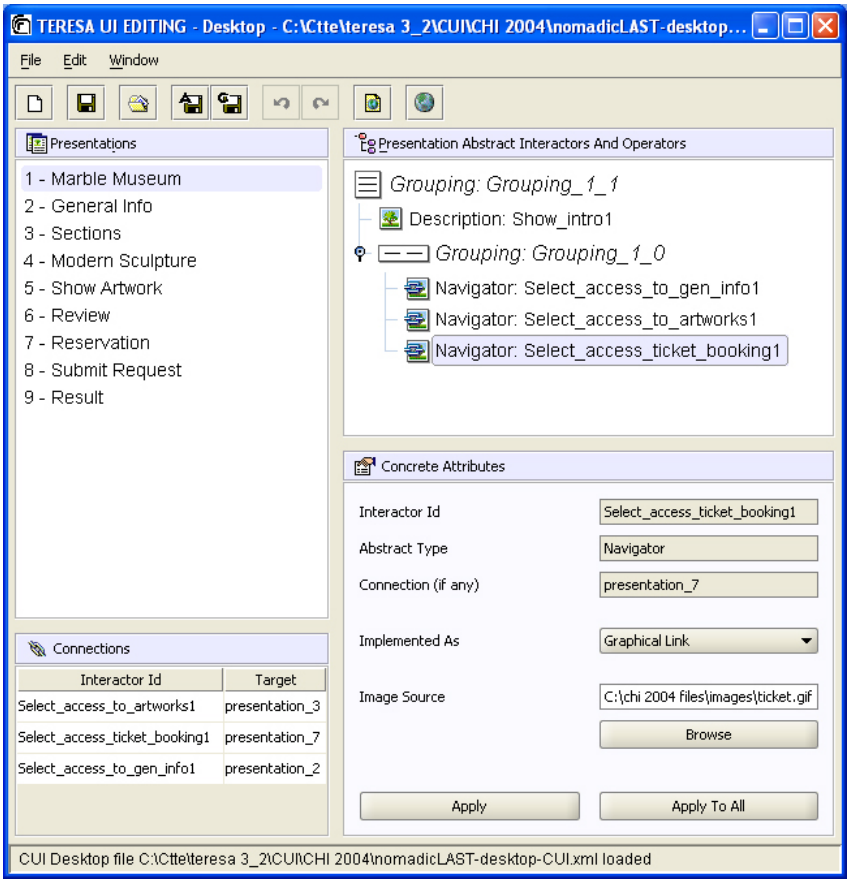

Fig. 3. Tool support for editing the abstract and the concrete user interface.

Lastly, a preview of the associated interface can be provided in order to allow designers to get a more precise idea of the resulting interface. Figure 4 shows the interface corresponding to the abstract/concrete presentation in Figure 3. Three navigator interactors are implemented through graphical links to other points in the application, and are grouped on the same row. In turn, this group is included in an additional group arranged vertically together with a description element that is implemented through images and text.

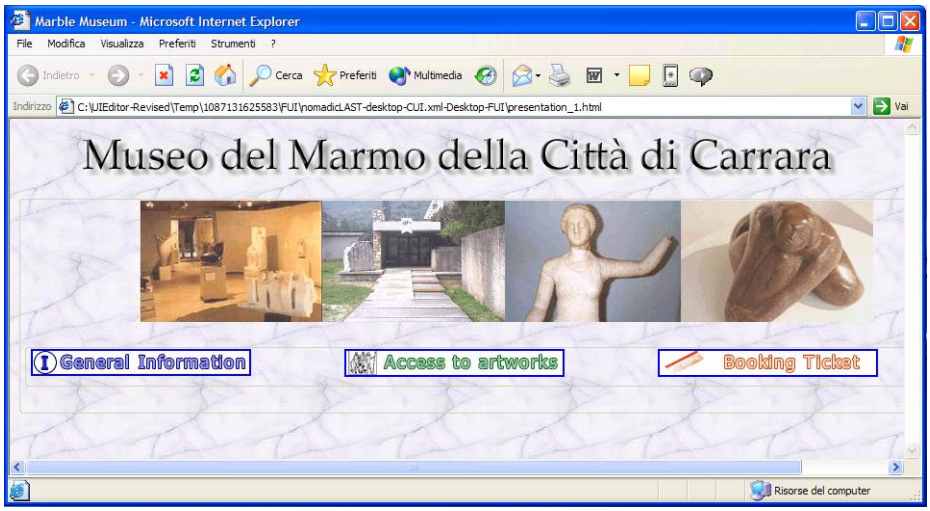

Fig. 4. The user interface corresponding to the concrete interface obtained through preview.

4 Support for Redesign

Nowadays many devices provide access to Web pages: computers, mobile phones, PDAs, etc.. Often there is a need for redesigning the user interface of an application for desktop systems into a user interface for a mobile device. Some authors call this type of transformation graceful degradation [5]. One main difference between such platforms is the dimension of the screen (a mobile phone cannot support as many widgets as a desktop computer in a presentation), so the same page will be displayed differently or through a different number of pages on different devices. Transcoding techniques (such as those from HTML to WML) are usually based on syntactical analysis and transformations, thus producing results which are poor in terms of usability because they tend to propose the same design in devices with different possibilities in terms of interaction resources.

In this section we describe the solution adopted to transform pages written for a desktop computer into pages for a mobile phone. In our transformation we have

classified the type of mobile phones based on the screen size and other parameters, which determine the number of widgets that can be supported in a presentation. We thus group such devices into three categories: large, medium or small. In the transformation we consider that a Web page for a specific device can display a limited number of interactors [11] that depends on the type of platform. Obviously, the number of interactors supported in a desktop presentation will be greater than the number of interactors contained in a mobile phone presentation, so a desktop Web presentation will be divided into many mobile phone presentations to still support interactions with all the original interactors.

In our transformation we consider the user interface at the concrete level. This provides 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, …). We also consider some information from the abstract level (see Figure 5): in particular the abstract level indicates what type of interactors and composition operators are in the presentation analysed. The redesign module analyses such inputs 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. Thus, settings and attributes should change consequently depending on the platform. For example, a grouping operator can be represented by a field set in a desktop page but not in a page for a small mobile phone.

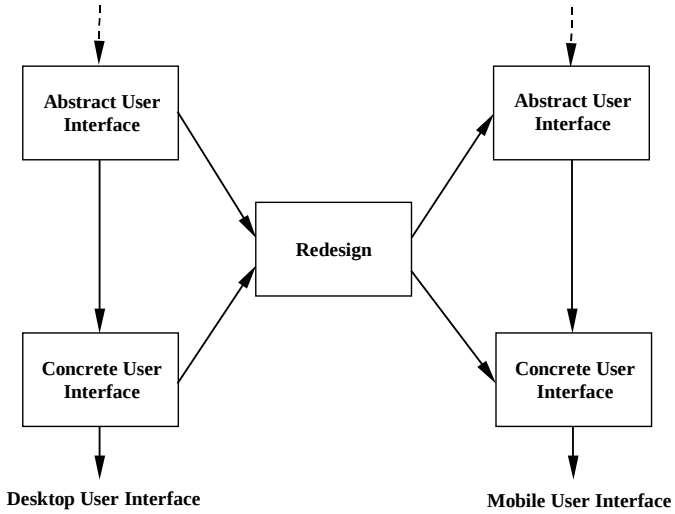

Fig. 5. The architecture of the redesign feature in TERESA.

In order to automatically redesign a desktop presentation for a mobile presentation we need to consider the limits of the available resources and semantic information. If we only consider the physical limitations we could divide large pages into small pages which are not meaningful. To avoid this, we also consider the composition operators indicated in the presentation specification. To this end, the algorithm tries to maintain groups of interactors (that are composed through some operator) for each page, thus preserving the communication goals of the designer. However, this is not always possible because of the limitations of the target platform. In this case, the algorithm aims to equally distribute the interactors into presentations of the mobile device. For example if the number of interactors supported for a large mobile presentation is six, and a desktop presentation contains a *Grouping* with eight interactors, this can be transformed into two mobile presentations, each one containing respectively a Grouping of four interactors. Since the composition operators capture semantic relations that designers want to communicate to users, this seems to be a good criterion for identifying the elements that are logically related and should be in the same presentation. In addition, the splitting of the pages requires a change in the navigation structure with the need of additional navigator interactors that allow the access to the newly created pages. The transformation also considers the possibility of modifying some interface elements. For example, the images are either resized or removed if there is no room for them in the resulting interfaces.

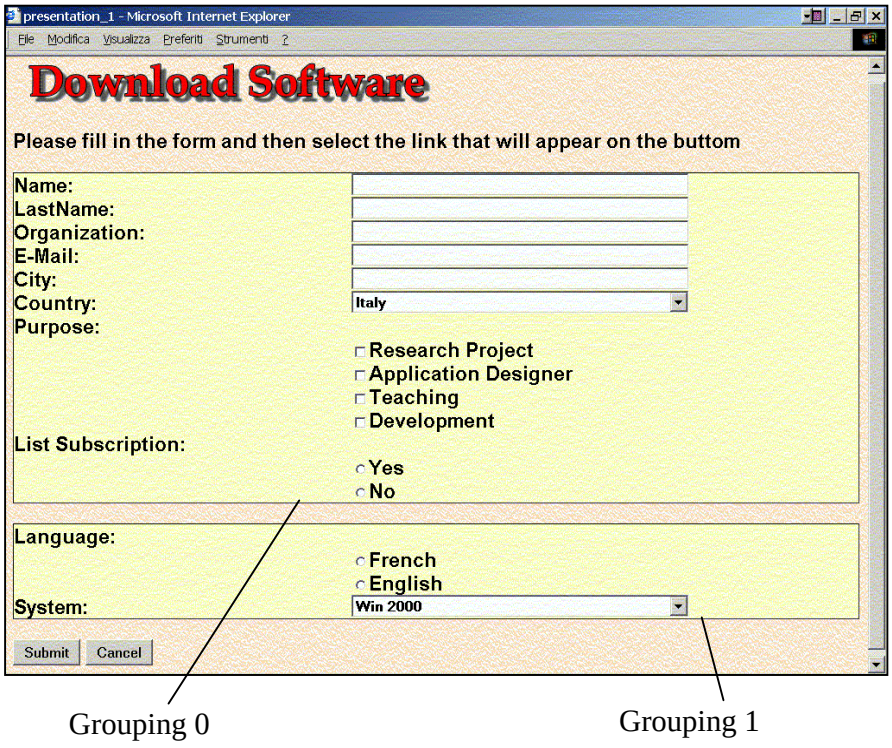

Fig. 6. Example of desktop Web user interface.

In order to explain the transformation we can consider a specific example of a desktop Web site and see how one of its pages (Figure 6) can be transformed using our method. The automatic transformation starts with the XML specification of the Concrete Desktop User Interface and creates the corresponding DOM tree-structure. The concrete user interface 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 and also other composition operators. Figure 7 represents the tree-structure of the XML file for the *desktop_ Download* presentation shown in Figure 6.

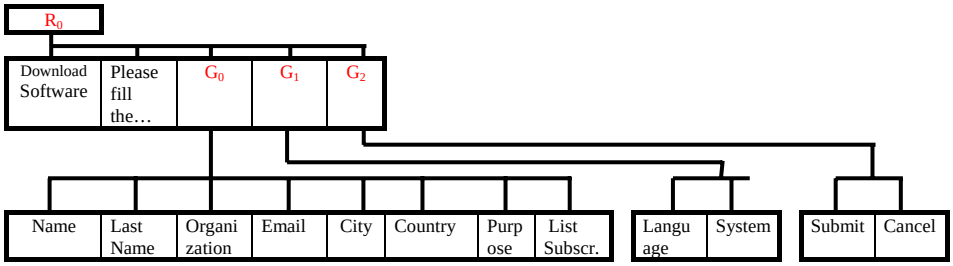

Fig. 7. Tree-structure of XML file for the "desktop_Download" *presentation*.

The resulting structure contains the following elements:

- composition operator R_0 , contains 2 interactors ("Download Software", "Please fill the form…") and 3 groupings (G_0, G_1, G_2);
- composition operator G_0 , contains 8 interactors (Name, Lastname, Organization, Email, City, Country, Purpose, List Subscription);
- composition operator G_1 , contains 2 interactors (Language, System);
- composition operator G_2, contains 2 interactors (Submit,Cancel);

The *relation* operator 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 the three aforementioned grouping operators. In general, the relation operator identifies a relation 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. In Figure 7 we can see the names of the interactors used in the *desktop_Download* presentation. There are also two *grouping* operators (G_0 and G_1) representing the two fieldsets in the user interface in Figure 6 and a grouping operator (G_2) involving the two buttons "Submit" and "Cancel".

Overall, this desktop presentation contains 14 interactors, which are too many for a mobile phone presentation. We assume that a presentation for a large mobile phone

(such as a smartphone) can contain a maximum number of six interactors. 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 in Figure 7, the algorithm makes a depth first visit starting with the root, and generates the mobile presentations by inserting elements contained in each level until the maximum number of widgets supported by the target platform is reached.
The algorithm substitutes each composition operator (in the example G_0 and G_1) that cannot fit in the presentation with a link pointing to a mobile presentation containing their first elements. In this case the two links point to the *mobile_Download2* and *mobile_Download4* presentations, which contain the first elements of G_0 (i.e., "Name") and the first elements of G_1 (i.e., "Language"), respectively.

So looking at the example, the algorithm begins to insert elements in the first "*mobile_Download1*" presentation and when it finds a composition operator (such as G_0), it starts to generate a new mobile presentation with its elements; so we obtain:

mobile_download1 = {**R(**"Download Software", "Please fill the form…", G_0, ….**)**}

The composition operator for the elements in mobile_Download1 is the Relation R_0. Continuing the visit, the algorithm explores the composition operator G_0. It has 8 elements but they cannot fit in a single new presentation. Thus, two presentations are created and the algorithm distributes the elements equally between them. We obtain:

mobile_Download2 = {**G(**Name, Lastname, Organization, Email**)**}
mobile_Download3 = {**G(**City, Country, Purpose, List Subscription**)**}

The composition operator for these two mobile presentations is grouping because the elements are part of G_0. The depth first visit of the tree continues and reaches G_1. It inserts a corresponding link in the mobile_Download1 presentation, which points to the new generated mobile_Download4 presentation where it inserts the elements of G_1.

Finally, we obtain:

mobile_Download1 = { **R(**"Download Software", "Please fill the form…", G_0, G_1, G_2) }
mobile_Download2 = {**G(**Name, Lastname, Organization, Email**)**}
mobile_Download3 = {**G(**City, Country, Purpose,List Subscription**)**}
mobile_Download4 = {**G(**Language, System**)**}

The entire last element of a Relation should be in the same presentation containing the elements composed by a Relation composition operator because it is the element that defines the association with the others elements. When the last element is another composition of elements (such as G_2), it is inserted into the presentation completely.

Thus, mobile_Download1 presentation becomes:

mobile_Download1 = { **R(**"Download Software", "Please fill the form…", "Form – part 1", "Form – Part 2", **G(**Submit,Cancel**))** }

Figure 8 shows the resulting presentations for the mobile device.

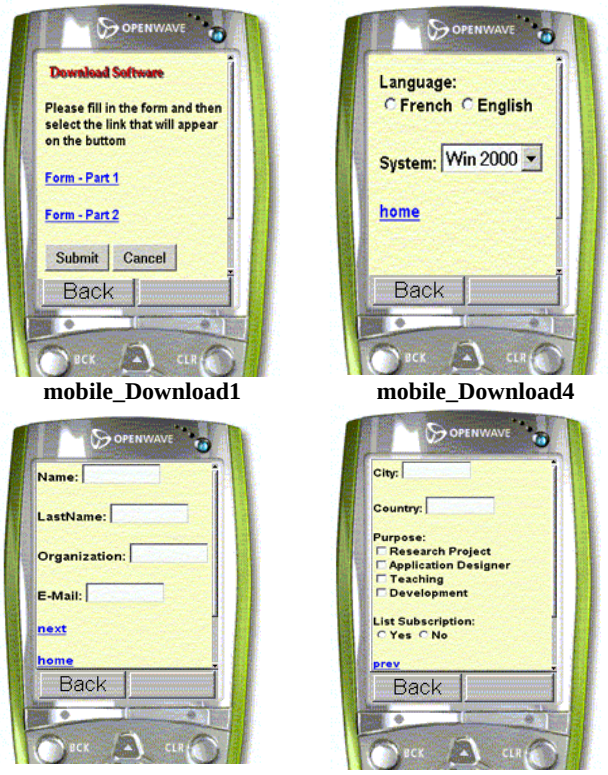

Fig. 8. Result of example desktop page transformed into four mobile pages.

Connections

The XML specifications of 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 the following connections among the presentations for the mobile phone:

- original connections of desktop presentations are associated to the mobile presentations that contain the interactor triggering the transition. In the example the connection associated with the "Submit" button is asociated with the *mobile_Download1* presentation. The destination for each of these connections is the first mobile presentation obtained from the splitting of the original desktop destination presentations;
- 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, we have two new links "Form - Part 1" and "Form – Part 2" which support access to the pages associated with the first interactor of G_0 and the first interactor of G_1 respectively:

  mobile_Download1 ===== Form – Part 1 ======> mobile_Download2
  mobile_Download1 ==== Form – Part 2 ======> mobile_Download4

- 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 tool and we obtain the following connections:

  mobile_Download2 ===== next ======> mobile_Download3
  mobile_Download3 ===== prev ======> mobile_Download2

  the connections above, are useful to navigate between presentations "*mobile_Download2*" and "*mobile_Download3*" which contain the results of the splitting of the G_0 elements.

  mobile_Download2 ===== home ======> mobile_Download1
  mobile_Download4 ===== home ======> mobile_Download1

  the connections above are the corresponding connections for going back from presentations containing the first elements to presentations containing the links to the newly created pages. In the example, we have the "Form – Part 1" link, which is contained in "*mobile_Download1*" presentation. Likewise, we have the "Form – Part 2" link contained in "*mobile_Download1*" presentation. Thus, we need two home links that allow going back to mobile_Downolad1 from mobile_Download2 and mobile_Download4.

- complex desktop connections may need to be split into elementary connections if the associated interactors are included in different mobile presentations (in the example of Figure 6 there are no complex connections).

Other considerations

Our transformation addresses a number of further issues. Attributes for desktop presentations must be adapted to mobile presentations. For example, the maximum dimension for a font used in a desktop presentation different from the maximum for a mobile device, and consequently large fonts are resized. The transformation of desktop presentations containing images produces mobile presentations also containing images only if the target mobile devices support them. Because of the dimension of mobile screens, original desktop images need to be resized for the specific mobile device. In our classification, images are only supported by large and medium mobile phones.

In consideration of the screen size of most common models of mobile phones currently on the market, we have calculated two distinct average screen dimensions: one for large models and another for medium size. From these two average screen dimensions (in pixels), we have deduced the reasonable max dimensions for an image in a presentation for both large and medium devices. The transformed images for mobile devices maintain the same aspect ratio as those of the original desktop interface. In *mobile_Download1* presentation we have an example of resize of image "Download Software".

Interactors often do not have the same weight (in terms of screen consumption) and this has consequences on presentations. From this viewpoint, *single_selection* and *multiple_selection* interactors can be critical depending on their cardinality. For example, a single_selection composed of 100 choices can be represented on a desktop page through a list, but this is not suitable for a mobile page because users should scroll a lots of items on a device with a small screen. A possible solution could be dividing 100 choices in 10 subgroups in alphabetical order (a-c, d-f,.. ...w-z) and each subgroup is connected to another page containing a pull-down menu only composed of the limited number of choices associated with that subgroup and not of all the original 100 choices. For example, the menu for selection of a Country present in desktop presentation can be transformed as shown in Figure 9.

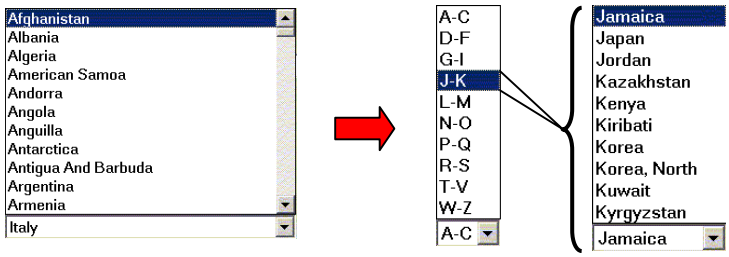

Fig. 9. Transformation of a single selection interactor for desktop system into one interactor for mobile presentations.

In the previous example of Figure 8 another simple solution has been applied, substituting the country pull-down menu of *desktop_Download* presentation with a text edit in the *mobile_Download3* presentation.

In general, the problem of redesigning and transforming a set of presentations from a platform to another is not easy and often involves many complex aspects related to user interface design.

Conclusions and Future Work

We have presented an approach to flexible multi-user interface design. The approach is supported by the new version of the TERESA tool, which is publicly available at http://giove.isti.cnr.it/teresa.html.

It provides designers with multiple entry points to the design process (which can be the task, abstract, or concrete user interface level) in order to change the results of automatic transformations from the task to the lower levels, and support redesign for different platforms. This last feature has also been considered in the CAMELEON project where the Vaquita tool has been used for reverse engineering of the design of a desktop Web interface. Its results are then input into the TERESA tool for redesigning for a mobile platform.

Future work will be dedicated to integrating natural interaction techniques in this environment in order to allow even people with little programming experience to easily use it in the design of multi-device interfaces. We also plan to add a feature in TERESA so that when a description at a lower level is modified, then such modifications are reflected into the description at the upper levels.

Acknowledgments

This work has been supported by the CAMELEON EU IST Project (http://giove.isti.cnr.it/cameleon.html). We also thank our colleagues in the project for useful discussions.

References

1. Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S., Shuster, J. UIML: An Appliance-Independent XML User Interface Language, Proceedings of the 8th WWW conference, 1999.
2. Bouillon, L., Vanderdonckt, J., Retargeting Web Pages to other Computing Platforms, Proceedings of IEEE 9th Working Conference on Reverse Engineering WCRE'2002 (Richmond, 29 October-1 November 2002), IEEE Computer Society Press, Los Alamitos, 2002, pp. 339-348.
3. Calvary, G. Coutaz, J. Thevenin, D. Limbourg, Q. Bouillon, L. Vanderdonckt, J., "A Unifying Reference Framework for Multi-target User interfaces", Interacting with Computers Vol. 15/3, Pages 289-308, Elsevier.
4. G. Calvary, J. Coutaz, D. Thevenin. A Unifying Reference Framework for the Development of Plastic User Interfaces. IFIP WG2.7 (13.2) Working Conference, EHCI01,Toronto, May 2001, Springer Verlag Publ., LNCS 2254, M. Reed Little, L. Nigay Eds, pp.173-192.

5. Florins M., Vanderdonckt J., Graceful degradation of user interfaces as a design method for multiplatform systems, Proceedings ACM IUI'04, Funchal, ACM Press.
6. G. Mori, F. Paternò, C. Santoro, Design and Development of Multi-Device User Interfaces through Multiple Logical Descriptions, IEEE Transactions on Software Engineering, August 2004, Vol.30, N.8, pp.507-520, IEEE Press.
7. G. Mori, F. Paternò, C. Santoro, "CTTE: Support for Developing and Analysing Task Models for Interactive System Design", IEEE Transactions on Software Engineering, pp. 797-813, August 2002 (Vol. 28, No. 8), IEEE Press.
8. Mullet, K., Sano, D., *Designing Visual Interfaces*. Prentice Hall, 1995.
9. Paganelli, L., 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), pp. 169-189 (2003).
10. Paternò, F., Model-Based Design and Evaluation of Interactive Application. Springer Verlag, ISBN 1-85233-155-0, 1999.
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. Pribeanu C., Personal Communication, 2004.
13. Puerta, A., Eisenstein, J., Towards a General Computational Framework for Model-based Interface Development Systems, Proceedings ACM IUI'99, pp.171-178.
14. Puerta, A., Eisenstein, XIML: A Common Representation for Interaction Data, Proceedings ACM IUI'01, pp.214-215.