Chapter

# Technological platforms, convergence and adaptation of interactive contents

Fabio Paternò

fabio.paterno@isti.cnr.it

ISTI-CNR, Pisa, Italy

**Abstract**
Our lives are becoming a multi-device experience where people are surrounded by different types of devices (e.g. cell phones, PDAs, desktop computers, digital television sets, intelligent watches, and so on) through which they can connect to networks in different ways. Most of them are mobile personal devices carried by users moving freely about different environments populated by various other devices. Such environments raise many issues for designers and developers, such as the need to obtain interactive contents able to adapt to the interaction resources of the available devices. The main objective of this book chapter is to allow readers to gain knowledge in methods and tools for the design of multi-device interactive services that can support designers and developers in addressing a number of the issues raised by multi-device environments.

**Introduction**
Multi-device interactive services are acquiring an increasing importance, however they raise a number of issues with regard to obtaining adaptation to the changing interaction resources. Indeed, the diversity in features of the potential devices, such as different screen size, interaction capabilities, processing and power supply, can make a user interface developed for a desktop unsuitable for a PDA and vice versa. For example, an interface layout designed for a desktop platform does not fit in the smaller screen of a PDA, or a graphic interface running on a desktop system must be transformed to a voice interface when the application migrates to a car. Thus, a user interface needs to adapt to the different features of the target platform taking into account usability principles.
This chapter first provides an introduction of the main requirements that approaches to interactive content should satisfy. Next, there is a discussion of why and how model-based approaches can be useful for this purpose followed by a description of authoring environments that can be helpful for this purpose. Then, the possible solutions for adaptation at run-time are discussed with an indication of the more promising approaches. The last part is dedicated to migratory interactive services that are acquiring an increasing interest. Lastly, some conclusions are drawn along with indications of interesting possible further developments in the area.

**Requirements in Device Adaptation**
The device choice has an influence on the possible tasks to accomplish and how the structure of such tasks can vary in terms of possible secondary tasks, inter-task temporal relations, and content required depending on the device.
One issue is that one-fits-all does not work. This means that it is not meaningful to try to support all tasks for all platforms. The big push for cross-platform design comes with the advent of mobile devices, which people can use when they are on the move. This means that there is a clear

distinction regarding what it is meaningful to do with a desktop and with a mobile device. For example, people can use a desktop to compare airfares and make reservations, while the mobile device can be used to check the real-time status of a particular flight. Likewise, the desktop is well-suited to reading a movie review and/or watching a trailer but not to purchasing a cinema ticket to avoid the line. There are many tasks that are not suitable at all for a mobile device. For example, currently in Europe with the advent of UMTS there is an interest in proposing football matches over phones. This seems senseless even with the last generation mobile phones, which have large displays and better connectivity. People like watching football matches but generally only when they are comfortable sitting and watching a large display in front of them. This allows them to appreciate how the footballers play, the details of the action, the tactics adopted and so on. Few people would ever do this through a mobile phone, even if it is technically possible, because of the small display and discomfort of using it on the move. Also considering that a football match lasts 90 minutes, this would be a terrible experience. What they could appreciate with a mobile device is a different task: receiving real-time updates regarding the match score. Another useful option is to have just the vocal description of the match that can be followed while driving the car or on the move. Thus, they could do this in parallel with other activities.

In general, when a multi-platform application is considered, it is important to understand what type of tasks can actually be performed in each available platform. There are various possibilities:

- *The same task can be performed on multiple platforms in the same manner* (there may be only some changes in attributes of the user interface objects from platform to platform). This is the case of tasks whose presentation remains mostly unchanged on different platforms: an example is when in a museum application textual links are provided to access general information about the museum (how to reach, timetable, etc.).
- *Same task on multiple platforms but with different user interface objects*. An example of this case is highlighted in Figure 1. In both systems users can select a geographical area (e.g. London/South East, Devon/Cornwall, etc). However, while in the desktop system a large, coloured interactive map of the museum is at the users' disposal, in the phone, because of its limited capabilities, a text link is available for each area.

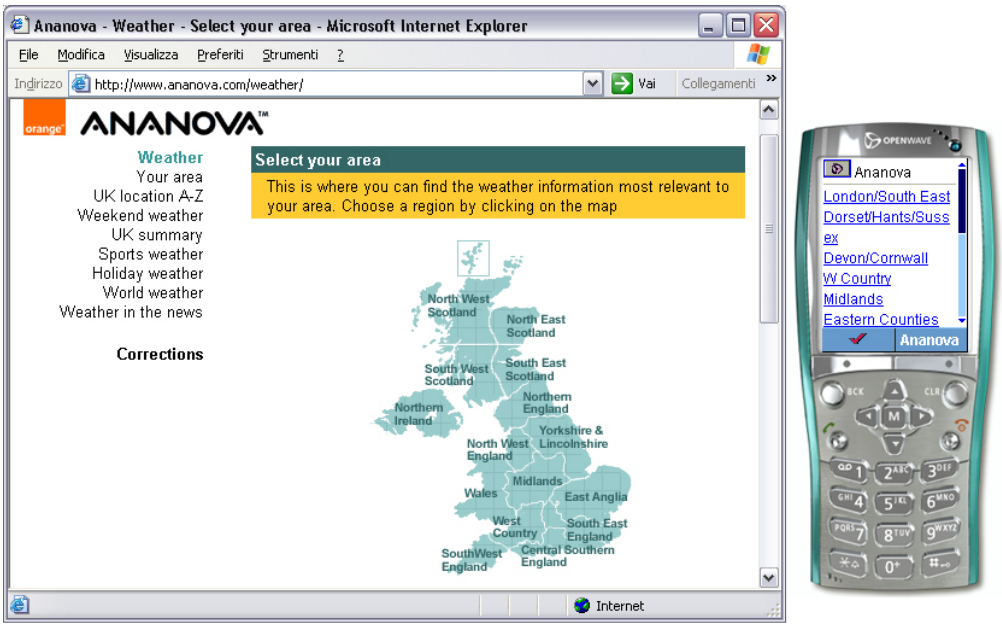


*Figure 1:* Example of same task and different user interface objects.

- *Same task on multiple platforms but with different domain objects*. This means that during the performance of the same task different sets of domain objects are manipulated. An example of this is presentations of different information on a desktop system and a mobile phone for a museum application: while in the desktop system it is possible to access a wider set of domain elements (title, image type, description, author, material, and date of creation), the mobile interface supports access to only an image (which is a low resolution image just to give users a rough idea of what the work of art is), along with the indications of the title and associated museum section.
- *Same task on multiple platforms but with different task decomposition.* This means that the task is sub-divided differently, with different sets of sub-tasks, depending on the platform. An example of this possibility is displayed in the Figure 2 that shows how differently the task access work of art is supported in a desktop and in a mobile device. In the desktop system, the users can accomplish additional sub-tasks, which are not supported in other systems. An example concerns the possibility of reading reviews of a particular work of art, which is a lengthy information-processing task that users can perform satisfactorily when sitting in front of a desktop computer, but which is not appropriate with handheld devices.

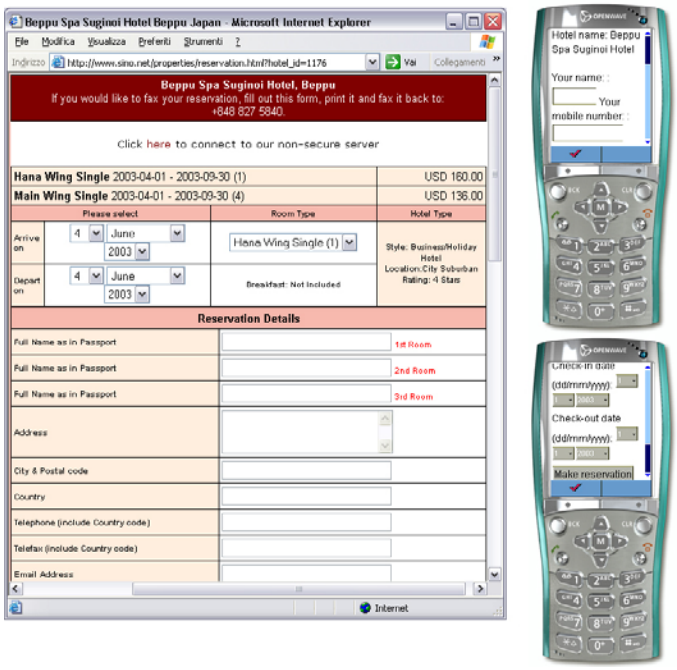


*Figure 2:* Example of same main task and different task decomposition.

- *Same task on multiple platforms but with different temporal constraints.* In this case the difference is in the temporal relationships among the subtasks. With reference to the museum application, consider the case of users who wish to electronically reserve their tickets for a particular visit in order to avoid queues. In both systems they have to provide personal information. However, while in the desktop system they are free to choose the order to follow for filling in the various fields, within the phone application they are constrained by the mobile interface to follow a sequential order.

- *Dependencies among tasks performed on different platforms.* An example of this can be found when the users have to reserve their flight tickets. Through the desktop system users can access, compare and contrast the different options about the best time for the flight ticket. Once they have selected their preferences and entered personal data, the system

automatically enables their mobile phone to access real-time data regarding their flight, which ca be useful for example to know whether it is on time (see Figure 7). Capturing this type of task relationships is particularly important when there is some task relevant to only a particular platform and that affects the performance of another task through a different platform. A typical situation occurs when users physically visit the museum and simultaneously annotate the most interesting works of art on the PDA. When they arrive home they would appreciate being able to receive information regarding such works first during their access to the museum web site through a desktop system.

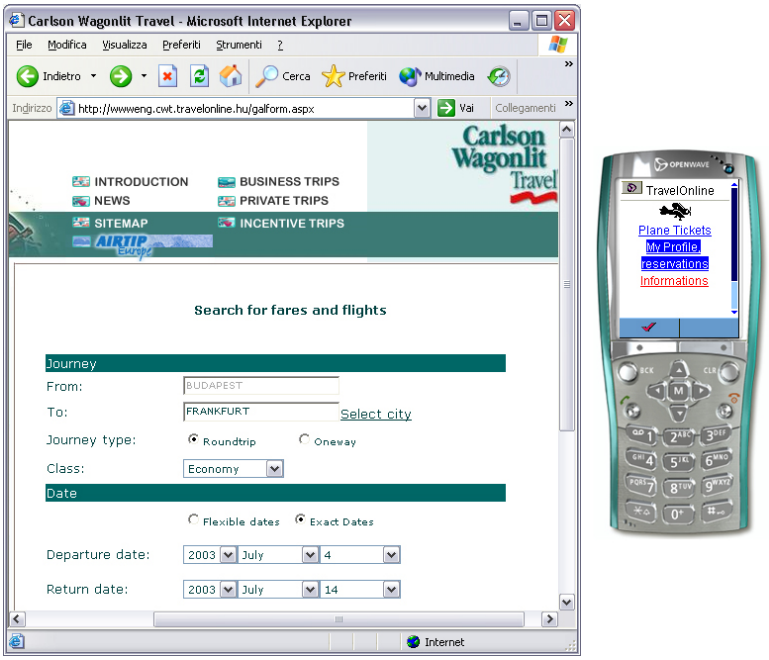


*Figure 3:* Example of dependencies among asks performed through different platforms.

## Model-based Approaches

In this section we provide an overview of the results that can be obtained through model-based approaches when multi-device interfaces, even using different modalities, are considered, and will link up the discussion to projects currently underway. Indeed, as Myers, Hudson, and Pausch (2000) indicated while discussing the future of user interface tools, *the wide platform variability encourages a return to the study of some techniques for device-independent user interface specification, ….. Then, the system might choose appropriate interaction techniques taking all of these into account.* The basic idea is that instead of having separate applications for each device that exchange only basic data, there is some abstract description and then an environment that is able to suggest a design for a specific device that adapts to its features and possible contexts of use. Thus, a key aspect is to be able to have different views on interactive systems, each view associated with a different abstraction level. With the support of tools, XML-based languages and transformations, it is possible to move from one level to another and tailor a description for one abstraction level to one more refined for the target interaction platform.

The model-based community has long discussed such possible description levels, see for example (Szekely, 1996). It is possible to have various viewpoints on an interactive system. Such viewpoints differ for the abstraction levels (to what extent the details are considered) and the focus (whether the task or the user interface is considered). Such abstraction levels are:

- *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 (Paternò and Leonardi, 1994). 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.

The main issue underlying the last generation of model-based approaches is the design of multi-device interfaces. In current practise the design of multi-platform applications is often obtained through the development of several versions of the same applications, one for each platform considered. Then, such versions can at most exchange data. This solution with no tool support is rather limited, because it implies high implementation and maintenance costs. Thus, there is a need for authoring environments able to support the development of multi-device interfaces by providing design suggestions taking into account the specific features of the devices at hand.
LiquidUI is an authoring environment whose main goal is to reduce the time to develop user interfaces for multiple devices. It is based on the User Interface Markup Language (UIML) (Abrams et al., 1999), a declarative language that then can be transformed in Java, HTML, and WML through specific rendering software. A UIML program, with its generic vocabulary, is specific to a family of devices (such as the desktop family, the PDA family, the WAP family). There is a transformation algorithm for each family of devices. For example, using a generic vocabulary for desktop applications, the developer can write a program in UIML once and have it rendered for Java or HTML.

Another approach is UsiXML (Limbourg and Vanderdonckt, 2004), the semantics of the UsiXML models are based on meta-models expressed in terms of UML class diagrams, from which the XML schema definition are derived. Right now, there is no automation between the initial definition of the semantics and their derivation into XML schemas. Only a systematic method is used for each new release. UsiXML aims to address the development of user interfaces for multiple contexts of use and has the advantage of providing a graphical syntax for a majority of constituent models. However, UsiXML renderers are still at the development stage.

TERESA (Mori, Paternò, and Santoro, 2004) is intended to provide a complete semi-automatic environment supporting a number of transformations useful for designers to build and analyse their design at different abstraction levels, including the task level, and consequently generate the concrete user interface for a specific type of platform. Currently, the tool supports generation of user interface implementations in XHTML, XHTML mobile device, VoiceXML, multimodal user interfaces in X+V and Java for the digital TV. The tool is able to support different level of automations ranging from completely automatic solutions to highly interactive solutions where designers can tailor or even radically change the solutions proposed by the tool. The last version of the tool supports *different entry-points*, so designers can start with a high-level task models but they can also start with the abstract user interface level in cases where only a part of the related design process needs to be supported. With the TERESA tool, at each abstraction level the designer is in the position of modifying the representations while the tool keeps maintaining forward and backward the relationships with the other levels thanks to a number of automatic features that have been implemented (e.g. the possibility of links between abstract interaction objects and the corresponding tasks in the task model so that designers can immediately identify their relations). This is useful for designers to maintain a unique overall picture of the system, with an increased consistence among the user interfaces generated for the different devices and consequent improved usability for end-users. Even recent W3C standards, such as XForms (XForms, 2004), have introduced the use of abstractions similar to those considered in the model-based community to address new heterogeneous environments.

**Authoring Multi-device User Interfaces**

The concepts discussed in the previous section can be incorporated in authoring environments for multi-device interfaces able to deal with a variety of platforms with different modalities (such as graphical and vocal interfaces, digital TV, tilt-based interaction, …). Examples of such tools will be discussed (e .g. Multimodal TERESA).

Once we have identified the tasks that are meaningful to support for each platform then we have to identify how they performance can vary according to the target platform. The idea is to first identify potential interactors in terms of their semantics (how they can change the state of the application) and then to generate corresponding interfaces depending on the platform features. All devices belonging to a given platform will receive an interface with consistent implementation.

An abstract user interface is structured into presentations and connections indicating how it is possible to move from one presentation to another. Each presentation is structured into interactors and composition operators. We have defined a number of composition operators, which aim to capture communication effects that often designers want to achieve when they structure their user interfaces. The purpose of the composition operators is to indicate how to put together interactors. Each composition operator is associated with a communication goal. Depending on such goals, different implementation techniques will be used to support the composition operator. Figure 4 shows an example of a Web page taken from a frequently accessed Web site. We can note how the designer used various techniques to highlight groups of related interface elements. On the top there

are elements that are ordered according to the potential user interest. Some elements are grouped using implementation techniques such as same background, same structure, bullets and so on. There are elements that are related to the rest of the Web site, such as the search element. Other elements are highlighted using large image and fonts because they are considered important.
In general, the composition operators can involve several interactors or even compositions of interactors. In addition, their definition is modality-independent. They are:

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

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). 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.


*Figure 4:* Web page with indication of some associated communication goals.

The Multimodal TERESA authoring environment allows designers and developers to start from two possible points: the task model description or the abstract interface description. In both cases they have to specify the target platform (in the current tool version either multimodal desktop or multimodal PDA). If they start with the task model then the tool automatically generates the corresponding abstract interface. As you can see in Figure 5, the main area is mainly divided into four parts: the top-left dedicated to the list of presentations composing the user interface, the bottom-left indicating the connections defining how it is possible to move from one presentation to another, the top-right indicating the abstract description of the currently selected presentation and the bottom-right part displays the description of the possible concrete implementation of the currently selected element in the abstract part.
The concrete part has three tabbed panes, one for the concrete graphical attributes, one for the concrete vocal attributes and one to specify how to compose the multimodal attributes.
For example, if we consider the single selection interactor used to indicate the time of a cinema reservation, then the tool as a first suggestion for an implementation in a graphical+vocal desktop

interface would propose that the input be equivalent (either graphical or vocal) and the prompt and feedback both redundant. Then, in the vocal section there would be an indication of the corresponding label, and the associated vocal message and feedback, in addition to the definition of the possible choice elements. The graphical part indicates the interaction technique for implementation (e.g. a radio-button), and the corresponding label and elements. The tool keeps information in the graphical and vocal parts consistent. So, if the designer indicates five possible choice elements in the vocal part, then this is indicated when the graphical part is accessed as well. Likewise, in the case of a text output, if the corresponding multimodal property is complementarity, then different texts can be specified for vocal and graphical rendering, while if the multimodal attribute is redundant, then the text modified in either part will be updated for the other one as well.

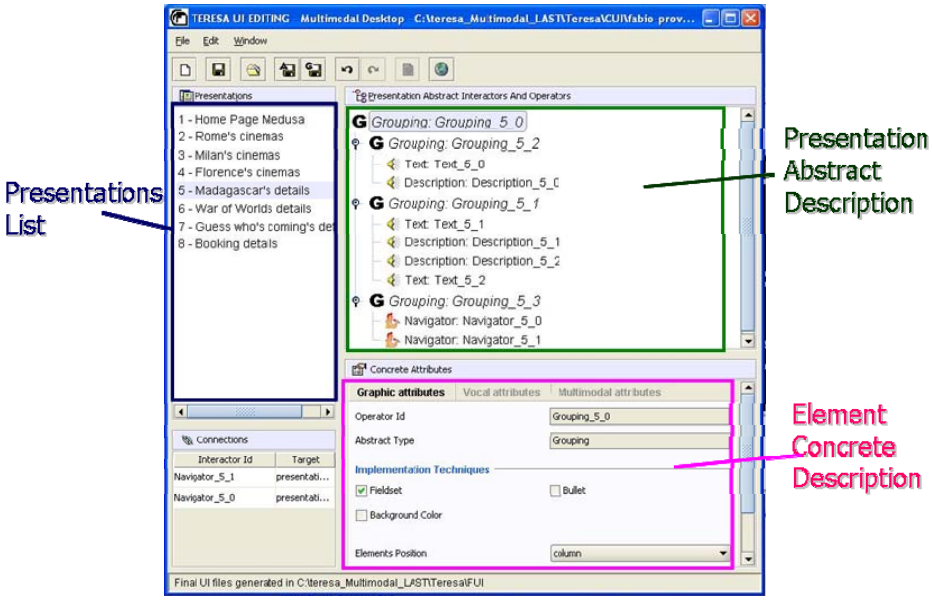


*Figure 5:* The MultiModal TERESA Environment.

**Run-time adaptation to the device**

This section is dedicated to run-time support for multi-device environments. Different types of solution and associated software architectures will be first introduced. Issues and solutions for automatic transformation from desktop interfaces to different platforms (especially mobile ones) will be discussed, showing how presentation, navigation and content can be transformed and the usability issues to address in this process. I will show examples of results that can be obtained by tools provided by main software companies such as Google, Nokia, Microsoft, Opera, along with research results from various groups (including mine).

The increasing availability of mobile devices has stimulated interest in tools for adapting the large amount of existing Web applications originally developed for desktop systems into versions that are accessible and usable for mobile devices. This adaptation process implies transforming various aspects:

- the *presentations*, the perceivable aspects, including choice of media and interaction techniques, layout, graphical attributes, etc.;
- the *dynamic behaviour*, including navigation structure, dynamic activation and deactivation of interaction techniques;
- the *content*, including text, labels, images.

In carrying out this adaptation it is important to consider the main characteristics of the target platform, in this section the mobile device will be mainly considered. By platform we mean a group

of devices that share similar characteristics (such as the desktop, the mobile, the vocal, …). For example, one aspect to consider is that often mobile devices have no pointing device, thus users have to navigate through 5-way keys, which allow them to go left, right, up, down and select the current element. There are also softkeys, which are used to activate commands, but their number and purpose vary depending on the device. In addition, text input is slow and users often have to pay to access the information, and thus prefer short sessions.

Regarding the devices, the need for describing their features derives from the increasing availability of various types of interactive devices. Thus, in order to have applications able to adapt to their features there should be a way to represent them. This is particularly important in the area of mobile phones, in which the possible characteristics are the most variable. The generic Composite Capabilities/Preference Profiles (CC/PP) framework (www.w3.org/2001/di/) provides a mechanism through which a mobile user agent — a client, such as a browser, that performs rendering within a mobile device — can transmit information about the mobile device. It is based on RDF and aims to provide a firm foundation for UAProf. The user agent profile (UAProf; www.openmobilealliance.org/release_program/uap_v20.html) is an application of the CC/PP framework. It includes device hardware and software characteristics, information about the network the device is connected to, and other attributes. It is possible ot identify a device through the header of HTTP requests. All the devices complying with UAProf have a CC/PP description of their characteristics in a repository server, which can be queried for knowing them. The description of the devices is in the *Resource Description Framework* (RDF), language XML-based. When a mobile device sends a request, ity also informs about the URL where its profile is through a specific field in the request called *X-Wap-Profile*. For example, the *x_wap_profile* for a Nokia N9500 is:

*x_wap_profile:***"http://nds1.nds.nokia.com/uaprof/N9500r100.xml"**;

The table below shows an excerpt of a profile for a mobile device:

```
<?xml version="1.0"?>
 <rdf:RDF
       xmlns:rdf= "http://www.w3.org/..."
       xmlns:prf="http://www.openmobilealliance.org/..."
       xmlns:mms="http://www.wapforum.org/..."
       xmlns:pss5="http://www.3gpp.org/...">
       <rdf:Description rdf:ID="Profile">
          ……
            <prf:component>
                 <rdf:Description rdf:ID="HardwarePlatform">
                 ……
                      <prf:PixelAspectRatio>1x1</prf:PixelAspectRatio>
                      <prf:PointingResolution>Pixel</prf:PointingResolution>
                      <prf:ScreenSize>640x200</prf:ScreenSize>
                      <prf:ScreenSizeChar>29x5</prf:ScreenSizeChar>
           <prf:StandardFontProportional>Yes</prf:StandardFontProportional>
                      <prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
                      <prf:TextInputCapable>Yes</prf:TextInputCapable>
                      <prf:Vendor>Nokia</prf:Vendor>
                      <prf:VoiceInputCapable>No</prf:VoiceInputCapable>
                  ……
                  </rdf:Description>
            </prf:component>
```

```
        ……
      </rdf:Description>
</rdf:RDF>
```

In this area another proposal is WURFL (Wireless Universal Resource File) (Passani, 2006), which is an XML configuration file which contains information about capabilities and features of several wireless devices. The main scope of the file is to collect as much information as we can about all the existing wireless devices that access WAP pages so that developers will be able to build better applications and better services for the users. This proposal aims to support Web applications for mobile devices. The goal is programmatically abstract away devices differences, avoid that we need to modify applications whenever a new device ships, avoid that we need to track new devices that ship (particularly those in uninteresting markets). The basic idea is a global database of all devices and their capabilities. Starting from the assumption that browsers are different, but they also have many features in common with one another; browsers/devices coming from the same manufacturer are most often an evolution of the same hardware/software. In other words, differences between, for example, a Nokia 7110 and a Nokia 6210 are minimal; devices from different manufacturers may run the same software. WURFL has created a compact, small, and easy to update matrix. The WURFL is based on the concept of family of devices. All devices are descendent of a generic device, but they may also descend from more specialized families (such as those who use the same browser). Its goal is to overcome some limitations of the UAProf standard developed in the OMA. Uaprof seems to rely too much on someone else's setting up the infrastructure to request profiles. There are cases of manufacturers just associating the profile of a different phones into a new one. The WURFL can be installed in any site and does not need to access device profiles from a repository on the net.

In general there are various approaches to authoring multi-device interfaces:

- Device-specific authoring, which means that a specific version for each target platform is developed separtely. One example is the Amazon Web site, which has a separate version (http://www.amazon.com/anywhere) for mobile devices. This approach is clearly expensive in terms of time and effort.
- Multiple-device authoring, this is similar to the previous one but there is a single application that has separate parts for different platforms. An example is the use of CSS depending on the media.
- Single authoring, in this case only one version of the application is built, which is then adapted to various target platforms. There are two main possibilities for this purpose: either to include the authors' hints or to specify an abstract description, which is then refined according to the target platform. Interesting contributions in this area are PIMA (Banavar et al. 2004) and SUPPLE (Gajos et al., 2005).
- Automatic re-authoring, in this case there is an automatic transformation of a version for a given platform, usually the desktop, into a version for the target platform.

Recently, particular attention has been paid to the latter category, with the aim of obtaining a solution that does not require particular effort in terms of time but is still able to produce meaningful results. Various solutions have been proposed in this category, a first distinction can be made depending on where the re-authoring process occurs: the client device, the application server or an intermediate proxy server. The last solution seems particularly interesting because performing

the transformation on the client device can raise performance issues with limited capabilities devices, while a solution on the application server would require duplication of installations in all the applications of interest. The feasibility of proxy-based solutions is also shown by its widespread use in tools, such as Google for mobile devices (www.google.com/xhtml) (Kamvar and Baluja, 2006), which converts the Web pages identified by the search engine into versions adapted for mobile devices. Indeed, depending on user agent in http request Google search redirects to www.google.com/xhtml in the case of mobile devices. This version (see Figure 6) has radio buttons instead of tabs to access the various sections. There is no advertisement in the XHTML version, the pieces of text for each link are smallers in the XHTML version, the results for the XHTML version do not contain link cached or similar pages and don't indicate the size of the page, and the user can access only the previous and next result page.

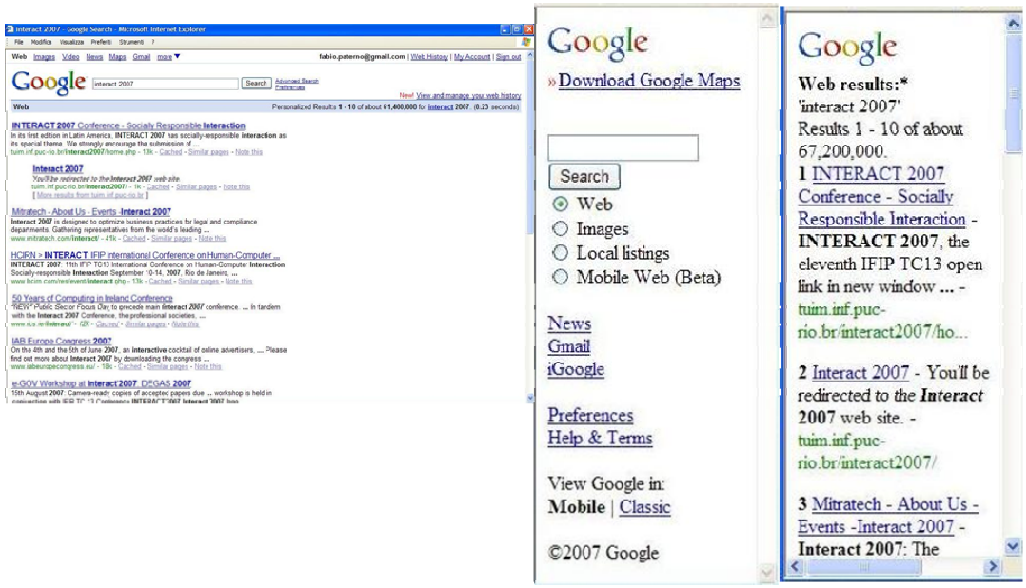


*Figure 6:* The two user interfaces for the Google search engine.

In addition, we think that effective solutions for transforming desktop Web sites for mobile access should be based on semantic aspects. Unfortunately, the semantic Web so far has mainly focused on the data semantics through the use of ontologies and languages that allow for more intelligent processing. It would also be important to consider the semantics of interaction, which is related to the tasks to support in order to reach the users' goals.

Several solutions for automatic re-authoring from desktop-to-mobile have been proposed in recent years. The simplest one just proposes resizing the elements according to the size of the target screen. However, they often generate unusable results with unreadable elements and presentation structures unsuitable for the mobile device. Some solutions such as AvantGo (www.avantgo.com) translate elements and images into other formats, and compress and convert images to match the device characteristics but suffer from similar limitations as the former approach. Thus, research work has focused on transformations able to go further, to modify both the content and structure originally designed for desktop systems to make them suitable for display on small screens. Even in this case various possibilities have been explored. The most common transformation supported by current mobile devices is into a single column (the narrow solution): the order of the content follows that of the mark-up file starting from the top, the images are scaled to the size of the screen, and the text is always visible and the content compacted without blank spaces. It eliminates scrolling in one dimension, though it greatly increases the amount of scrolling in the other dimension. For example Opera SSR (Small Screen Rendering, www.opera.com/products/smartphone/ smallscreen/) uses a remote server to pre-process Web pages before sending them to a mobile device, Web content is

compressed to reduce the size of data transfers (see Figure 7). In general, in this solution content requiring a good deal of space such as maps and tables can become unreadable; and often it is difficult to understand that the corresponding desktop page has changed because the initial part of several desktop pages is indistinguishable.


*Figure 7:* The Example of transformation with OPERSA SSR.

Various approaches have considered the application of information visualization (Spence 2001) techniques to address these issues. Fish-eye representations have been considered, for example Fishnet (Baudish et al., 2004), which is a fisheye Web browser that shows a focus region at a readable scale, while spatially compressing page content outside the focus region. However, generating fish-eye representations in mobile devices can require excessive processing. Overview + detail splits a Web page into multiple sections and provides an overview page with links to these sections. The overview page can be either a thumbnail image, or a text summary of the Web page. Within this approach various solutions have been proposed. Smartview (Milic-Frayling et al., 2002) is a thumbnail view of the original Web page in zoom-out, fitting the screen horizontally. The approach partitions the page in logical regions; when one is selected, content is presented inside the screen space in a detailed view. In Gateway (MacKay et al., 2004) the detailed view uses a focus-plus-context technique, enlarging the selected region above the detailed view. Summary Thumbnails (Lam and Baudish, 2005) uses the same thumbnail approach but the texts are summarized enabling good legibility (fonts are enlarged to a legible size and characters are cropped from right to left until the sentence fits in the available area). The main issue with this type of approach is that it works well in some cases, less in others because they mainly focus on the transformation of specific elements (for example Summary Thumbnail mainly works on text snippets). Another contribution in this area is MiniMap (Roto et al., 2006) a browser for Nokia

6600 mobile phones developed at Nokia Research. It removes the need for horizontal scrolling to read text and provides enough contextual information to give an idea of the page structure and the current location without destroying the original page layout. The user interface is organised in such a way that text size should not exceed the screen space and provides an overview+detail representation. The overview is given by an area dedicated to showing where the current mobile page is located in the original desktop page. However, this solution is effective only with mobile devices with relatively large screens.

We believe that model-based approaches (Paternò, 1999) (Szekely, 1996) can provide a more general solution to adaptation to the interaction device. They are based on the use of logical descriptions that capture the main semantic aspects of the user interface and hide low-level details. Some first studies of how to apply them in this context have already been proposed (Bandelloni and Paternò, 2004), (Eisenstein et al., 2001) (Florins and Vaderdonckt, 2004). These interesting works were useful to provide solutions to specific issues raised by supporting interactions of mobile users, but they did not address the issue of providing a general solution for taking Web sites originally developed for desktop systems and dynamically transforming them into accessible and usable versions for mobile devices while users are accessing them. Another interesting application of model-based approaches in this area is PUC (Nichols et al., 2002) which dynamically generates user interfaces for mobile devices able to control a domestic appliance starting with its logical description. A novel solution (Bandelloni et al., 2007) for a different domain (Web applications), which is characterised by wider variability in terms of content and tasks to support, is based on the use of a migration/proxy server able to support both adaptation and state preservation across multiple devices.

**Migratory User Interfaces**

This sections discusses how mobile users can be supported in multi-device environments. To this end, distributed and migratory interfaces are introduced.

One important aspect of ubiquitous environments is to provide users with the possibility to freely move about and naturally continue the interaction with the available applications through a variety of interactive devices (i.e. cell phones, PDAs, desktop computers, digital television sets, intelligent watches, and so on). Indeed, in such environments one big potential source of frustration is that people have to start their session over again from the beginning at each interaction device change. Migratory interactive services can overcome this limitation and support continuous task performance. This implies that interactive applications be able to follow users and adapt to the changing context of use while preserving their state. Migratory interfaces are interfaces that can transfer among different devices, and thus allow the users to continue their tasks. This definition highlights important concepts: task performance continuity, device adaptation and interface usability. Task performance continuity means that when migration occurs users do not have to restart the application on the new device, but they can continue their task from the same point where they left off, without having to re-enter the same data and go through the same long series of interactions to get to the presentation they were accessing on the previous device. General solutions for migratory interactive services can be obtained by means of addressing three aspects: adapt and preserve the state of the software application parts dedicated to interacting with end users; support mechanisms for application logic reconfiguration; and define suitably flexible mechanisms from the underlying network layers.

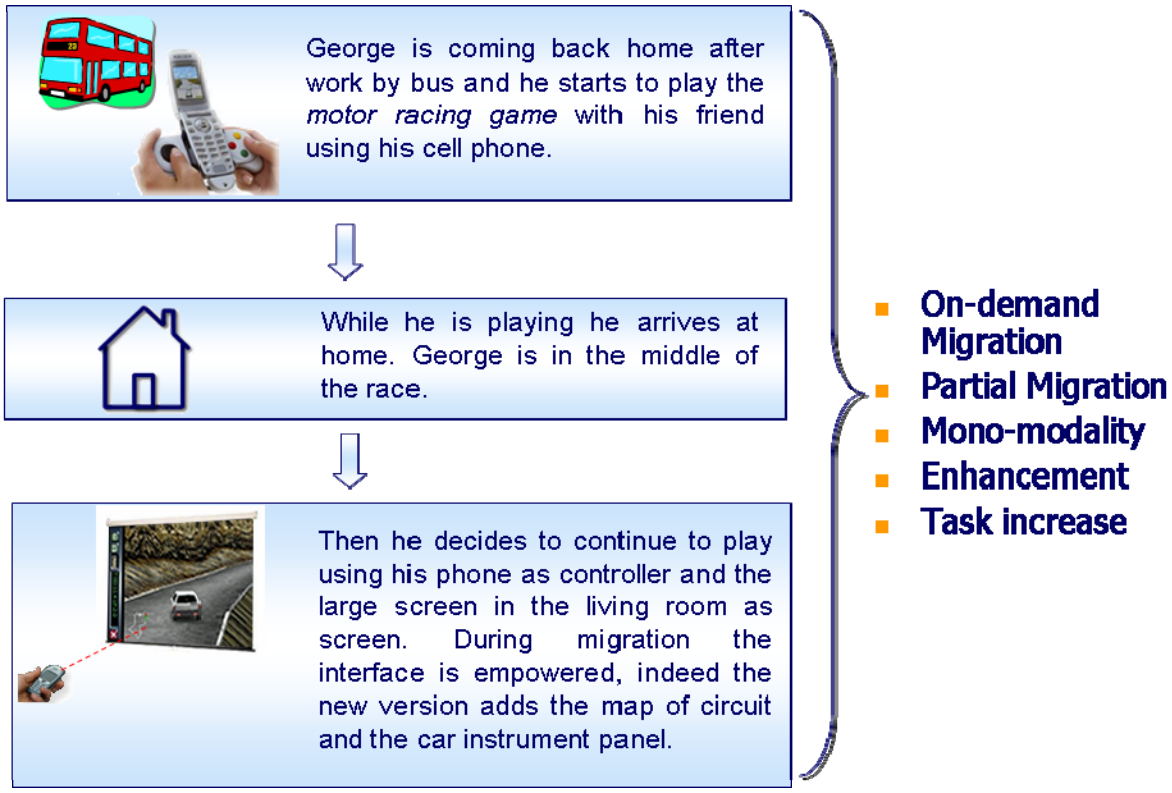


*Figure 8:* A Migration example scenario.

There are many applications that can benefit from migratory interfaces. In general, applications that require time to be completed (such as games, business applications) or applications that have some rigid deadline and thus need to be completed wherever the user is (e.g.: online auctions). Other applications that can benefit from this flexible reconfiguration support are those that have to provide users with continuous support during the whole day through different devices (for example, in the assisted living domain).

ICrafter (Ponnekanti et al., 2001) is a solution to generate adaptive interfaces for accessing services in interactive spaces. It generates interfaces that adapt to different devices starting with XML-based descriptions of the service that must be supported. However, ICrafter is limited to creating support for controlling interactive workspaces by generating UI for services obtained by dynamic composition of elementary ones and does not provide support for continuity of task performance across different devices. Aura (Garlan et al., 2002) provides support for migration but it is obtained by changing the application depending on the resources available in the device in question, while there is a need for solutions able to generate interfaces of the same application that adapt to the interaction resources available. Bharat and Cardelli (Bahrat and Cardelli, 1995) addressed the migration of entire applications (which is problematic with limited-resource devices and different CPU architectures or operating systems), while we focus on the migration of the UI part of a software application. Kozuch and Satyanarayanan (Kozuch and Satyanarayanan, 2002) identified a solution for migration based on the encapsulation of all volatile execution state of a virtual machine. However, their solution mainly supports migration of applications among desktop or laptop systems by copying the application with the current state in a virtual machine and then copy the virtual machine in the target device. This solution does not address the support of different interaction platforms supporting different interaction resources and modalities, with the consequent ability to adapt to them. Chung and Dewan (Chung and Dewan, 1996) proposed a specific solution for migration of applications shared among several users. When migration is triggered the environment starts a new copy of the application process in the target system and replay the saved sequence of input events to the copy to ensure that the process will get the state where it left off.

This solution does not consider migration across platforms and consequently does not support run-time generation of a new version of the UI for a different platform. A discussion of some high-level requirements for software architectures in multi-device environments is proposed in (Balme et al., 2004) without presenting a detailed software architecture and implementation indicating a concrete solutions at these issues. In this section, we also introduce a specific architectural solution, based on a migration/proxy server, able to support migration of user interfaces associated with applications hosted by different content servers. At the HHIS Laboratory of ISTI-CNR, it was (Bandelloni et al., 2004) found a solution based on the use of pre-computed interfaces for different platforms that are dynamically activated, which then evolved in a new solution (Bandelloni et al., 2007), implemented in an engineered prototype, supporting also migration through different modalities (such as voice).

Especially in heterogeneous environments (namely, environments in which different types of devices exist), the concept of migratory user interfaces raises a number of design issues that should be appropriately analysed and addressed in the attempt to identifying an effective migration architecture/process. A suitable framework for migration should consider at least the dimensions described hereafter. Such dimensions are:

- *Activation Type*: how the migration is triggered. The simplest case is on demand, in which the user actively selects when and how to migrate. Otherwise, in automatic migration, it is the system that activates the device change (depending on e.g. mobile device battery consumption level, device proximity, etc.).
- *Type of Migration*: This dimension analyses the 'extent' of migration, as there are cases in which only a portion of the interactive application should be migrated. A number of migration types can be identified:
    - Total migration allows basically the user to change the device used to interact with the application.
    - Partial migration is the ability to migrate a portion of the UI (the remaining portion remains in the source device).
    - In the distributing migration the user interface is totally distributed over two or more devices after migration.
    - The aggregating migration performs the inverse process: the interface of multiple source devices are grouped in the user interface of a single target device.
    - The multiple migration occurs when both the source and the target of the migration process are multiple devices.
- *Number/Combinations of Migration Modalities* This dimension analyses the modalities involved in the migration process. Mono-modality means that the devices involved in the migration adopt the same modality interaction. Trans-modality means that the user can migrate changing the interface modality. An example of migration from graphical interface to vocal interface is the case of users navigating the Web through a PDA or Desktop PC and afterwards migrate the application to a mobile phone supporting only vocal interaction. Lastly, with multi-modality the migratory interface contemporaneously supports two or more interaction modalities at least in one device involved in the migration. Work in this area often has mainly focused on graphical interfaces, investigating how to change the graphical representations depending on the size of the screens available.
- *Type of Interface Activated*: This dimension specifies how the user interface is generated in order to be rendered on the target device(s). With precomputed user interfaces the UI has been produced in advance for each type of device. Thus, at runtime there is no need for further adaptation to the device but only for adaptation of the state of the UI. On the contrary, if a runtime generation of user interfaces is considered, the migration engine generates the UI according to the features of the target device when migration occurs. In an intermediate approach

the migration engine adapts dynamically to the different devices using some 'templates' previously created.

- *Granularity of Adaptation*: The adaptation process can be affected at various levels: the entire application can be changed depending on the new context or the UI components (presentation, navigation, content).
- *Migration time*: The device change that characterise migration can occur in different temporal evolutions:
    - Continuous: The user interface migration occurs immediately after its triggering.
    - Postponed: in this case the request of migration can be triggered at any time but it actually occurs after some time. This can happen when the target device is not immediately available (for example when migrating from the desktop in the office to the vocal device in the car, which has to be first turned on by the user when he enters the car).

- *How the UI is Adapted*: Several strategies can be identified regarding how to adapt user interfaces after a migration process occurs:
    - Conservation: this strategy maintains the arrangement and the presentation of each object of the user interface: one possible example is the simple scaling of the user interface to different screen sizes.
    - Rearrangement: in this case all the UI objects are kept during the migration but they are rearranged according to some techniques (e.g.: using different layout strategies).
    - Increase: when the UI migrates from one device with limited resources to one offering more capabilities, the UI might be improved accordingly, by providing users with more features.
    - Reduction: this technique is the opposite of increase and it can be applied when the UI migrates from desktop to mobile device because some activities that can be performed on the desktop might result unsuitable on a mobile device.
    - Simplification: in this case all the user interface objects are kept during the migration but their representation is simplified, for example, different resolutions are used for figures or figures are substituted with textual descriptions.
    - Enhancement: this technique represents the opposite of simplification (e.g.: a textual description might be substituted with multimedia information).
- *The Impact of Migration on Tasks*: The impact of migration on tasks depends on how the user interface is adapted because reduction and increase can produce some change on the range of tasks supported by each device. Differently, conservation and rearrangement do not produce any effect on the set of tasks supported. Then some possible cases are: 1) after a partial or distributing migration some tasks can be performed on two or more devices in the same manner (task redundancy), which means, for instance, that the decomposition of the different tasks into subtasks is unchanged, as well as the temporal relationships (sequencing, concurrency, etc.) occurring among them; 2) after a partial or distributing migration a part of a task can be supported on one device and the other part/s is/are available on different devices (task complementarity). Additional cases are when the number of task supported 3) increases (task increase) or 4) decreases (task decrease) after migration. Obviously, a final case might be identified when the migration has no impact on tasks, as they remain substantially unchanged.
- *Context Model*. During adaptation of the UI the migration process can consider the context in terms of description of device, user and environment. Generally, the context dimension that is most taken into account with the aim of producing usable UI is the device and its properties, together with the surrounding environment.
- *Context Reasoning*. A context modelling process begins by identifying the context of interest. This depends on the specific task which should be associated with the context of interest. The

context of interest can be a primitive context, which can directly be captured by employing a sensor; or, it can be a higher-level context, which is a result of manipulation of several primitive contexts. If a context of interest is a higher-level context, a reasoning scheme is inadvertently required, which can be either a logic-based reasoning scheme or a probabilistic reasoning scheme. A logic-based reasoning scheme considers a primitive context as a factual data while a probabilistic reasoning scheme does not. Depending on the nature of the sensed data available and the way the data are manipulated, ignorance can be classified as follows:

Incompleteness: refers to the fact that some vital data about the real-world situation to be reasoned about is missing; the available data, however, are considered to be accurate.

Imprecision: refers to inexact data from sensors. Inexact data arises due, partly, to the physical limitations of the sensing elements employed. Different sensors have different resolution, accuracy, and sensing range. Besides, the performance of physical sensors can be influenced by external factors such as surrounding noise or temperature.

Uncertainty: refers to the absence of knowledge about the reliability of the data sources – this knowledge might be information about the parameters listed above to determine (characterise) the degree of imprecision incorporated in sensory data.

- *Implementation Environment*. The migration process can involve different types of applications. Probably due to their diffusion, the most recurrently considered applications are web-based systems (static/dynamic pages), but also other applications (Java, Microsoft .NET etc.) can be considered.
- *Architecture* With regard to the architecture of the migration support environment there are different strategies, for example: proxy-based, in which there is an intelligent unit managing all migration requests and sending all data to target devices; and peer to peer, where the devices directly communicate and negotiate the migration parameters.

## Conclusions

This chapter has discussed a number of issues related to technological platforms, convergence and adaptation of interactive contents and it has described the state of art in this area. Particular attention has been dedicated to solutions for adaptation to the device. Migratory interactive services has been described along with an indication of the relevant design dimensions.

This is an area that is acquiring an increasing importance, with a strong technological push. One important research area for the next years is dedicated to making the transformation rules driving adaptation accessible and modifiable by end users in order to allow their customization for specific preferences.

## References


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.

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

Banavar, G., Bergman, L. D., Gaeremynck, Y., Soroker, D., Sussman J. B.: Tooling and system support for authoring multi-device applications. Journal of Systems and Software 69(3): 227-242 (2004)

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

Bandelloni, R., Paternò, F., Flexible Interface Migration, Proceedings ACM IUI 2004, pp.148-157, ACM Press, Funchal, January 2004.

R.Bandelloni, F. Paternò, C. Santoro, A.Scorcia, Web User Interface Migration through Different Modalities with Dynamic Device Discovery, Proceedings Second International Workshop on Adaptation and Evolution in Web Systems Engineering (AEWSE'07), pp.58-72, ISBN: 978-88-902405-2-2, Como, September 2007.

Bharat K. A. and Cardelli. L., Migratory Applications. In proceedings of User Inteface Software and Technology (UIST '95). Pittsburgh PA USA. November 15-17, 1995. pp. 133-142.

Baudisch, P., Lee, B., Hanna L.: Fishnet, a fisheye Web browser with search term popouts: a comparative evaluation with overview and linear view. AVI 2004: 133-140.

Bouillon, L., and Vanderdonckt, J. Retargeting Web Pages to other Computing Platforms. In Proceedings of IEEE 9th Working Conference on Reverse Engineering WCRE'2002), IEEE Computer Society Press, Los Alamitos, 2002, 339-348.

Calvary, G., Coutaz, J., Thevenin, D., Bouillon, L., Florins, M., Limbourg, Q., Souchon, N., Vanderdonckt, J., Marucci, L.,, Paternò, F. and Santoro, C. 2002. The CAMELEON Reference Framework, Deliverable D1.1.

Chung G., Dewan P., A mechanism for Supporting Client Migration in a Shared Window System, Proceedings UIST'96, pp.11-20, ACM Press.

Eisenstein, J., Vanderdonckt, J., Puerta A. R.: Applying model-based techniques to the development of UIs for mobile computers. Intelligent User Interfaces 2001: 69-76.

Florins, M., Vanderdonckt J.: Graceful degradation of user interfaces as a design method for multiplatform systems. Intelligent User Interfaces 2004: 140-147.

Gajos K., Christianson D., Hoffmann R., Shaked T., Henning K., Long J. J., and Weld D. S.. Fast and robust interface generation for ubiquitous applications. In Proceedings of UBICOMP'05, pp. 37–55. Springer Verlag, September 2005. 3660.

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

Kamvar M., Baluja S.. A Large Scale Study of Wireless Search Behavior: Google Mobile Search. Proceedings CHI 2006, ACM Press.

Han, R., Perret, V., Naghshineh, M., "WebSplitter: Orchestrating Multiple Devices for Collaborative Web Browsing", ACM Conference on Computer Supported Cooperative Work (CSCW), December 2, 2000, Pages: 221 – 230.

Kozuch M., Satyanarayanan M., Internet Suspend/Resume, Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02) IEEE Press, 2002.

Lam H., Baudisch P., Summary Thumbnails: Readable Overviews for Small Screen Web Browsers, Proceedings ACM CHI'05, Portland, pp. 681-690, ACM Press, 2005.

Limbourg, Q., Vanderdonckt, J., UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence, in Matera, M., Comai, S. (Eds.), Engineering Advanced Web Applications, Rinton Press, Paramus, 2004.

MacKay, B., Watters, C., Duffy, J. Web Page Transformation When Switching Devices. MobileHCI 2004, LNCS 3160, pp 228-239, Springer Verlag.

Milic-Frayling N., Sommerer R.. Smartview: Enhanced document viewer for mobile devices. Technical Report, Microsoft Re-search, Cambridge, UK, November 2002.

Myers, B., Hudson, S., Pausch, R., 2000. *Past, Present, Future of User Interface Tools.* Transactions on Computer-Human Interaction, ACM, 7(1), March 2000, pp. 3-28.

G. Mori, F. Paternò, C. Santoro, 2004. Design and Development of Multi-Device User Interfaces through Multiple Logical Descriptions, accepted for publication on IEEE Transactions on Software Engineering, August 2004.

Mori, G., Paternò, F., Automatic semantic platform-dependent redesign, Proceedings Smart Objects and Ambient Intelligence 2005, pp.177-182, Grenoble, October 2005.

Mullet, K., Sano, D., Designing Visual Interfaces. Prentice Hall, 1995.

Nichols, J. Myers B. A., Higgins M., Hughes J., Harris T. K., Rosenfeld R., Pignol M.. "Generating remote control interfaces for complex appliances". Proceedings ACM UIST'02, pp.161-170, 2002.

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

Passani L. 2006, Welcome to the WURFL the Wireless Universal Resource File, http://wurfl.sourceforge.net/

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

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..

Ponnekanti, S. R. Lee, B. Fox, A. Hanrahan, P. and Winograd T. ICrafter: A service framework for ubiquitous computing environments. In Proceedings of UBICOMP 2001. (Atlanta, Georgia, USA., 2001). LNCS 2201, ISBN:3-540-42614-0, Springer Verlag London UK. Pp 56-75.

Puerta A., Eisenstein J., "XIML: A Common Representation for Interaction Data", Proceedings IUI2002: Sixth International Conference on Intelligent User Interfaces, ACM, Gennaio 2002.

Roto, V., Popescu, A., Koivisto, A., Vartiainen E.: Minimap: a Web page visualization method for mobile phones. CHI 2006: 35-44

Spence, R.: Information Visualization. ACM Press (2001).

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.