Desktop-to-Mobile Web Adaptation through Customizable Two-Dimensional Semantic Redesign

Fabio Paternò, Giuseppe Zichittella

Desktop-to-Mobile Web Adaptation through Customizable Two-dimensional Semantic Redesign

Fabio Paternò, Giuseppe Zichittella

CNR-ISTI, HIIS Laboratory, Via Moruzzi 1, 56124 Pisa, Italy
{fabio.paterno, giuseppe.zichittella}@isti.cnr.it

Abstract. In this paper we present a novel method for desktop-to-mobile adaptation. The solution also supports end-users in customizing multi-device ubiquitous user interfaces. In particular, we describe an algorithm and the corresponding tool support to perform desktop-to-mobile adaptation by exploiting logical user interface descriptions able to capture interaction semantic information indicating the purpose of the interface elements. We also compare our solution with existing tools for similar goals.

Keywords: Ubiquitous Applications, Multi-Device Environments, Adaptation.

Introduction

One of the main issues in current technological settings is how to design and develop interactive applications that can be accessed through a wide variety of devices (ranging from small watches to very large screens, including various types of smartphones, PDAs and Digital TVs). This is particularly important in Web applications, which are the most common ones.

The vision of ubiquitous computing [16] is that the users operate in intelligent environments, which are aware of users' needs and able to assist, even proactively, the users in performing their activities and reaching their goals. To this end, one important aspect is the possibility for a user surrounded by multiple devices to freely move about and receive user interfaces adapted to the current context of use.

In current mobile devices various solutions are adopted for accessing Web applications originally developed for desktop systems. Some just cut the page to the display area, thus showing only a limited portion. Others, such as those using the Small Screen Rendering Technique in the Opera mini browser, provide the narrow view in which the content is vertically arranged in order to avoid horizontal scrolling. The most sophisticated solutions are those, such as the Safari browser in the IPhone, which automatically resize the Web page to the screen size and allow the user to zoon in and out through gestures in the touch interface. However, their usability is often low in terms of Web navigation, since users have to make various zoom in and out interactions in order to identify the part of content that they are looking for.

The solutions for such issues can benefit from user interface model-based approaches, in which declarative descriptions of the user interface are used in order to avoid dealing with a plethora of low-level implementation details associated with the

wide number of available devices and implementation languages. Despite such potential benefits, their adoption has mainly been limited to professional designers, but new solutions have recently been emerging that are able to extend such approaches in order to achieve natural development by enabling end users to develop or modify interactive applications still using conceptual models, but with continuous support that facilitates their development, analysis, and use [1].

Model-based languages are utilized at design time to help the user interface designer cope with the increasing complexity of today's applications and contexts. The underlying user interface models are mostly used to generate a final user interface code, which is then executed at run time. However, approaches utilizing the models at run time are receiving increasing attention. We agree with Sottet et al. [13], who call for keeping the models alive at run time to make the design rationale available.

In the following, we present some research work that exploits model-based approaches for multi-device ubiquitous applications. We show a new tool for desktop-to-mobile adaptation, called customizable two-dimensional semantic redesign. We present its underlying algorithm and compare its results with those of other current tools. The environment also allows end users to customize the adaptation process. Lastly, some conclusions are drawn along with indications for future work.

Related Work

Various approaches are possible to support adaptation for mobile devices. Bickmore [2] proposed a classification into five categories: device-specific authoring (one version for each target device type), multiple-device authoring (one version, with subversions for the various targets, e.g. using different stylesheets), client-side navigation (adaptation is performed directly by the client), Web page filtering (adaptation is obtained by content filtering) and automatic re-authoring (one version exists, which is then automatically adapted for the target device). Automatic re-authoring can be further divided into transducing (the original structure is preserved and the elements are adapted, e.g. images resized) and transforming (the structure is adapted as well). Our approach is an example of automatic re-authoring, supporting transforming (since the original pages can even be split into multiple mobile pages if they are too expensive in terms of space consumption).

Various contributions have been put forward in this area and it is not possible to mention all of them. The OPA browser [14] allocates various functions for Web browsing on each numerical key of a cellular phone. Buyukkokten et al. [4] proposed a novel technique for form summarization, which is also able to automatically summarise texts according to various policies. Laakko and Hiltunen [6] proposed a technique for server side adaptation. We too support a solution using an adaptation proxy but we also exploit logical descriptions that allow us to propose a more general solution. The Roam system [5] is another environment for multi-device applications. It also logically partition an application in a set of components but then it requires that developers provide various implementations for different types of devices. Thus there is little support for automatic adaptation. Studies on usability of mobile adaptation [7] by Kaikkonen and Roto indicate that adaptation should not completely destroy the

original structure of the desktop pages in order to allow users to still be able to associate the mobile pages with the original ones. One important issue in this adaptation process is how to handle table adaptation. In [10] there is a proposal that allows users to interactively fold and unfold the tables rows and/or columns. However, such manual adaptations are lost when users access the tables again.

A Model-based Architecture for User Interface Adaptation

We have designed and developed a model-based architecture for user interface adaptation, which supports reverse and forward transformations that are able to transform existing desktop Web applications for various interaction platforms. The basic assumption is that there exists a huge amount of easily accessible content for desktop Web applications, which can be processed and transformed to support multi-device interfaces, even across non-Web implementation languages. The advantage of this solution with respect to others (e.g. [9]) is that it does not require that the applications be implemented using a particular toolkit in order to make them able to adapt.

When the user accesses the application through an interaction platform other than the desktop, the intermediate adaptation server (which includes a proxy server) transforms its user interface by building the corresponding logical description and using it as a starting point for creating the implementation adapted to the accessing device (see Figure 1). Lastly, the user interface implementation for the target device is generated.

The reverse engineering module analyses the content of the HTML and the associated CSS files and builds the logical description of the desktop user interface, which is provided as input to the adaptation module.

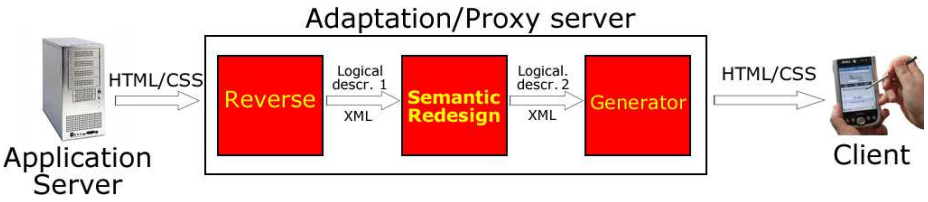

Fig. 1. The Main Phases of the Adaptation Process

In the process of creating an interface version suitable for a platform different from the desktop, we use a semantic redesign module. This part of the environment automatically transforms the logical description of the desktop version into the logical description for the new platform. Therefore, the goal of this transformation is to provide a description of the user interface suitable for the new platform. This means that intelligent rules are used for adapting the description of the user interface to the new platform taking into account its capabilities (e.g. using interface elements that are more suitable for the new platform) but ensuring at the same time that the support for the original set of tasks is maintained. This solution allows the environment to exploit

the semantic information contained in the logical description. In this case the semantic information is related to the basic tasks that the user interface elements are expected to support.

This software architecture for user interface adaptation currently uses MARIA [12], a recent model-based language, which allows designers to specify abstract and concrete user interface languages according to the CAMELEON Reference framework [3]. This language represents a step forward in this area because it provides abstractions also for describing modern Web 2.0 dynamic user interfaces and Web service access. It provides an abstract language independent of the interaction modalities and concrete languages for a number of platforms. In general, concrete languages are dependent on the typical interaction resources of the target platform but independent of the implementation languages.

In MARIA an abstract user interface is composed of one or multiple presentations, a data model, and a set of external functions. Each presentation contains a number of user interface elements (interactors) and interactor compositions (indicating how to group or relate a set of interactors), a dialogue model describing the dynamic behaviour of such elements, and connections indicating when a change of presentation should occur. The interactors are classified in abstract terms: edit, selection, only_output, control, interactive description, .. Each interactor can be associated with a number of event handlers, which can change properties of other interactors or activate external functions.

The Adaptation Transformation

We have designed a new tool for adaptation: *Customizable Two-dimensional Semantic Redesign*. It supports adaptation from desktop-to-mobile devices and overcomes some of the limitations of previous approaches in the area [11] because it allows users to configure the adaptation process and provides more control over costs calculation and the adaptation results. For example, while previous solutions calculated the screen space requested by the user interface elements mainly in terms of vertical extension, the new algorithm calculates both the horizontal and the vertical consumption of screen space.

The new algorithm takes as input the concrete description of a desktop user interface in the MARIA language and goes through a number of steps. First, it performs some basic transformations: if the user provides preferences regarding the minimum and maximum fonts for the target device then the system transforms all the textual content in order to fit it into the given range. Next, it calculates the cost of all the interactors and composition operators in the provided specification. If the resulting total cost is sustainable for the target device, then the corresponding logical description is generated, otherwise it starts the process to reduce the cost in order to make it sustainable. The basic elements are adapted for the target device first: the images are shrunk, while preserving their aspect ratio, some interactors are replaced with others that are semantically equivalent but need less screen space (e.g. a list can be replaced with a drop-down menu), long texts are reduced in such a way that the part exceeding a limit is shown only on request, image and text in tables are reduced

in size. After these basic transformations the overall cost is recalculated and if it is not yet sustainable by the target device then the part of the algorithm related to page splitting is activated. The purpose of this phase is to split the original desktop presentation into two or more presentations that are sustainable for the target mobile device. For this purpose the algorithm considers the interactor compositions (groupings of elements or relations that involve two groups) and tables of elements, and associates some of them to newly generated mobile presentations, removing them from the current presentation in order to decrease its overall cost.

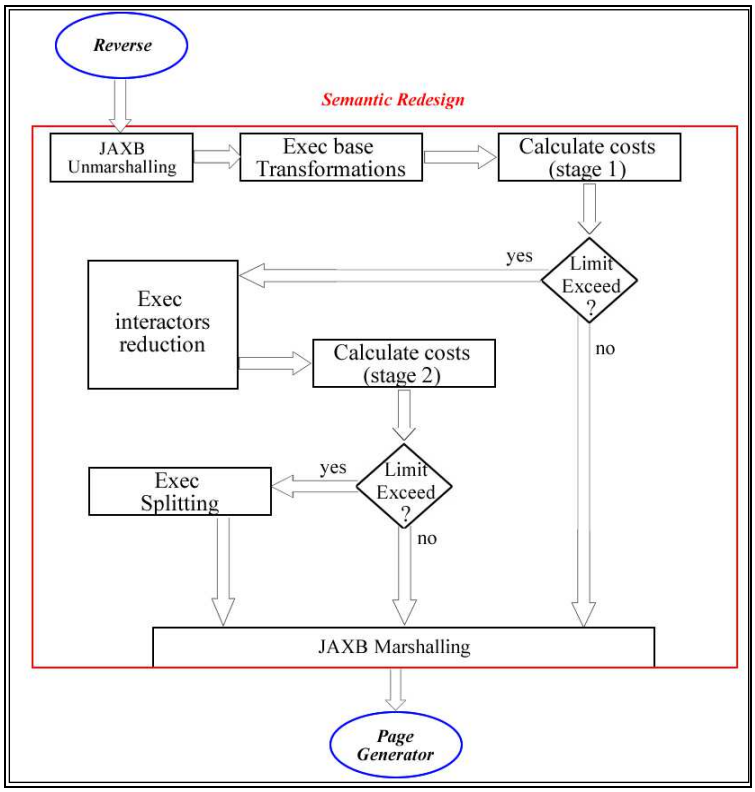

Fig. 2 The adaptation algorithm

The elements that determine the cost of the interactors are: the font attributes (size, style, type), the vertical and horizontal space required by a text, image dimensions, interline value, interactor type, ...

The algorithm has a parameter (Scrolling to Avoid), which allows the specification of which scrolling (vertical or horizontal) to avoid in the case that the presentation cost exceed the limits in both directions.

When the splitting part is activated the algorithm looks for a structured element in the logical description whose cost is sufficiently high that removing it would make the presentation sustainable for the target device. Then, such structured element would be allocated to a newly generated mobile presentation, which would be accessible through a link inserted in the original one. The structured elements

considered are groupings, relations, data tables and layout tables. When the element candidate for removal is a data table, the splitting is implemented differently. The table is split into two parts, the part composed of the columns visible without horizontal scrolling remains in the original presentation with an additional link allowing the user to continue to browse it in a separate presentation containing the remaining columns, from which it is possible to return by a similar link.

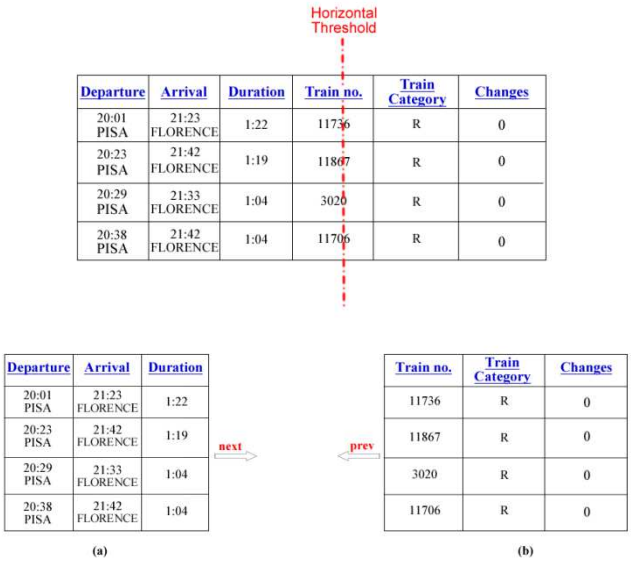

Fig. 3. Example of table splitting

In particular, the tool supports two ways to determine how splitting should be performed. In both cases it analyses the cost of the structured elements, which includes those of the composed interactors, and the cost of the tables (both data and layout tables). Then, the decision of the set of elements to allocate to the newly generated mobile presentation is given in one case by the most expensive element. In the other case the algorithm first calculates the elements whose removal would make the current presentation sustainable by the target device, and then selects the one that has the lowest cost. The rationale for this second option is that it allows obtaining a sustainable presentation but by removing the least amount of information possible, thus preserving the original design as much as possible.

End-User Adaptation Customization

In the research on user interface adaptation, one issue that we are considering is how to provide users with more control over the adaptation process in order to improve the usability of its results. In this context more control can mean various things. One important aspect is control over the rules that drive adaptation to the

various platforms (the most common case is desktop-to-mobile adaptation). For example, the adaptation engine is able to split the desktop pages when they require considerable amounts of interaction resources but some users may like to have more control over the splitting algorithm. End-User Development [8] (EUD) can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact. End-users already have difficulties with single device applications, thus it easy to understand how such difficulties increase when considering applications for multi-device environments. This is one further reason for providing better support for EUD in ubiquitous applications.

Figure 5 and 6 show the user interface that allows end users to configure the adaptation process. The various parameters are grouped according to the related user interface aspect considered. For the fonts, it is possible to specify the minimum and maximum font size in the target device, and the associated measure unit. For the radio buttons it is possible to indicate whether they should be transformed into an interactor that supports the same semantics but using less space screen. In this case, it is possible to specify the threshold, in terms of number of choice options, which should trigger the transformation and the type of interactor to use for its replacement. Similar parameters are available for the list boxes. Other parameters concern the maximum number of characters for texts, maximum and minimum dimensions for images. These parameters determine the cost of rendering a presentation. This cost is compared with the overall sustainable cost in the target device, which is given by the screen resolution multiplied by the horizontal and vertical tolerance. The higher the tolerance coefficient values are, the more scrollable the generated user interface will be. This means that end users have the possibility to specify to what extent the adapted content will be scrollable in the target device. The table tolerance provides an additional factor to consider when calculating the sustainable cost. In practise, this means that when there are tables, more scrolling will be acceptable before deciding to split the presentation.

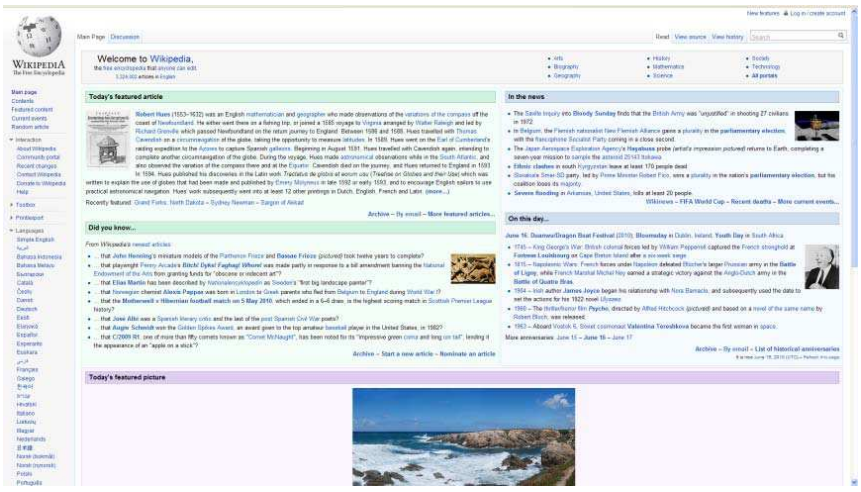

Fig. 4. An example application: Wikipedia.

Figure 4 shows the structure of the user interface of a well-known application, Wikipedia, and next we see how the splitting changes depending on the customization parameters specified. In next Figures we show two example configurations, which mainly differ for the scrolling to avoid parameter (in one case is vertical and in the other is horizontal) and the coefficients for display tolerance (in one case they are 20 and 80, in the other one they are 20, 500).

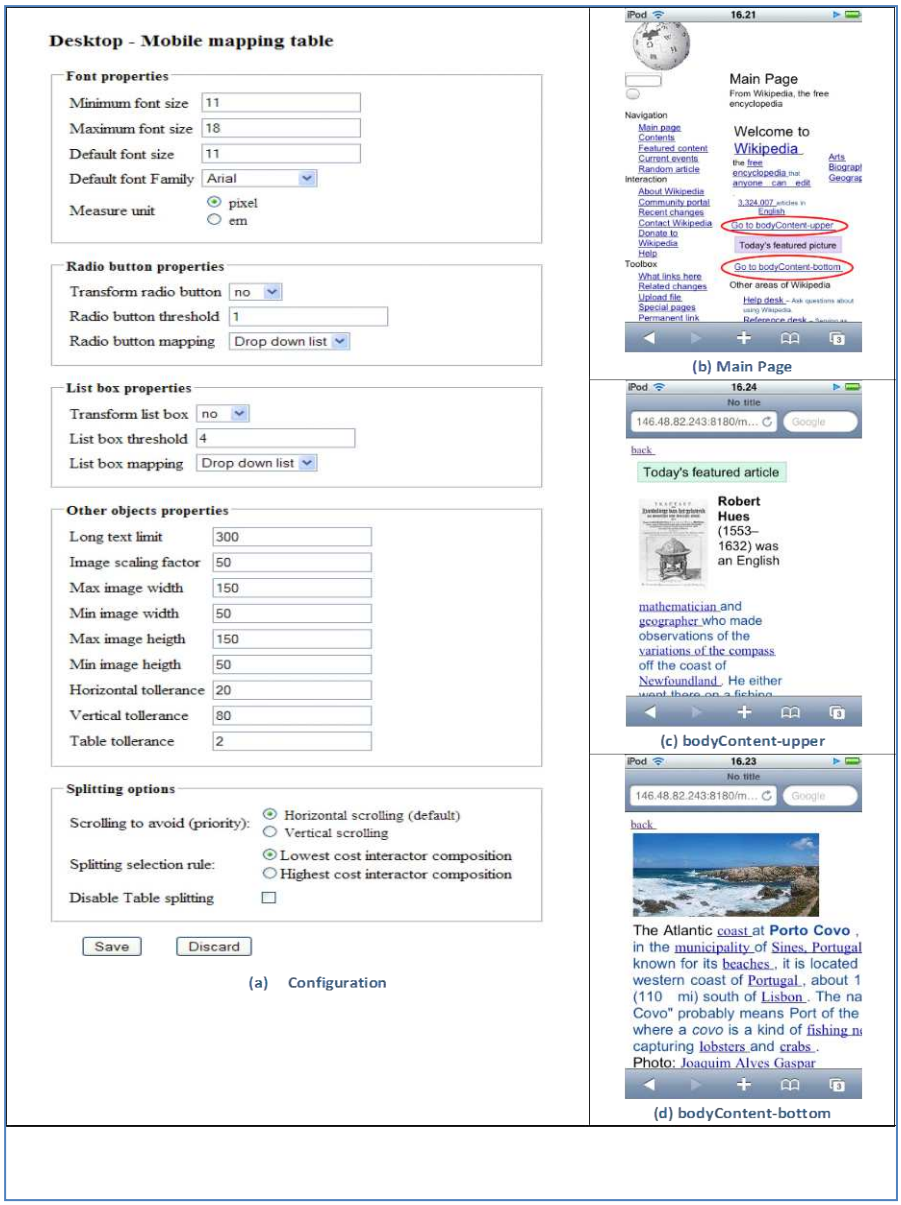

Fig. 5. First Example of Adaptation Configuration and Associated Results.

The customization interface also allows the user to set the priority of the type of scrolling (horizontal or vertical) to avoid and the algorithm splitting policy to apply. In this way, we obtain the specification of user preferences regarding adaptation, which can also be reused for other applications more easily than solution such as collapse-to-zoom [15], where the user can express preferences only associated to a given application.

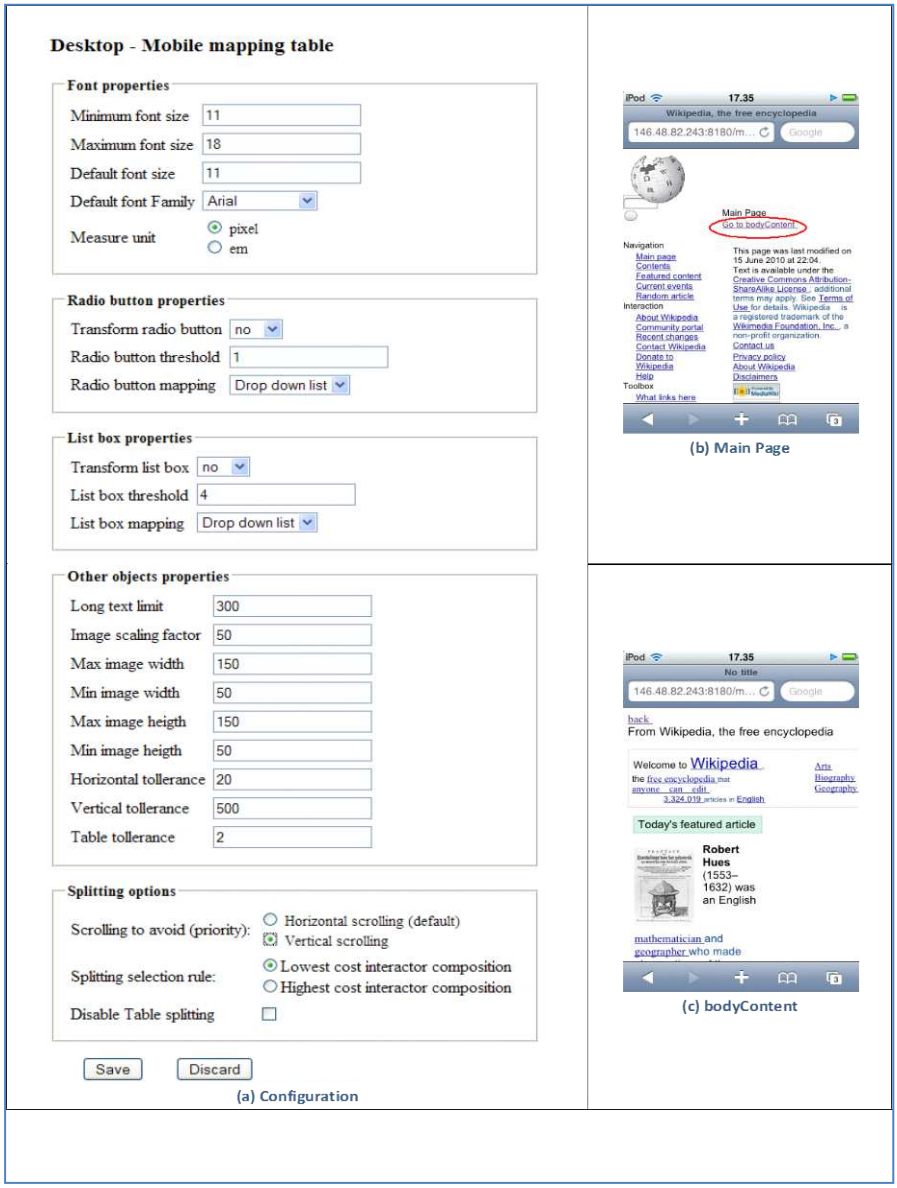

Fig. 6. Second Example of Adaptation Configuration and Associated Results.

Then, we can see for each configuration the resulting adapted mobile pages. In the first case the main page is split into three mobile pages (Figure 5). In the first mobile presentation we have highlighted the automatically generated links to the other mobile pages. In the second case (Figure 6), only two pages are generated from the splitting. This is because in order to fit with the vertical scrolling was sufficient to cut only one big element, which referred to the main central content part.

Please note that the results of the adaptation applied to Web sites such as Wikipedia can change depending on the change of the actual content, which continuously varies in such sites.

An Example Application

In order to better understand how our approach works we can consider an example desktop Web application (see Figure 7). For the sake of clarity we do not use a particularly complex example.

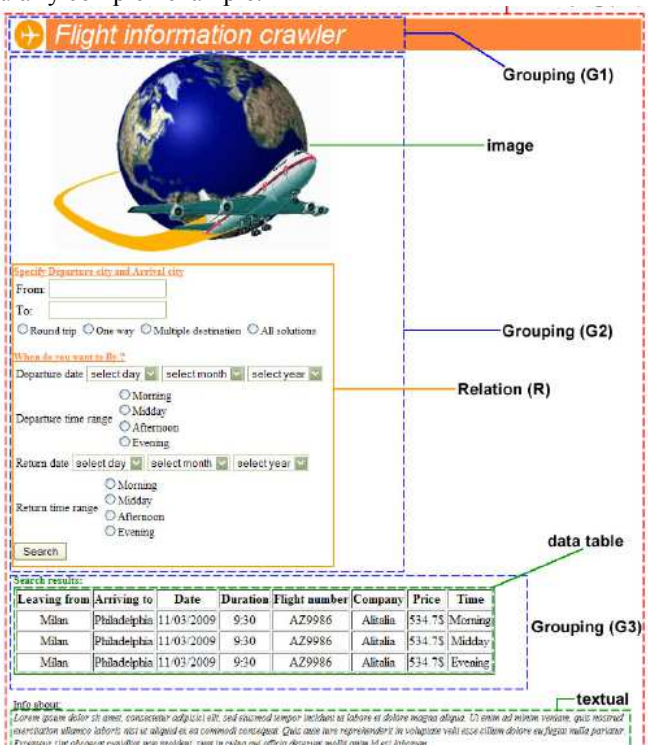

Fig. 7. An example user interface

When the reverse engineering module performs the analysis of this page code, it builds the corresponding logical description (which is highlighted in the Figure). At the first level it identifies a group (G1) associated with the header, a group (G2) associated with the central part, and one group (G3) mainly associated with the data

table. Lastly, a final long text at the bottom is identified. Recursively it identifies the elements included in each group. The top group is composed of an image and some text, the central group is composed of an image and a form, the bottom group is composed of text and a data table. The form is then composed of a number of interactive elements and texts. Now, let us assume that the following parameters have been specified to configure the adaptation process:

Minimum font size = 10px
Maximum font size = 18px
Max image width = 200px
Max image height = 150px
Horizontal tolerance = 10%
Vertical tolerance = 10%
Radio button transformation = yes
Radio button threshold = 3
Radio button mapping = drop down list
Scrolling to Avoid = horizontal
Interactor composition to cut = highest
Long text limit = 300

According to the algorithm previously described, first some basic textual content adaptation is performed. For example, the text “Flight information crawler", contained in Grouping G1, has a size (33px) greater than the value specified in the parameter maximum font size, and consequently is reduced to this limit.

Then, the algorithm calculates the costs in terms of screen consumption of the basic interface elements, and then consequently calculates the costs of the higher elements in the logical structure.

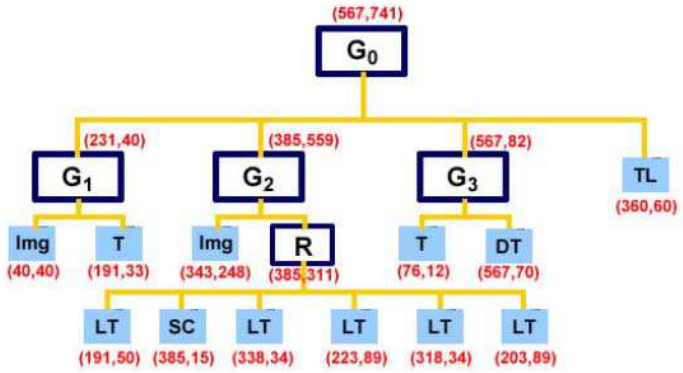

Fig. 8. The costs of the example

Figure 8 shows the resulting costs. For each element a pair of values is provided indicating both the horizontal and the vertical costs. If we consider the specified values for horizontal and vertical tolerance and the resolution of the current device

(360x480 pixels), the maximum sustainable horizontal cost would be 396 pixel, and the vertical 528 pixels. If we look at the overall page cost, given by the cost of G0, we can notice that it is higher than the sustainable cost and consequently the adaptation transformation should move on to the next phase, which involves adaptation of the user interface elements. In particular, in this case we have:

- *The transformation of long texts*, since G0 contains a text longer than 300 characters, the text is split into two parts, one reachable only on request through a link;
- *The transformation of images*, the image contained in the G2 grouping is larger than the limits indicated by max image width and max image height, thus it is scaled from 343x248 pixel to a resolution of 198x143 pixels.
- *Conversion into equivalent interactors*, the radio buttons (an example is the interactor SC in the Figure) are transformed according to the adaptation parameters that indicate that radio buttons be converted into drop-down menus when there are more than three options.
- *Reduction of space taken up by data tables*, the data table DT, contained in the Grouping G3, is reduced by decreasing the size of all the texts contained in it.

Figure 9 shows the updated costs in the user interface versions with the elements adapted as described. It is possible to note that even the resulting overall cost is still too large for the target device. Thus, the phase dedicated to page splitting is activated.

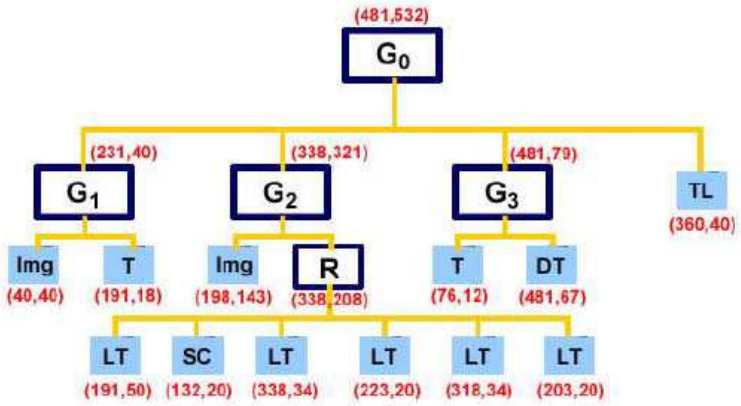

Fig. 9. The updated costs of the example

As described previously the splitting algorithm is driven by two parameters: Scrolling to Avoid and Interactor composition to cut. In our example the first one is set to *horizontal*, and the second one to *highest*. According to these values, the splitting algorithm looks for the element with the highest cost, which is suitable to

avoiding horizontal scrolling. In this case it is the data table DT in Figure 9. According to the rules previously introduced the table is split in such a way as to allocate to a newly generated mobile presentation the portion exceeding the horizontal limit. Thus, at the end of the first cycle the algorithm produces two newly generated additional mobile presentations: one for the excessive table portion and one for the excessive text (see Figure 10).

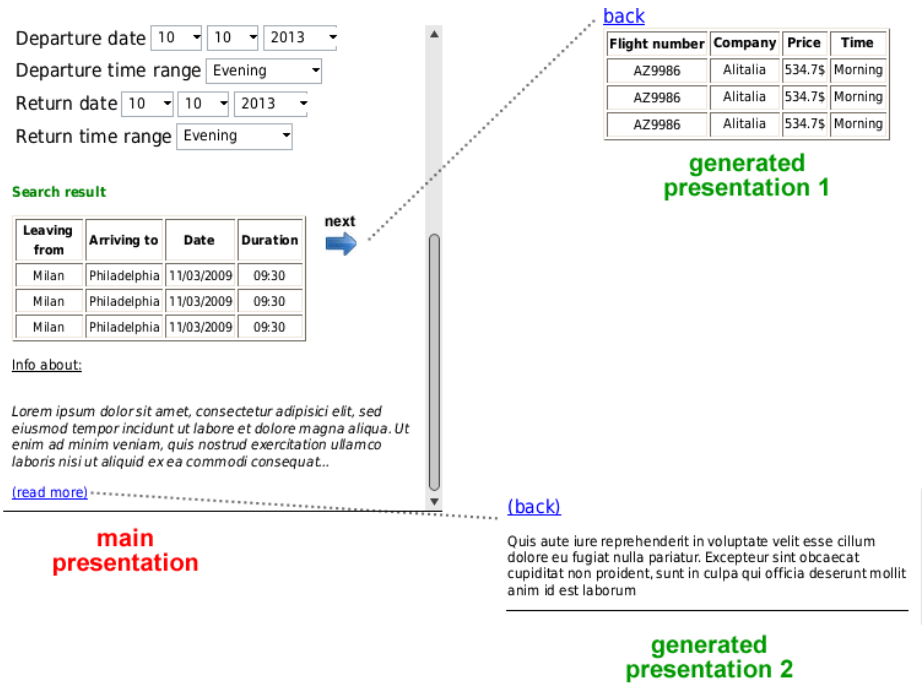

Fig. 10. The presentation generated by the semantic redesign

Comparison with Other Approaches

We have conducted a study comparing our tool, in terms of adaptation results, with two publicly available tools for desktop-to-mobile adaptation: Mowser (http://mowser.com) and Skweezer (http://www.skweezer.com). Figure 11 shows an example form interface adapted using the three systems.

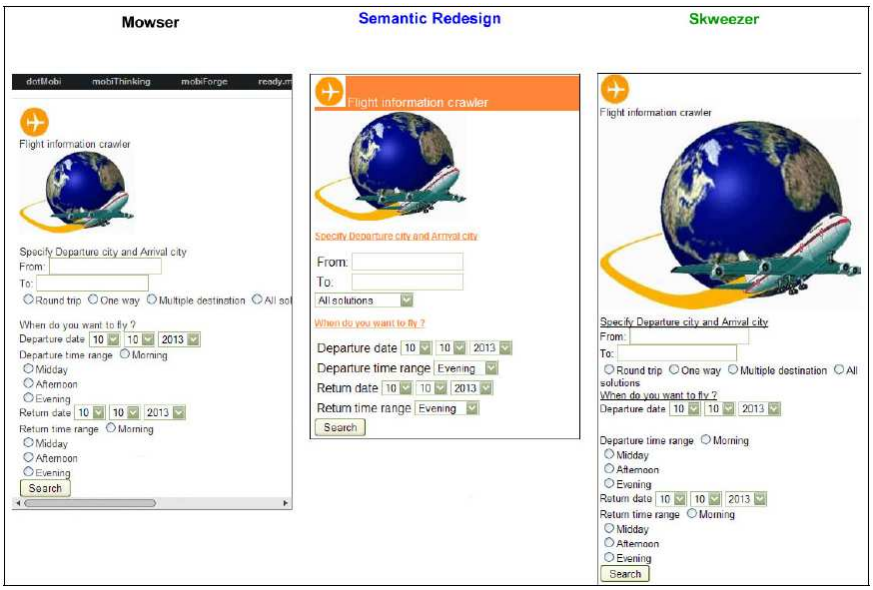

Fig. 11. Form adaptation comparison

By comparing the three versions we have noted that Mowser resizes only the images larger than 150 pixels, ignores style sheets and text attributes indicated in the pages because it associates them with predefined sheets. It provides no particular support for long texts, tables, or change of interactors. In addition, it aims to reduce vertical use of screen space, but this is obtained by requiring users to perform considerable horizontal scrolling.

Squeezer follows a different policy. It reduces the image quality but it does not change their dimensions. Like Mowser, it ignores the colours and the properties specified by the style sheets but it preserves some HTML tags (<b>, <i> and <u>) for the text formatting. Also Squeezers does not support long text transformations, table management (see Figure 12), or interactor changes. Differently from Mowser, Squeezer aims to reduce horizontal scrolling, which implies increasing the vertical one. It also aims to reduce the page download time by reducing the size of its content in terms of bytes.

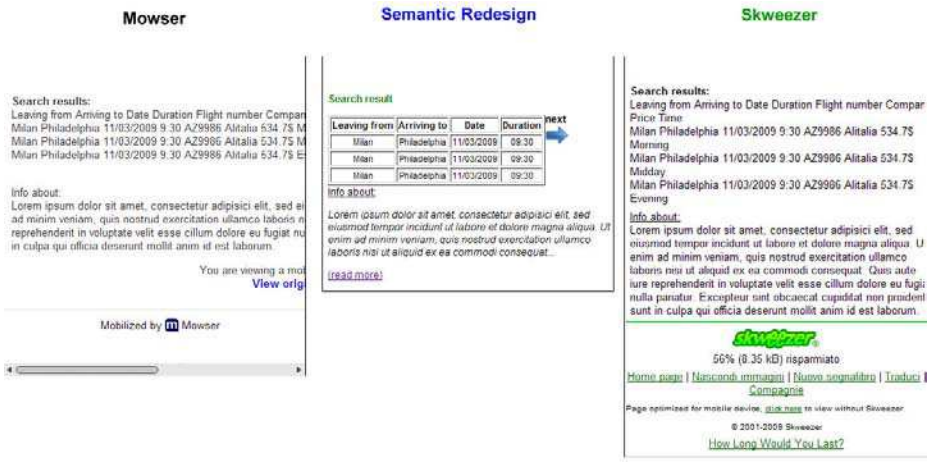

Fig. 12. Table adaptation comparison

The results of this comparison were encouraging because our tool has shown to be more flexible. Indeed, it allows end users to customize the adaptation parameters and is able to adapt a higher number of interface element types than the other two tools (e.g. tables and long texts do not receive specific adaptation transformations with the other two tools).

Conclusions and Future Work

Ubiquitous environments call for adaptive systems in order to adapt to the varying interaction resources. Model-based approaches can provide useful support in this context. We propose a solution for desktop-to-mobile adaptation of Web user interfaces, which overcome limitations of previous ones.

The solution is able to dynamically handle Web pages and build the corresponding logical description through a reverse engineering module able to analyse all the HTML and CSS associated tags. In the adaptation interface elements can be replaced with others that are semantically equivalent but require less screen space. The scripts are preserved in the adapted version. Content such as Flash, Java applets are not currently adapted.

In addition, there is a need for providing users with more control on ubiquitous interfaces, according to the end-user development paradigm. In this paper we have presented a solution that also allows end-users to customize the desktop-to-mobile adaptation in order to change the results that can be obtained by automatic user interface generation.

We plan to further extend this work in various directions. The customization user interface can be improved in order to make the effects of the various customization parameters more understandable. In addition, in this work we have considered only desktop-to-mobile adaptation but other types of transformations can benefit from the approach proposed, e.g. graphical-to-vocal adaptation.

Acknowledgments

This work has been partially supported by the EU ICT STREP Project OPEN (http://www.ict-open.eu/)

References

1. Berti, S., Paternò, F., Santoro C., “Natural Development of Ubiquitous Interfaces”, Communications of the ACM, September 2004, pp.63-64, ACM Press.
2. Bickmore T. et al. Web page filtering and re-authoring for mobile users. Computer Journal special issue on Mobile Computing, vol. 42(no. 6):pp. 534-546, 1999.
3. Calvary, G., Coutaz, J., Bouillon, L., Florins, M., Limbourg, Q., Marucci, L., Paternò, F., Santoro, C., Souchon, N., Thevenin, D., and Vanderdonckt, J. 2002. The CAMELEON reference framework. CAMELEON Project. Deliverable 1.1
4. Buyukkokten O., Kaljuvee O., Garcia-Molina H., Paepcke A., Winograd T. ,. Efficient web browsing on handheld devices using page and form summarization. TOIS, pages 82-115, 2002.
5. Chu H., Song H., Wong C., Kurakake S. and Katagiri M., Roam, a seamless application framework, Journal of Systems and Software, Volume 69, Issue 3, 2004, pp. 209-226, Elsevier.
6. Laakko T. Hiltunen T. Adapting web content to mobile user agents. IEEE Internet Computing, Vol. 9(no. 2):46-53, March-April 2005.
7. Kaikkonen A. Roto, V. Perception of narrow web pages on a mobile phone. Proc. Human Factors in Telecommunications, 2003.
8. Lieberman, H., Paternò, F., Wulf W. (eds), End-User Development, Springer Verlag, ISBN-10 1-4020-4220-5, 2006.
9. Melchior, J., Grolaux, D.,Vanderdonckt, J.,Van Roy, P., A Toolkit for Peer-to-Peer Distributed User Interfaces: Concepts, Implementation, and Applications, pp. 69.78, EICS’09, July 15–17, 2009, Pittsburgh, Pennsylvania, USA.
10. Ohnishi K., Tajima K.. Browsing large html tables on small screens., ACM Symposium on User Interface Software and Technology (UIST '08).
11. Paternò, F., Santoro, C., Scorcia A Automatically Adapting Web Sites for Mobile Access through Logical Descriptions and Dynamic Analysis of Interaction Resources. AVI 2008, Naples, May 2008, ACM Press, pp. 260-267.
12. Paternò F., Santoro C., Spano L.D., "MARIA: A Universal Language for Service-Oriented Applications in Ubiquitous Environment", ACM Transactions on Computer-Human Interaction, Vol.16, N.4, November 2009, pp.19:1-19:30.
13. Sottet J., Ganneau V., Calvary G., Coutaz J., Demeure A., Favre J., Demumieux R.: Model-Driven Adaptation for Plastic User Interfaces. INTERACT (1) 2007: 397-410.
14. Uemukai T., Nishio S.. Y. Arase, T. Hara. Opa browser: a web browser for cellular phone users. ACM Symposium on User Interface Software and Technology (UIST '07), pages 71-80.
15. Wang-Wei-Ying C., Baudisch P., Xie X.. Collapse-to-zoom: Viewing web pages on small screen devices by interactively removing irrelevant content. ACM Symposium on User Interface Software and Technology (UIST '04), pages 91-94, October 2004.
16. Weiser M., "The Computer for the 21st Century" - Scientific American Special Issue on Communications, Computers, and Networks, September, 1991.