Automatic Semantic Platform-dependent Redesign

Giulio Mori, Fabio Paternò
ISTI-CNR
Via G.Moruzzi 1
56124 Pisa, Italy
fabio.paterno@isti.cnr.it

ABSTRACT
In this paper, we describe a solution to support automatic platform-dependent semantic redesign. This is obtained within the framework of TERESA, a model-based environment supporting flexible development of multi-device interactive applications. The goal is to show how semantic information regarding the interactive part of a Web application can support more flexible and effective solutions for adapting the interface to different platforms.

KEYWORDS
Multi-Device Interfaces, Model-based Design, Semantic Analysis.

INTRODUCTION
The increasing availability of various types of devices, including mobile devices poses a number of challenges to designers and developers of interactive applications. In particular most applications are becoming accessible through multiple devices. However, in order to satisfy the user experience it is important that the interface be able to adapt to the interaction resources actually supported by each device. Having a separate development of the user interface software for each potential support is rather expensive. This has raised interest in research results in model-based approaches for interactive applications [S96, P09], which provide declarative descriptions of the user interface and then are supported by tools that generate the corresponding implementation taking into account the features of the target devices. This approach has also been adopted in W3C standards (such as XForms[XW3C]). The model-based approach seems suitable to better manage the complexity involved in multi-device interactive applications because they allow the development of the logical models and then rendering and tailoring the results to the various interaction platforms. Thus, there is a need for tools able to support them. An example is TERESA [MPS04], which provides support for various methods based on the model-based approach.

This paper provides an introduction to the TERESA environment, and how it can support designers and developers of multi-device interfaces, and then it discusses how this environment can 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 the mobile devices, which is able to adapt to their features while reusing elements already generated for the desktop version. This paper shows how platform-dependent semantic redesign can support this type of adaptation. We mean for platform-dependent redesign 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 interfaces that capture the possible tasks that user intend to accomplish.

RELATED WORK
The Web is the most common user interface. There is a general agreement on the models that are relevant in user interface design [S96], [CCT03], [P04]:

- *Task and object model*, at this level, the logical activities that need to be performed in order to reach the users' goals are considered. Often they are represented hierarchically along with indications of the temporal relations among them and their associated attributes. The objects that have to be manipulated in order to perform tasks can be identified as well.
- *Abstract user interface*, in this case the focus shifts to the user interface supporting task performance. Only the logical structure is considered, in a modality-independent manner, thereby avoiding low-level details. Interaction objects are described in terms of their semantics through interactors [PL94]. Thus, it is possible to indicate, for

example, that at a given point there is a need for a selection object without indicating whether the selection is performed graphically or vocally or through a gesture or some other modality.

- *Concrete user interface*, at this point each abstract interactor is replaced with a concrete interaction object that depends on the type of platform and media available and has a number of attributes that define more concretely how it should be perceived by the user.
- *Final user interface*, at this level the concrete interface is translated into an interface implemented by a specific software environment (e.g. XHTML, Java, …).

To better understand such abstraction levels we can consider an example of a task: making a hotel reservation. This task can be decomposed into selecting arrival and departure dates and other subtasks. At the abstract user interface level we need to identify the interaction objects needed to support such tasks. For example, for easily specifying arrival and departure days we need selection interaction objects. When we move on to the concrete user interface, we need to consider the specific interaction objects supported. So, in a desktop interface, selection can be supported by a graphical list object. This choice is more effective than others because the list supports a single selection from a potentially long list of elements. The final user interface is the result of these choices and others involving attributes such as the type and size of the font, the colours, and decoration images that, for example, can show the list in the form of a calendar.

Many transformations are possible among these four levels for each interaction platform considered: from higher level descriptions to more concrete ones or vice versa or between the same level of abstraction but for different type of platforms or even any combination of them. Consequently, a wide variety of situations can be addressed. More generally, the possibility of linking aspects related to user interface elements to more semantic aspects opens up the possibility of intelligent tools that can help in the design, evaluation and run-time execution.

In order to ease the development of models and apply them also to interfaces obtained through other approaches reverse engineering techniques have been developed, which take the user interface implementation (usually Web pages) and automatically build the corresponding underlying conceptual descriptions. Examples are Vaquita [BV02] and PIMA [BBG04] that build abstract/concrete user interface descriptions from their implementations or WebRevnEnge [PP03] which builds the corresponding task models. On the other hand, even following a forward engineering approach people can think to reuse some design elements for the interface of a different platform.

The problem of adapting the interface to different platforms can be addressed in many ways. There are automatic tools which mainly translate from an implementation language to another one (such as XHTML-to-WML tools). This type of low-level syntactical transcoding often provides poor results in terms of usability because it follows rigid rules and mainly try to fit the same design into different devices. We do not think that just resizing elements is sufficient for obtaining general solutions. The more the environment is able to go up in terms of corresponding abstraction levels, the more substantial the design choices that can be performed taking into account the characteristics of the new target device. This means that 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 for the new device whereas if it is able to identify the corresponding abstract object then the environment can change the choice of the interaction technique to use for implementing it 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 has no longer importance and new tasks should be enabled and supported. It is difficult to perform completely automatically this level of reasoning. TERESA is able to support all such possible abstraction levels differently from other approaches, such as UIML [APB94], which are mainly limited to the concrete/abstract level.

THE TERESA FRAMEWORK AND ENVIRONMENT

TERESA [MPS04] is a model-based authoring environment for design and development of multi-device interfaces. While the tool provides support for various types of user interface abstractions, it does not force designers to follow a specific method (rif. EHCI04).

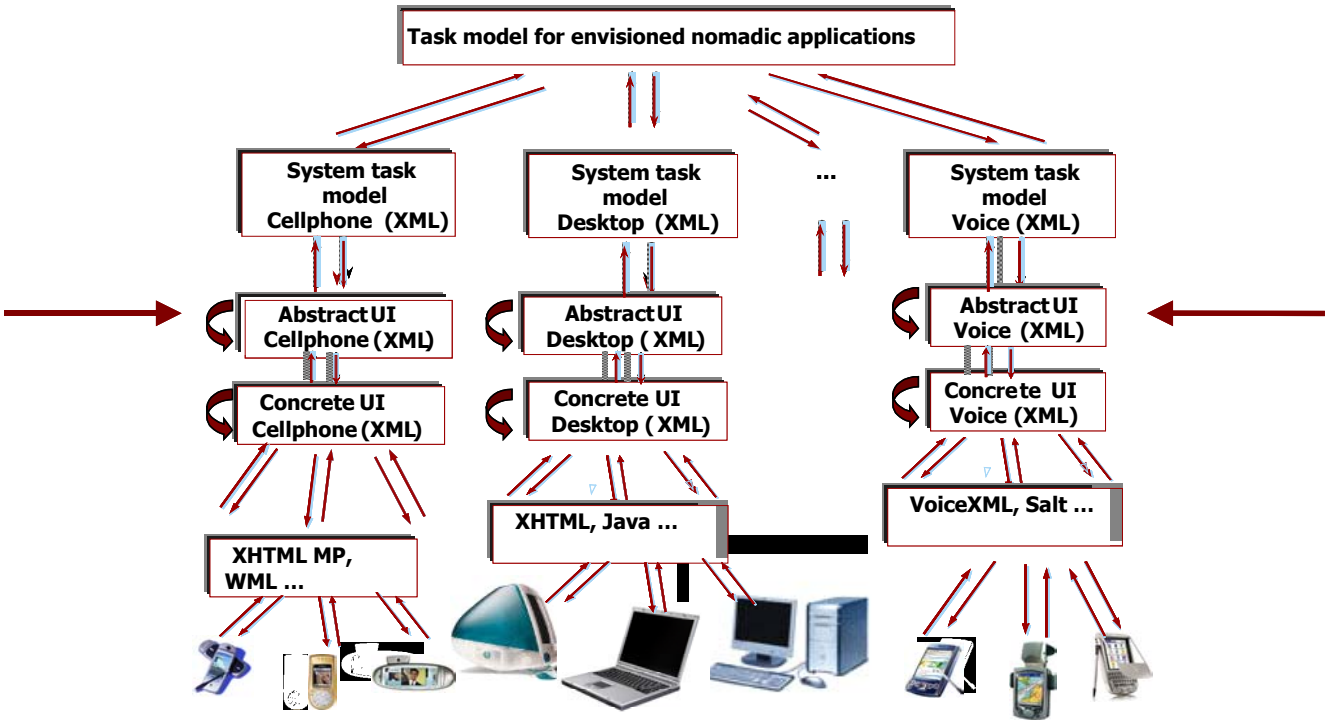


Figure 1: The TERESA Environment

As Figure 1 shows, the starting point of the design process can vary. It is possible to start with the task model of the nomadic application where designers can specify for each task what platform is suitable to support it and the possible temporal relations among tasks supported by different devices. However, it is also possible to start the development process from the task model or the abstract description of a given platform.

The tool supports a number of transformations that allow designers to take a logical description and obtain a more refined one taking into account the features of the target platforms. This solution allows designers to concentrate on conceptual aspects without having to deal with a plethora of low-levels details. The corresponding final user interfaces are generated automatically. Currently, various user interface implementation languages are supported (XHTML, XHTML Mobile Profile, VoiceXML) and support for others (such as X+V and SVG) is under development. In case a new implementation language has to be supported just the last transformation from the concrete interface description (the most refined logical description) to it has to be implemented, without changing the other transformations or the tool architecture.

The task and the abstract levels are described by two device independent languages whereas the concrete level is described through platform-dependent refinements of the abstract level in order to be able to describe the platform attributes and specify the values that define their state. The tool is able to keep relations between elements in one abstraction level and the corresponding elements in another element. This is useful to receive some semantic information regarding an aspect that is under consideration. So, for example it is possible to ask what task corresponds to a given interactor.

In TERESA an abstract user interface is composed of a number of presentations and connections among them. Each presentation defines a set of presentation and interaction techniques perceivable by the user at a given time. The connections define the dynamic behaviour of the user interface. More precisely, they indicate what interactions trigger a change of presentation and what the next presentation is. They can be associated with conditions in case a specific combination of interactions should trigger the change of presentation. The structure of the presentation is defined in terms of interactors (abstract descriptions of interaction objects classified depending on their semantics) and their composition operators. It is possible to distinguish between interactors supporting user interaction (interaction elements) and those that present results of application processing (only_output elements). The interaction elements imply an interaction between the user and the application. There are different types of interaction elements depending on the type of task supported. We have selection

elements (to select between a set of elements), edit (to edit an object), control (to trigger an event within the user interface, which can be useful to activate either a functionality or the transition to a new presentation). Differently, an only_output element defines an interactor which implies an action only from the application. There are different types of only_output elements (text, object, description, feedback) depending on the type of output the application provides to the user: a textual one, an object, a description, or a feedback about a particular state of the user interface.

The composition operators can involve one or two expressions, each of them can be composed of one, several interactors or, in turn, compositions of interactors. In particular, the composition operators have been defined taking into account the type of communication effects that designers aim to achieve when they create a presentation. 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 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.

SEMANTIC REDESIGN

Semantic redesign is an example of feature that is supported in TERESA through different possible processes. It can be obtained in a process where there exist a desktop version and through reverse engineering tool (1 and 2 in Figure 2) the corresponding concrete and abstract user interfaces are obtained, these are given as input (3) for the redesign module which generates (4) the abstract and concrete descriptions (and their mappings) for the mobile interface and finally the corresponding user interface adapted to its features.

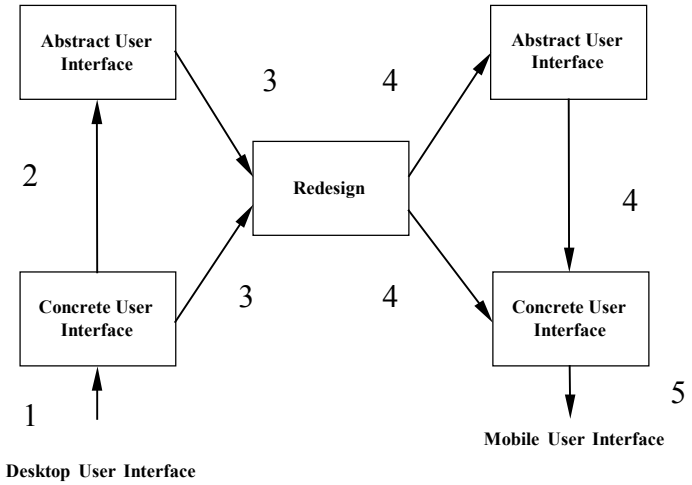


Figure 2: Semantic Redesign with Reverse and Forward Engineering

Another process that can benefit from the redesign module is in Figure 3, which mainly differs for the initial part. In this case the designer uses TERESA for creating first the abstract user interfaces, next the corresponding concrete and lastly the final interface for the desktop system. Then, the abstract and concrete descriptions are again considered as input for the redesign module that can produce the new interface and corresponding logical descriptions.

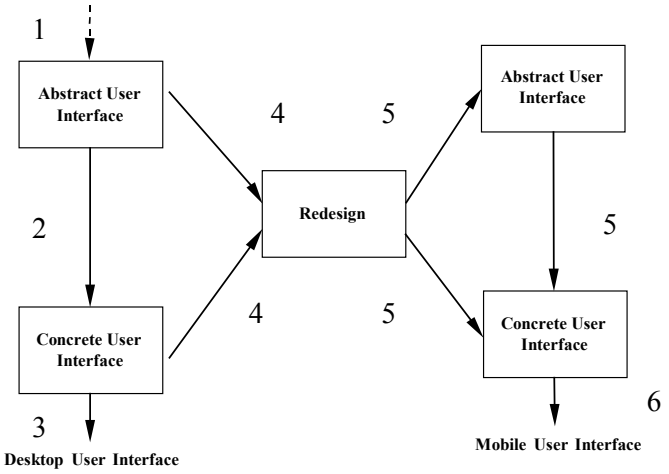


Figure 3: Semantic Redesign with Forward Engineering

A variant of this solution is represented in Figure 4. The difference in this case is that the redesign module receives as input also information from the nomadic task model. This means that it also receives information regarding which platforms are suitable to support a given task. This means that if a task is considered suitable 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 be not created in the logical description of the mobile interface and, consequently, in its implementation.

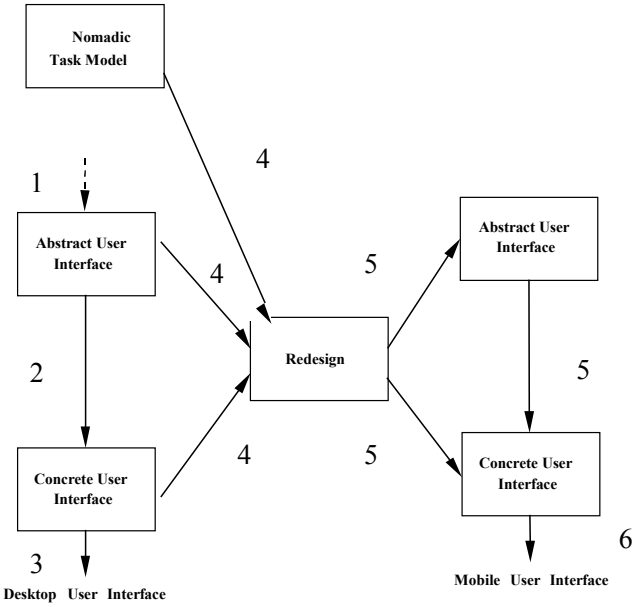


Figure 4: Semantic Redesign with Nomadic Task Model Support

THE TRANSFORMATION SUPPORTING SEMANTIC PLATFORM 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. 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. 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 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 platform-dependent redesign 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 Figures 2, 3, 4): 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.

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.

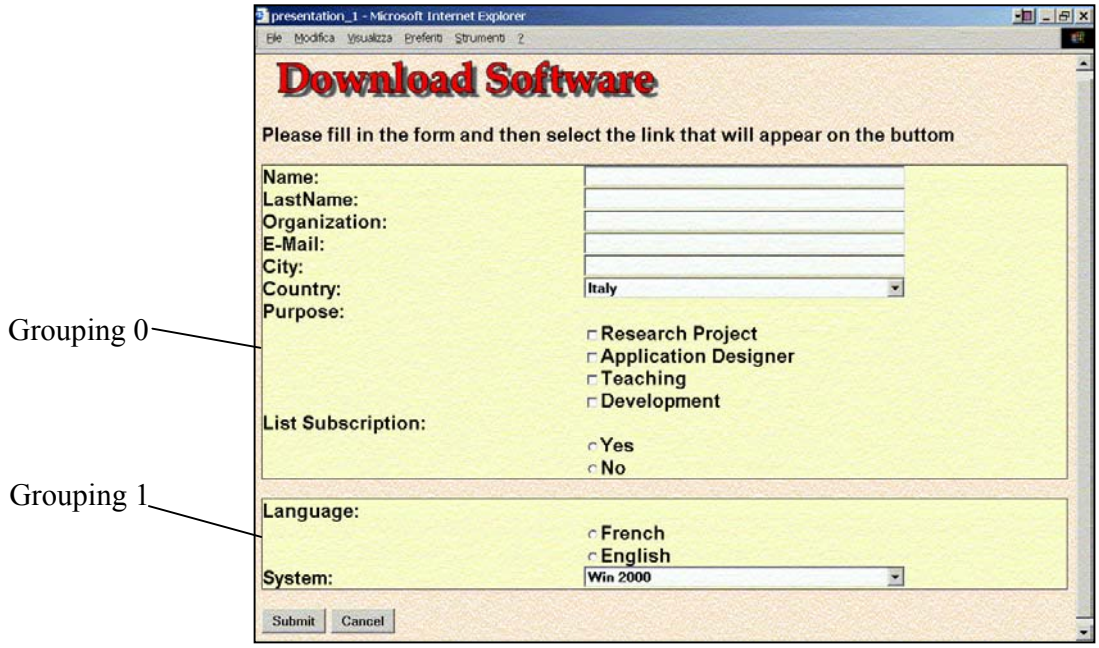


Figure 5: Example of desktop Web user interface.

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 5) 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 6 represents the tree-structure of the XML file for the *desktop_ Download* presentation shown in Figure 5.

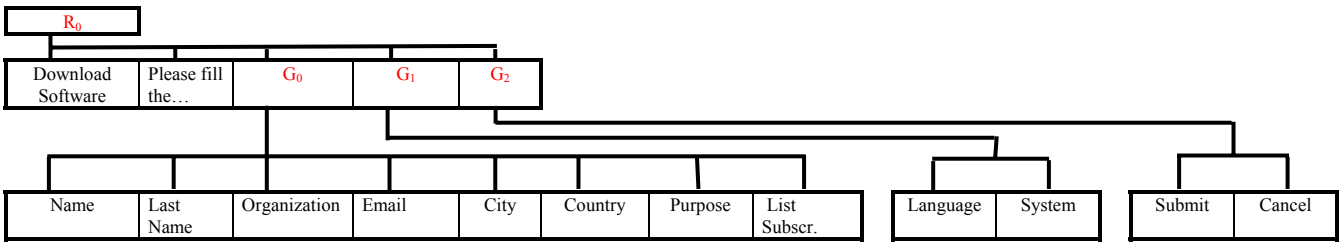


Figure 6: Tree-structure of XML file for the "desktop_Download" *presentation*.

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 6 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) implemented by the two fieldsets in the user interface in Figure 5 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 6, 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 (in the example *mobile_Download2*) 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 the elements of G_1 are inserted.

Thus, 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**)**}

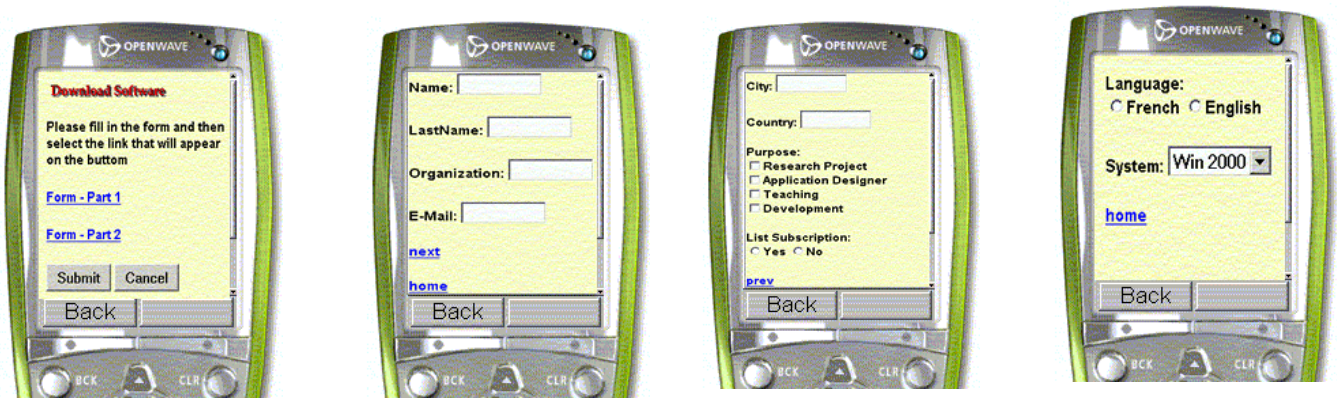
mobile_Download4 = {**G(**Language, System**)**}

Since 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 7 shows the resulting presentations for the mobile device.



mobile_Download1 **mobile_Download2** **mobile_Download3** **mobile_Download4**

Figure 7: 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 associated 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.

- There is also the need of identifying the connections for going back from the new presentations containing the first elements of the composed 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:

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

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

- 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 5 there are no complex connections).

Figure 8 shows the resulting navigation structure.

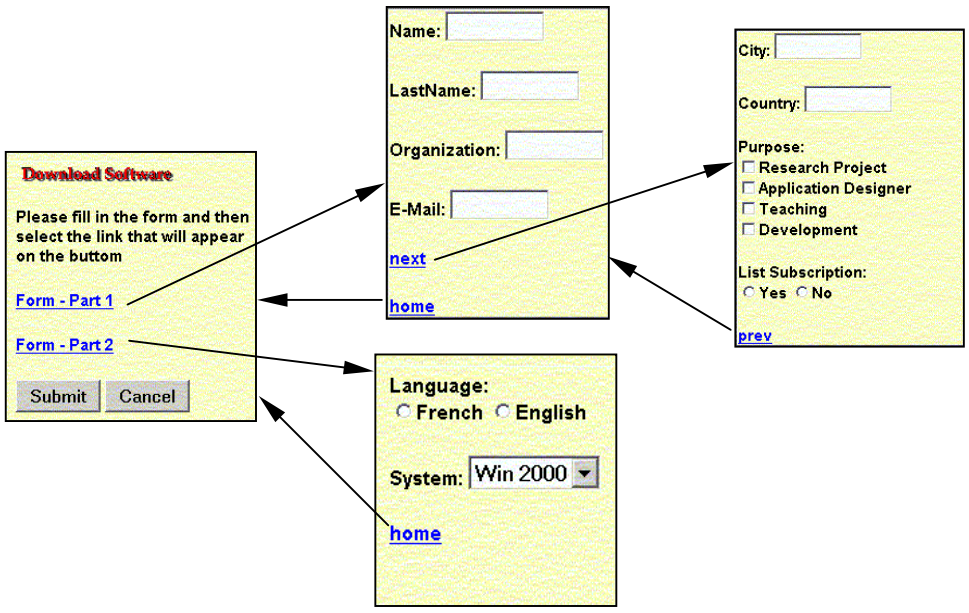


Figure 8: The Mobile Presentations and the Associated Links.

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 lot 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.

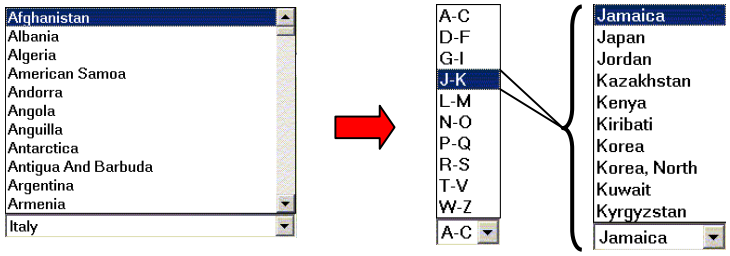


Figure 9: Transformation of a single selection interactor for desktop system into one interactor for mobile presentations.

In the previous example of Figure 7 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.

Another issue considered is that also labels need to adapt to the devices' interaction resources. The current solutions is to use table of synonyms, so that in case of labels too long they are replaces with shorter similar terms.

CONCLUSIONS

We have discussed the TERESA framework for authoring multi-device interactive applications and presented how it supports semantic platform redesign. Such feature can be used in at least three different design processes whose main features have been explained. An example of application of the approach proposed has been reported.
Further work will be dedicated to integrating other techniques within this approach, such as those for automatic text summarization. In addition, TERESA is being extended in order to support generation of interfaces in other W3C languages, such as SVG and X+V. TERESA is publicly available at giove.isti.cnr.it/teresa.html

ACKNOWLEDGMENTS

This work has been supported by the CAMELEON EU project http://giove.isti.cnr.it/cameleon.html and the SIMILAR NoE http://www.similar.cc.

REFERENCES

[APB94] Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S., Shuster, J., 1994. UIML: An Appliance-Independent XML User Interface Language, Proceedings of the 8th WWW conference.
[BBG04] Banavar, G. Bergman L. D., Gaeremynck, Y. Soroker, D. Sussman, J., Tooling and system support for authoring multi-device applications, The Journal of Systems and Software 69 (2004) 227–242

[BV02] 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.
[CCT03] 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. 2003
[CMP04] F.Correani, G.Mori, F.Paternò, Supporting Flexible Development of Multi-Device Interfaces, Proceedings EHCI-DSVIS'04, Hamburg, July 2004, Springer Verlag, Lecture Notes Computer Science.
[MPS04] G. Mori, F. Paternò, C. Santoro, 2004. 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..
[PP03] 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).
[P09] Paternò, F., 1999. *Model-Based Design and Evaluation of Interactive Applications*. Springer Verlag, ISBN 1-85233-155-0, 1999.
[P04] F.Paternò, Model-based Tools for Pervasive Usability, accepted for publication on Interacting with Computers, Elsevier, 2004.
[PL94] Paternò, F., Leonardi, A. 1994. "*A Semantics-based Approach to the Design and Implementation of Interaction Objects"*, Computer Graphics Forum, Blackwell Publisher, Vol.13, N.3, pp.195-204..
[S96] Szekely, P., 1996. Retrospective and Challenges for Model-Based Interface Development. 2nd International Workshop on Computer-Aided Design of User Interfaces, Namur, Namur University Press.
[XW3C] XForms. 2004. – The Next Generation of Web Forms, http://www.w3.org/MarkUp/Forms/