OPEN Project

STREP Project FP7-ICT-2007-1 N.216552

Title of Document:	Document about guidelines for multi-device user interfaces
Editor(s):	Fabio Paternò, Carmen Santoro
Affiliation(s):	CNR-ISTI
Contributor(s):	Giancarlo Cherchi, Francesca Mureddu, Henning Meyerhenke, Enrico Croce, Agnese Grasselli, Susan-Marie Thomas, Kay-Uwe Schmidt
Affiliation(s):	Arcadia Design, NEC, Vodafone, SAP
Date of Document:	May 25, 2009
OPEN Document:	D2.4
Distribution:	Project
Keyword List:	Migration, User Interface, Multi-Device Environments, Adaptation, Continuity
Version:	1.0

OPEN Partners:

CNR-ISTI (Italy)
Aalborg University (Denmark)
Arcadia Design (Italy)
NEC (United Kingdom)
SAP AG (Germany)
Vodafone Omnitel NV (Italy)
Clausthal University (Germany)

Title: Document about guidelines for multi-device user interfaces	**Id Number: WP2 – D2.4**

Abstract

The main goal of migration is to support users freely moving in such a way to allow them to continue their tasks while changing interaction device. Thus, adaptation to the device is an important aspect of migration, in particular adaptation concerning the interactive part of an application. This deliverable discusses guidelines for supporting effective user interface migration in multi-device environments. It considers state of the art proposals and design criteria that have been deemed useful for this purpose by the OPEN partners.

Table of Contents

1 Introduction

This deliverable discusses guidelines for effectively supporting various aspects associated with migratory multi-device user interfaces. In particular, we will address issues connected with how to support effective adaptation and continuity in migratory multi-device user interfaces. The guidelines have been identified by reviewing state of the art works that have been carried out in the field, and also taking into account the experiences of the OPEN partners within the Project. The aim of this deliverable is to provide a number of design rules that can be used for delivering effective user interfaces, which are able to preserve the continuity of the task and are adapted to the target device in various migration cases that have been identified. We aim at covering such aspects in both total and partial migration cases.

In the deliverable, we first review some state-of-the-art contributions that have dealt with possible usability issues that could be raised by applications that go across multiple devices, like migratory applications. To this respect, we judge useful considering *platforms* (namely: a platform is a cluster of devices sharing similar characteristics.) rather than single devices in order to abstract out some limited differences existing between the devices. In Section 3 we define the characteristics of the platforms that have been considered interesting in migration (the desktop, the mobile, the vocal one, the TV-based, and the multi-core platform). Afterwards (Section 4), we show some examples of rules for supporting multi-device user interfaces, that have been considered in various cases of *total* migration. In Section 5 we show some examples of adaptation techniques that have been considered in *partial* migration. Then, in Section 6 we show some examples of guidelines derived from the OPEN Project prototypes of the industrial partners. Lastly, we draw some conclusions.

2 Related Work

As it is highlighted in (Denis and Karsenty, 2004) an important aspect when dealing with multi-device interactive application is the so-called "inter-usability", namely the capability for the users to reuse their knowledge for a certain functionality when switching to other devices. Denis and Karsenty consider *continuity* of multiplatform systems at two levels: knowledge continuity and task continuity: i) Knowledge continuity is based on the retrieval and adaptation of knowledge constructed from the use of other devices; ii) Task continuity is based on "the memory of the last operations performed with the service, independently from the device used, and the belief that this memory is shared by the system". Task continuity requires that the user recovers the state of data and the context of the activity. Task continuity is one of the features that the OPEN Migration Platform is aimed to support.

Issues connected with switching devices are also described in (Pyla et al., 2006), where authors show that maintaining consistency in the rendering of the UI is not the only aspect to be considered when dealing with transition among different devices. Rather, *seamless task migration* is the most relevant aspect: the motivation is that the users should not be forced to be pro-active in planning their task migrations in advance. In the OPEN project, we are in line with this idea, since we basically aim to provide the automatic preservation of the state and its resumption on the new device. However, in order to support seamless task migration/execution among various devices, we should ensure that the potential discontinuity points due to the occurred change of context after migration (e.g. in terms of the interaction device, the surrounding environment, the time, the current focus/attention of the user etc.), are ameliorated as much as possible and therefore users are seamlessly supported not only in carrying out the transition (from one device to another one), but also they are able to continue their activity in the most effective way on the new device.

Another aspect connected with the migration is *adaptation*. Adaptation and the related issue of device independency have been a field which has received a great deal of attention. For example, if we consider Web applications, a group inW3C (W3C 2004) has identified a number of techniques and best practises that Web site authors should follow when creating content targeting multiple devices. In particular, several aspects have been considered (see following bullet list), which can vary in accordance to the specific context of use:

- *Style*: the visual appearance/rendering of a Web object
- *Layout*: the visual or temporal relationships of parts of a Web page
- *Content*: the media resources (e.g.: text, audio, video) existing in a Web page
- *Structure*: a typical technique for varying the structure of a Web page is *pagination* (namely: decomposing a single page into an ordered sequence of related, smaller pages; every smaller page shows only a portion of the original content)

- *Navigation*: the support offered to the user to move from the currently perceived unit to another one. Typical examples are hyperlinks, menus, etc.
- *Application interaction*: the way the user conveys information to the application through the features of the user interface, with the goal of influencing subsequent content provision (typical mechanisms are forms).

We note that in this classification the term "layout" is not the most suitable one for representing temporal constraints. In any case, the W3C document (W3C, 2004) discusses various adaptation techniques for each of the above mentioned aspects.

In the following sections, we describe the characteristics of some *platforms* that are involved in various migration cases. It is worth pointing out that with *platform* we mean a set of interaction *devices* that share similar capabilities. Examples of different platforms are: the graphical desktop, the vocal desktop, the (multimodal) graphical and vocal desktop, the Digital TV, etc.. Examples of devices belonging to the same platform are devices that, although of different brands, have the same characteristics. For instance, the Samsung Omnia SGH i900 phone and the Apple iPhone 3G are both equipped with a quite large screen, an accelerometer, and the interaction is touch screen -based: therefore, they both belong to what we will define later on in this document "mobile advanced" platform. Thus, a given platform identifies the type of interaction environment available for the user, and this clearly depends on the modalities supported by the platform itself. This clustering of devices into platforms allows us to abstract out some differences among the devices belonging to the same class. Nevertheless, the clustering mechanisms should be refined enough to allow to associate each device to only one category. If a device is equipped with capabilities that belong to more than one category, the association of the device with one specific category depends on the interaction resources that should be actually exploited. For example, there are devices that can support graphical or vocal interaction depending on which software is actually accessed.

3 Characteristics of the Platforms Considered

In this section we describe the characteristics of the platforms that we are going to consider.

3.1 Characteristics of the Desktop Platform

The *desktop* platform often represents the reference platform for most of the legacy applications. It generally includes a large screen with high (and flexible) resolution, high speed connection, unconstrained power and fast CPU processing. Usually, the interaction is carried out through keyboard/mouse interaction and it is also possible to present multimedia data such as images, videos, and audio files. Within this platform we consider the usual desktop computer and laptop computer but also tablet PC and Ultra Mobile PC (UMPC). A tablet PC is a laptop or slate-shaped mobile computer, equipped with a touchscreen or graphics tablet/screen hybrid to operate the computer with a stylus or digital pen, or a fingertip, instead of a keyboard or mouse.

The main disadvantage of such kind of platform is its limited (sometimes impossible) portability/mobility. The form factor of tablet PC and UMPC offers a more mobile way to interact with a computer.

We mention also new form factor concepts that have been announced such as a full-feature computer in a keyboard.

3.2 Characteristics of the Mobile Platform

The mobile platform is characterized by a wide range of variability, which is the mirror of the rapid evolution and continuously changing nature of this platform. In general, we can characterise this platform through a number of aspects: a quite variable display size and CPU power; text input is generally slow; often there is no pointing device; softkeys are used to activate commands but their number and purpose varies depending on the device; often the application exists only for the desktop; good cellular phones support for 3G or even WLAN diminishing download time in browsing.

Due to the high variability and fragmentation in interaction resources of devices belonging to such platform, we need more knowledge of the interaction resources and modalities available, since this can affect the possible adaptations that can be carried out. For instance, some phones have only a keypad and softkeys, which only allow the 5-way interaction (possibility to go up, down, left, right and select the current element). Other phones support pen-based interaction, while devices such as the iPhone also support multi-touch interaction and accelerometer-based interaction. As an example of approach that has been put forward in order to cope with such a device fragmentation, we can cite WURFL (http://wurfl.sourceforge.net/): it is basically a catalogue of descriptions regarding mobile devices capabilities, which can also act for framework for adaptation of mobile applications

Within this platform we can identify two basic categories of mobile devices: those "advanced", with pointing device (touch or multi-touch) and large screens, and the "traditional" ones.

3.2.1 Mobile "Advanced" Platform

With this term we mean the group of devices that have characteristics similar to e.g. the Apple iPhone (for instance: the HTC Magic, a Google Android -based Phone), which are: quite a large display (e.g. with a resolution of 480 by 320 pixel), (multi-)touch interaction available, and sensor-based interaction available. The devices belonging to this platform, although quite advanced, might have some limitations and restrictions on the resources available and technologies supported (see Figure 1 and Figure 2 for the Apple iPhone case).

Resource Constraints

Resource	Limitation
Downloaded text resource (HTML, CSS, JavaScript files)	10MB
JPEG images	128MB (all JPEG images over 2MB are subsampled—decoding the image to 16x fewer pixels)
PNG, GIF, and TIFF images	8MB (in other words, `width*height*4<8MB`)
Animated GIFs	Less than 2MB ensures that frame rate is maintained (over 2MB, only first frame is displayed)
Non-streamed media files	10MB
PDF, Word, Excel documents	30MB and up (very slow)
JavaScript stack and object allocation	10MB
JavaScript execution limit	5 seconds for each top-level entry point (`catch` is called after 5 seconds in a `try/catch` block)
Open pages in Mobile Safari	8 pages

Fig.1: Resource constraints of iPhone (from http://developer.apple.com/webapps)

http://developer.apple.com/safari/library/documentation/AppleApplications/Reference/SafariWebContent/CreatingContentforSafarioniPhone/chapter_2_section_6.html#//apple_ref/doc/uid/TP40006482-SW15

Area	Technologies not supported
Web technologies	Flash media, Java applets, SOAP, XSLT, SVG, and Plug-in installation
Mobile technologies	WML
File access	Local file system access
Text interaction	Text selection, Cut/Copy/Paste
Embedded video	In-place video (tapping an embedded element will put iPhone/iPod touch into video playback mode)
Security	Diffie-Hellman protocol, DSA keys, self-signed certificates, and custom x.509 certificates
JavaScript events	Several mouse-related events (see Chapter 5)
JavaScript commands	`showModalDialog()`, `print()`
Bookmark icons	`.ico` files
HTML	`input type="file"`, tool tips
CSS	Hover styles, `position:fixed`

Fig.2: Technologies not supported by iPhone (from http://developer.apple.com/webapps)

http://developer.apple.com/safari/library/documentation/AppleApplications/Reference/SafariWebContent/CreatingContentforSafarioniPhone/chapter_2_section_13.html#//apple_ref/doc/uid/TP40006482-SW21

The accelerometer sensor is aimed at improving interaction possibilities, since it allows for a scrolling based on inertial strength of the scrollbar. In addition, depending on the current orientation of the device (detected by the accelerometer), the layout of the currently visualised application is adapted accordingly.

Regarding the input techniques available in this platform, the user can interact in different ways, using one single finger (e.g. select action), two fingers, e.g.: zoom in/out a part of the UI through "pinch" gestures, that is: bring the thumb and index fingers closer together (*zoom out*) or move them further apart to scale images (*zoom in*)), or even three or four finger-based interaction.

3.2.2 "Classical" Mobile platform

With this term we mean traditional mobile devices whose interactions are generally performed through a number of joystick buttons, numerical keys and softkeys, which can vary from device to device. The screen is generally rather small; smartphones usually have resolutions between 128x128 and 240x240 pixels, while basic cellphones have even lower resolutions.

3.3 Characteristics of the Vocal Platform

With "vocal platform" we mean a platform with which the user interacts *only vocally*. This platform is generally equipped with some speech recognition features able to convert spoken words into computer-readable input (for example, key presses). Typical examples belonging to the vocal platform are, e.g. a phone connected with an Interactive Voice Response (IVR), which are generally used for

e.g. travel tickets reservation. Therefore, with a vocal platform the interaction is basically a spoken dialog between the system and the user. The main characteristic of the vocal platform is that the interaction is linear/sequential (it is not possible to carry out multiple interaction at the same time) and transient (that is: not persistent), although it sometimes reveals faster and more natural.

3.4 Characteristics of TV-Based Platform

3.4.1 Digital TV (DVB-T -based)

With this term we mainly refer to devices that exploit the Digital Video Broadcasting-Terrestrial (DVB-T) technology. DigitalTV (DTV) platform share some characteristics with a graphical desktop, such as unlimited power and large display, but differently from the latter platform the users have no mouse or physical keyboard for interacting with the device, but just a TV controller. In addition, DigitalTV sets has limited CPU power so they might not offer some interaction technology commonly found in Web applications (e.g.: javascript), which therefore could not be available in such platform. Applications for the DTV platform are generally represented by Xlets, which are java-based applications interpreted and executed by the MHP layer of the digital receiver (Set-Top-Box).

3.4.2 IPTV

IPTV (Internet Protocol Television) is a system where a digital television service is delivered using Internet Protocol over a network infrastructure, that is, a broadband connection. A general definition of IPTV is television content that, instead of being delivered through traditional broadcast and cable formats, is received by the viewer through the technologies used for computer networks, all of which employ Internet Protocol and related standards.

Although a possible environment for IPTV can be composed by a desktop platform with its display, in a typical IPTV scenario the user is seat on a sofa at a 2-3m distance from a television attached to a Set-top-box. Also the application usage differs, on IPTV media playing (live TV, video on demands, media galleries …) prevails over classic desktop application like office suite.

With these premises, is reasonable to expect that display hardware reflects these needs (at least the new ones); frequently the resolution varies from 1280x720 up to 1920x1080, with a 16:9 aspect ratio, acoustic diffusion is at least a stereo pairs of speakers and dolby surround is not so rare like classic desktop usage.

The Set-top-box hardware is generally cost/benefits optimized; typically is significantly less powerful than graphical desktop counterpart, has lower memory, lower processing power, … but also it has lower power consumption and hardware accelerated video (de)coders.

IPTV leverages two important aspects, the connectivity and content neutrality. Content neutrality means that IPTV is not a simple port of analog TV; in IPTV there are several applications that handle contents that could be videos (e.g. broadcast transmissions or on demand movies) or not (e.g. weather forecast data).

IPTV is a *connected* device, typical a connection to wired and wireless LAN is provided in addition to the support for wireless networks needed by the controllers (e.g. Infrared, Bluetooth, …) or by broadcast technologies (e.g. DVB-S/T).

The connectivity takes advantage of the bundled broadband connection in order to offer more opportunities to respond to user on demand requests both in terms of media (e.g. video on demand) and of applications. The connection allows the application and related data be downloaded immediately without have to wait the "broadcast rotation" typical of a DVB approach. The connectivity can also leverage some services (e.g. Internet browsing) that cannot be easily done using broadcast technology with limited uplink capacity (e.g. Digital TV).

The typical actuator on the hand of the users is a remote control. The control has keys for media playing (volume, start, stop, forward, reverse, …), channel selection (0-9 numbers, channel increase/decrease) and some keys dedicated to generic application (four arrows, ok, back, info and four color buttons). In a long term transmission technology will be only radio based, but today is not rare to have infrared (IR) remote control. An infrared remote control has the drawback that user must direct the control to the IR receiver (that with low battery the user have alto to move it to the IPTV IR receiver in order to reduce the distance). Keyboards are rare and typical with a compact form with an embedded trackball.

Concerning to interaction technology presents, a lot depends on the software installed on the equipment and on the system opening of the system. The common base is the capacity of reproducing media, including streaming of MPEG variants over RTSP transport. Web browser could be present also with support of JavaScript, but in general with limitations in terms of updates and support of plug-ins.

3.5 Characteristics of Multi-core Platform

There are several unique aspects to multi-core systems. Until recently, most desktop PCs typically contained a CPU with only one core. Yet, current CPUs are multi-core systems that integrate at least two processing cores, with four cores already being mainstream. The trend clearly indicates that systems with 16 or 32 cores will be common in the near future. Multi-core technology is not restricted to PCs or servers alone – it is used in a broad range of areas like embedded and mobile devices (high-end cell phones), multimedia appliances, or game consoles. This is shown in the following listing of a number of different popular consumer devices based on multi-core hardware:

Consumer devices:

- PCs with Intel "Core" and AMD Phenom CPU series (2 & 4 cores, SMP)
- Google/T-Mobile G1 cellphone (2 cores, Qualcomm MSM7201A ARM11, 528MHz)
  - Display: 3.2 in, 480x320, 65K colors
  - "The display switches from portrait to landscape mode when the keyboard is opened. Users can interact to bring up or move content

with a finger touch, tapping or touch-drag motion. The touchscreen hardware is capable of multitouch gestures, but Android does not currently support it."
(http://en.wikipedia.org/wiki/Android_G1)

- XBox360 game console (IBM PowerPC, 3 cores, shared memory architecture)
(http://en.wikipedia.org/wiki/Xenon_(processor))
- Playstation3 game console (STI Cell CPU: 1x PowerPC + 7x SPU cores with local memory)
(http://www.ibm.com/developerworks/power/cell/)

Embedded systems:

- NEC NaviEngine (4 cores, ARM11 MPCore)
  - "NaviEngine 1, System LSI for SMP-Based Car Navigation Systems" in *NEC Technical Journal*, Vol. 2 (2007) (http://www.nec.co.jp/techrep/en/journal/g07/n04/070409.html)
- ARM Cortex-A9 (4 cores, ~1GHz, ~250mW per core)
(http://www.arm.com/products/CPUs/ARMCortex-A9_MPCore.html)
- Fujitsu FR1000 (2 cores)
(www.fujitsu.com/downloads/MAG/vol42-2/paper03.pdf)
- Renesas SH multi-core (2-8 cores)
(http://www.renesas.com/fmwk.jsp?cnt=special01.htm&fp=/edge/Vol.17/child_folder/SpecialFeature/&title=Special%20Feature%2001)
- SPI Storm-1 (16 VLIW cores)
(http://www.streamprocessors.com/streamprocessors/Home/Products/Storm-1Series.html)

These systems differ not only in the number of CPU cores but also in their hardware architecture: Besides different CPU instruction sets, there are different solutions regarding memory access (cache vs. local memory) for example.

Another issue that is very important for embedded and mobile devices is energy consumption and battery lifetime: While PCs are able to provide higher clock frequencies (~2 GHz) and hence higher performance at the expense of higher power consumption (~300W), smaller devices are typically based on simpler, energy-efficient CPU designs (~2W) that operate at much lower frequencies (~0.1Ghz). To compensate the lower per-core performance, recent embedded systems also deploy multi-core processors.

These aspects further complicate the already difficult and time-consuming process of parallel programming for multi-core systems.

All these different factors pose requirements on the adaptation process and need to be taken into consideration when developing a solution. This complexity stemming from the wide variety of devices and additionally the difficulty of multi-core programming in general necessitate a unified solution. The final

outcome should be easy-to-use both for the developer and the end-user of the OPEN platform.

4 Guidelines for Total Migration

Total migration basically allows the user to change the device used to interact with the application. In this case, the application user interface is *entirely* migrated from one device to another one.

Among aspects to be considered for guidelines we can mention not only the UI adaptation but also possible rules for effectively supporting/rendering the *transition phase,* that is the phase starting with the user's migration trigger and finishing with the upload of the migrated interface in the target device. In fact, the users should be made aware of the incoming phase of device switching (transition phase), since they should be aware of the fact that during this phase the interaction with the application will not be available in any device, until such a phase is not completely finished. As a consequence, during the transition phase adequate and relevant *feedback* should be provided to the users, in order to ensure them that, although the application is temporarily unavailable, the migration platform is correctly working in the way they expect, by carrying out the necessary operations for migrating the concerned applications from one device to another one.

Also, another aspect to be taken into account is how the *continuity* is supported, namely how to effectively highlight and enable the users to continue the interaction from the point where they left off. Continuity is supported if not only the state produced as a consequence of interactions in the source device is preserved and then activated onto the new device, but also if the user is ideally capable of continuing from the "last" point within this state. Therefore, in order to provide the users with the feeling of continuing the interaction on the new device in a seamless way, the system should exploit any feature able to detect the part of the application which the user was focused on before the activation of migration. To this regard, one useful rule will be, e.g. in web applications to highlight, within the migrated application on the new device, the control which lastly had the *focus* in the source device (see e.g. `document.activeElement` property to obtain the element which has currently the focus in a page).

Regarding adaptation, and how to effectively support it in total migration, in (Berti et al., 2005) a list of possible different adaptation strategies that can occur after different types of migration are discussed.

- **Conservation**. This rule keeps the arrangement and the presentation of each object of the user interface. The typical example is scaling the user interface to different screen sizes.
- **Rearrangement**. In this case all the user interface objects are kept during the migration, but they are rearranged in a new manner according to some techniques (e.g., using different layout strategies).
- **Increase**. This strategy is typically applied when the user interface migrates from one device with limited resources to one offering more capabilities. In this case, the user interface is enhanced accordingly, by providing users with more features.
- **Reduction**. It is the opposite of increase. It can be applied when some activities that can be performed on a specific platform might result

unsuitable for another platform (e.g. see the desktop-to-mobile migration case).

- **Simplification**. In this case all the user interface objects are maintained during migration but their representation is *simplified*. For example, it is the case when either different resolutions are used for figures, or figures are replaced by corresponding textual descriptions.
- **Magnification**. It is the opposite of simplification. With this technique, e.g., a textual description might be substituted with multimedia information.

It is worth pointing out that such adaptation strategies can be applied even in combination: for instance, when passing from a desktop platform to a mobile one, there might be a reduction of tasks supported in the target platform and in addition, some UI elements can be simplified when they are rendered on the mobile device.

A different classification of adaptation processes is proposed in (W3C 2004): select/remove; navigation generation, adaptation via substitution, and adaptation via transformation. We found that such a classification is more oriented to consider the implementation techniques adopted, while the classification provided in the previous bullet list (conservation, rearrangement, ..) is more oriented to address aspects that are perceived by the user.

In the following sections we will detail such strategies depending on the particular types of migration considered, also providing some examples in the various migration possibilities we judged relevant.

4.1 Guidelines for User Interfaces in Desktop / Mobile Platforms

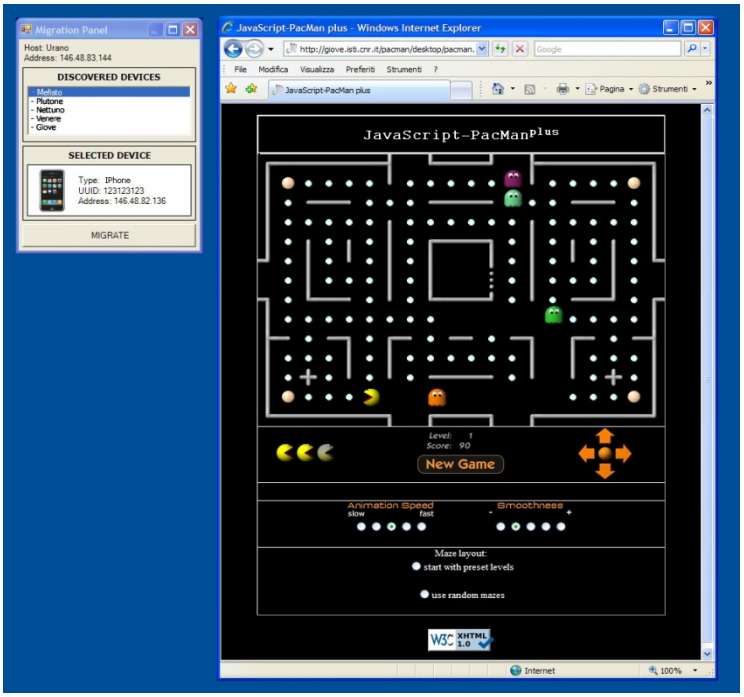
4.1.1 Adaptation Strategies

As a general rule the adaptation process should not radically change the way the activities were supported on the source device, but it should aim to maintain a consistent logical interaction model in the new device.

We identified four main cases that can be addressed when performing *desktop-to-mobile* adaptation:

1. *Splitting rules*: Such rules enable splitting large bodies of content into multiple sections which are semantically and logically related. Each of such section is rendered in a separate presentation. Therefore, with such rules one single presentation is transformed into multiple presentations. Within each new presentation, further links are added in order to enable navigation among the newly created pages. The splitted content should be presented in a way that is as much as similar as possible to the original content (therefore, any order in rendering the content should be preserved as much as possible in the adapted version). If we refer to the classification introduced in Section 4, splitting rules can be seen as an example of rearrangement. The objective of applying splitting rules might

be for instance the need of better highlighting the relationships occurring among different parts of a large presentation (which might be difficult to follow using a device with a small screen), as well as the opportunity of reducing the horizontal and vertical scrollings. An example of splitting can be found in the OPEN Web Pacman application (see Figure 3, top and bottom part), which is initially visualised on a desktop platform and then rendered on a mobile device. Since the original desktop page is too large for being entirely rendered on a single mobile device page, a splitting rule is applied. Therefore, different parts of the single original page are put onto multiple, smaller ones rendered onto the mobile device. One key point is the criterion used for splitting: generally the idea is to try to keep together the elements that are closely related each other. In Figure 3 additional adaptation rules have been applied apart from the splitting one: note that some radio-buttons in the desktop version have been translated into pull-down menus. Although we will refer to them later on in this document, details of the cost-based adaptation strategy that has been used can be found in (Paternò et al., 2008b).



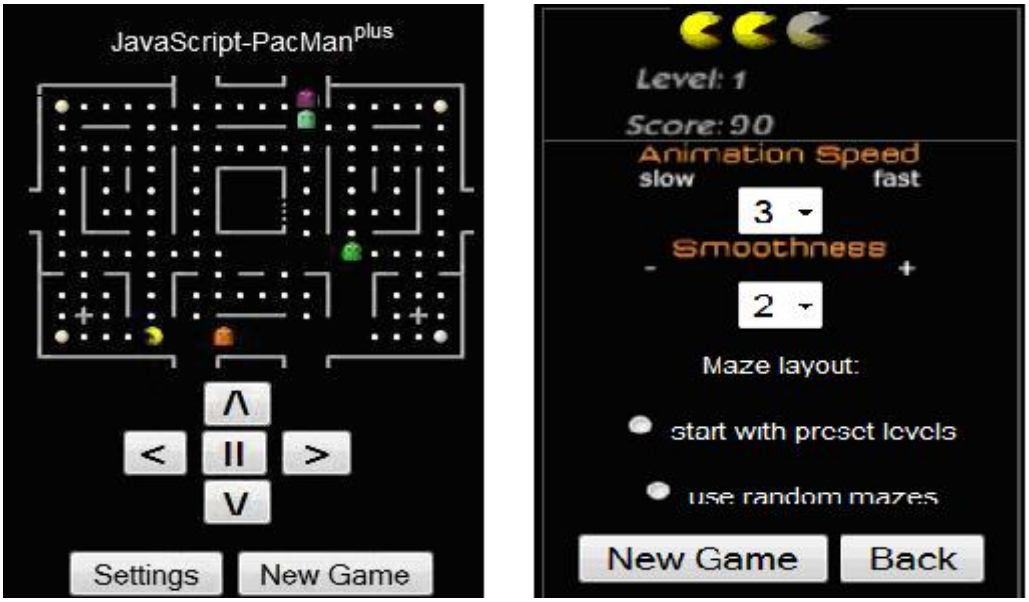


Figure 3: **An Example of Desktop-to-Mobile Adaptation.**

Top part: Desktop version; Bottom part: Mobile Version

Still within the scope of *splitting rules,* we can mention an additional technique for splitting *long text*. Indeed, when we consider long text, a specific splitting guideline can be applied in order to minimise the need for user's scrolling. In particular, if the number of characters exceeds a specific fixed threshold, the text can be split in shorter parts, which are visualised into multiple pages on the mobile device. The obtained pages all contain a similar number of characters, with the exception of the first page, which contains only the first row and a link to the remaining part. Figure 4 shows an example of this technique.

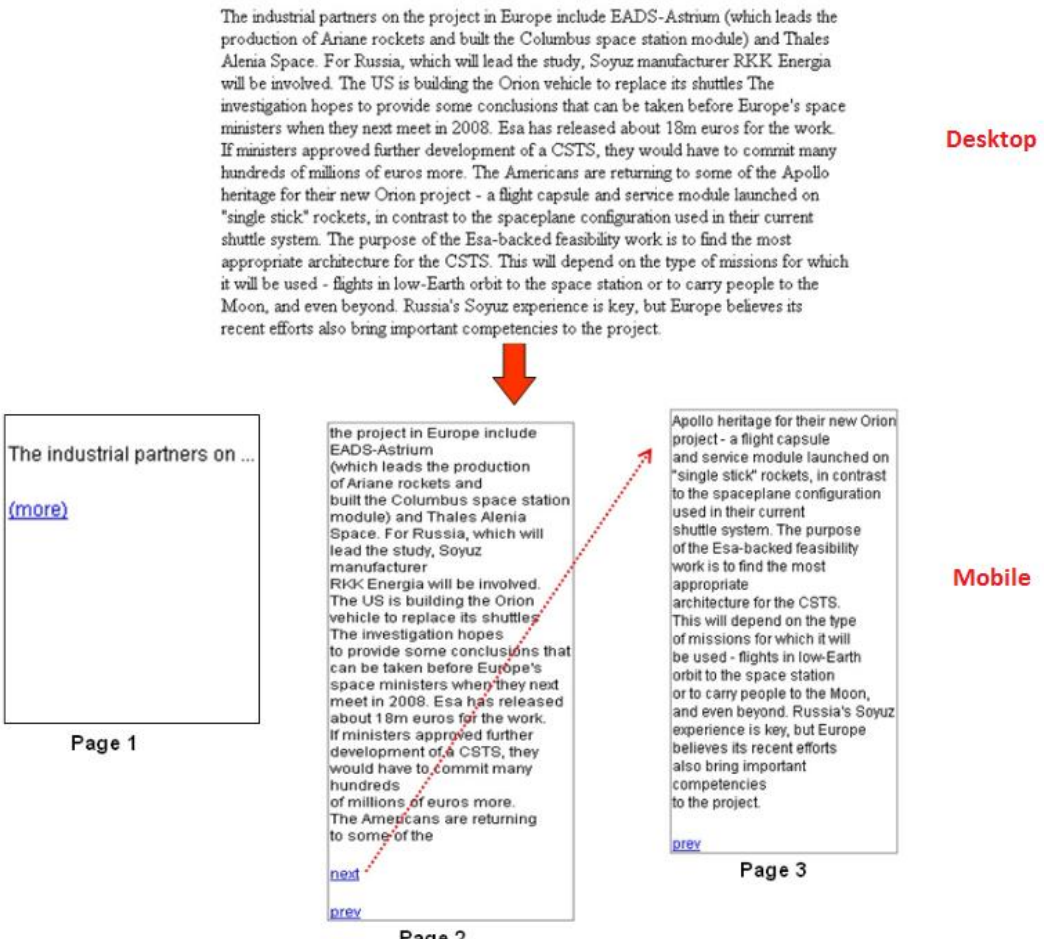


Figure 4: Splitting a long text

2. *Mapping rules*: they can be applied to both interactors (namely: elements of the UI) and *interactor composition* techniques. With such rules, one UI element (or UI composition technique) is mapped into a different one, which should support the same goal as much as possible. Such rules in some cases can be seen as an example of simplification. Two sub-cases can be identified:

   2.a: When mapping rules are *applied to elementary interactors*, they allow for transforming a specific UI object into a different one, which is judged more suitable for the new target platform but still supports the same interaction semantics. These rules should also consider the specific features available on the particular device at hand (for instance, if the device does not have video capability, then only the first frame of a video might be visualized). In addition, they should also exploit as much as possible any device-specific features (e.g.: the availability of softkeys on cellphones). An example is when a radiobutton is translated into a pull down menu in order to save some screen resources in the new target mobile device (see Figure 5). An example of this can also be seen in the Pacman example shown before (see Figure 3).

In order to provide an example of a bit more general rule, we can say that if the possible options have cardinality greater or equal than a given threshold (e.g.cardinality of options = 3) the radiobutton can be transformed into a drop-down list; if the cardinality is less than 3 the radiobutton can be maintained; if the number of elements is very high (e.g. higher than 20) the radiobutton can be transformed into a textfield. The latter rule can be adopted in order to reduce the time needed for scrolling between a long list of options. It is worth pointing out that in this case, the initial element (radiobutton, which is a selection object) is translated into another element having also a different type (textfield, which is a text editing object). In this case, in order to support at most the same original goal, we can consider the possibility to handle a input validation in the new adapted page, in order to ensure that the input provided by the user still belongs to the set of available options.

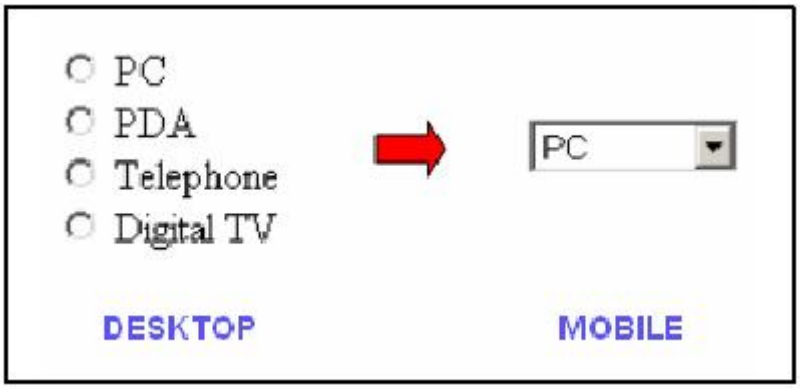


Figure 5: Transforming a radiobutton into a pulldown menu (or drop down list)

2.b When mapping rules are *applied to composition techniques*, they will be aimed at rendering the same relationship/grouping using different structural techniques. For instance, instead of using a table for visualizing multiple UI objects, a fieldset can be used for rendering such a grouping of UI objects. Another way to transform a table layout is "linearising" (that is, putting one after the other, using typically a vertical flow) the table's rows/columns, see for an example Figure 6.

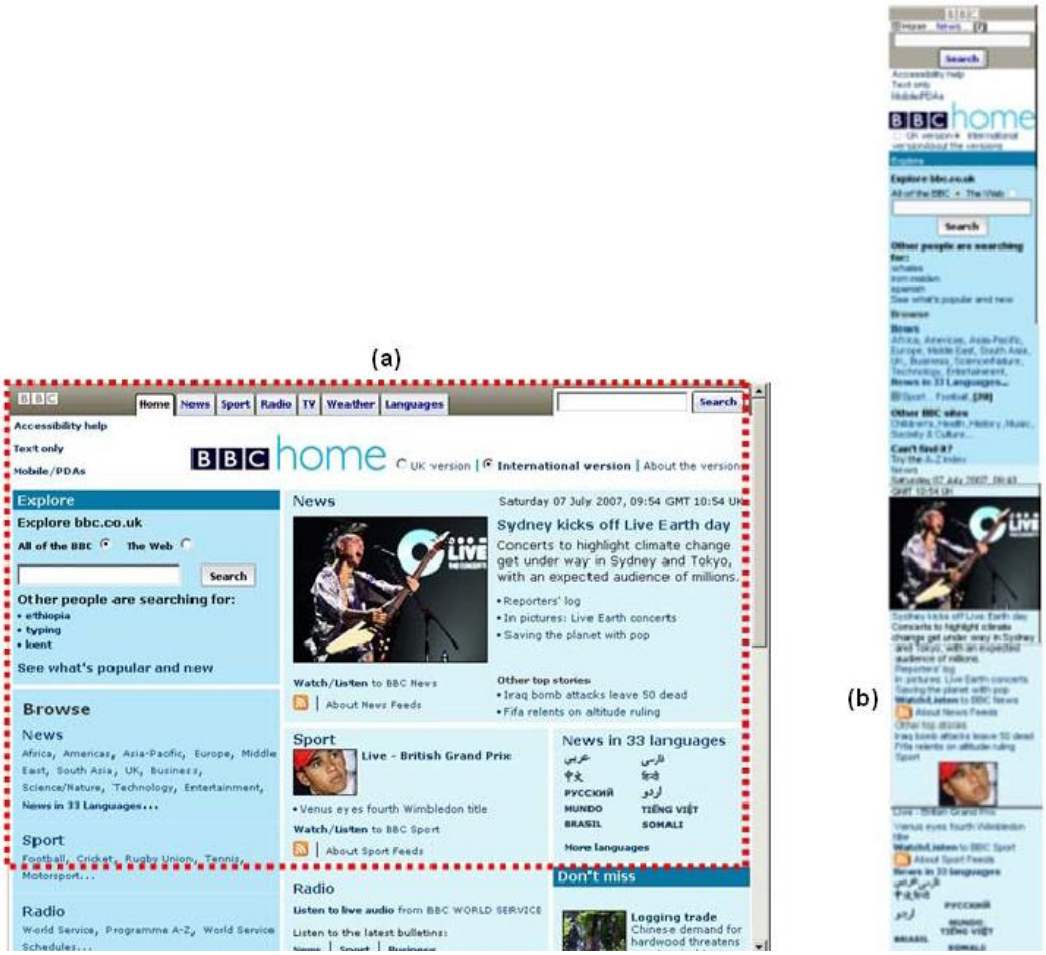


Figure 6: Transforming a composition of objects

3. *Modification rules*: Such rules are used for modifying only some attributes of a UI element. Therefore, by applying these rules, we obtain the same type of object, with some attributes modified. A typical example is when a large image is resized to a smaller one to be presented onto a device with more limited screen capabilities. Such rules are an example of simplification. Then, in this case, the UI element remains the same, while some of its attributes change. Figure 7 shows an example of resizing: the original image has been resized at a maximum horizontal dimension of 150 pixels, while maintaining the same *aspect ratio*.

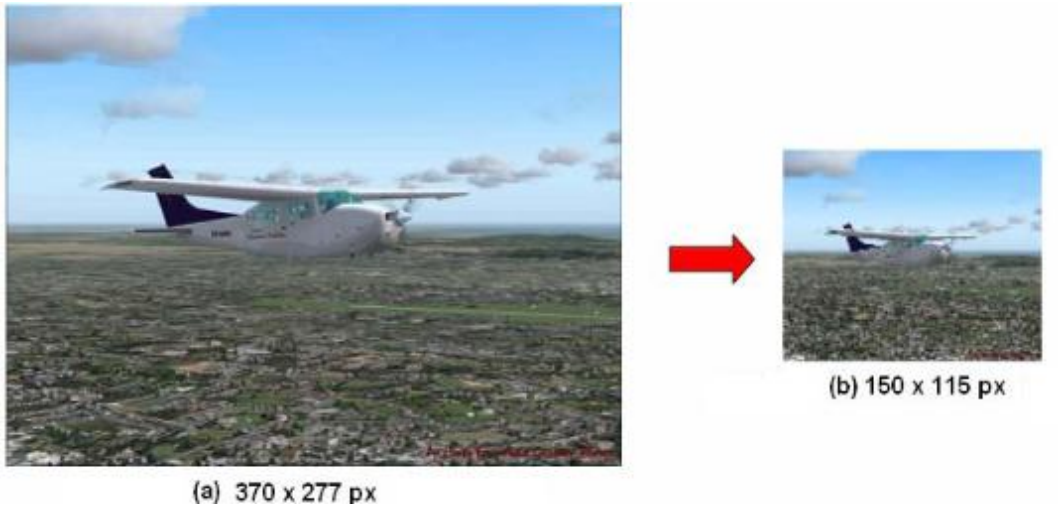


Figure 7: Resizing an image

4. *Removing rules*: such rules are used to hide some content that is judged unnecessary for being rendered on the new device at hand. Therefore, with such rules a UI object existing on the source device is no more presented on the target device. Such rules are an example of reduction. As an example of removing rule we can consider the case when some objects cannot be supported by the new platform (due to some technological limitations) and therefore the object is not rendered on the new device (e.g. Flash videos are not supported on iPhones). In other cases (e.g., when the provider of the page is also the owner of the page), even if the new device can support a specific object, the concerned UI object can be deleted since they are too "expensive" (in terms of resources used) for the new device. An example is removing an advertising video onto a mobile device in order to prevent battery consumption and/or to limit workload for the device's processor. Another case is when some UI objects, although might be supported in the new platform (from a technical point of view), support tasks that are considered irrelevant for the new platform. As a consequence, they are excluded from being presented in the new device. For instance, a task supporting a user reading a long review might be removed from a mobile device with a limited screen, since that activity is considered not appropriate for the considered platform.

A number of W3C guidelines/best practices for delivering Web content to mobile devices can also be found in (W3C 2008).

4.1.2 Continuity

As it has been already highlighted in OPEN Deliverable 6.1 (OPEN D6.1), a factor for usable migratory application is *continuity*. When the interaction moves to the target device, users should easily understand that the user interface presented refers to an activity that already started *before*. A factor that can

compromise this recognition is whether the adaptation process has changed the user interface rendered on the target device in such a way that the users do not recognise that it enables them to logically *continue* the performance of their tasks from the point where they left off in the source device. Indeed, due to screen size limitations, highlighting such information may not be easy, as it happens in desktop-to-mobile migration, where the characteristics of the device might differ a lot and therefore adaptation changes can modify quite radically the original layout. Therefore, the adaptation might make more difficult for the user to find a specific interaction object, since e.g. a change in the layout of the application might not help to recall the context in which the last task was carried out.

In Figures 8-11 there is an example of how continuity and adaptation have been supported within a shopping list scenario drawn from the OPEN Project experience. In particular, apart from highlighting the issues connected with adaptation, we will illustrate how the support for continuity has been provided, in the form of *task continuity*. In this example a user accesses a shopping application from her home desktop system, provides some personal data in order to register with the application and then decides to migrate to a mobile device so that she can edit her shopping list while freely moving about, possibly checking what is available in the fridge and kitchen. Figure 8 shows the form under editing and the migration client (part top-right) for selecting the target device and activate migration.

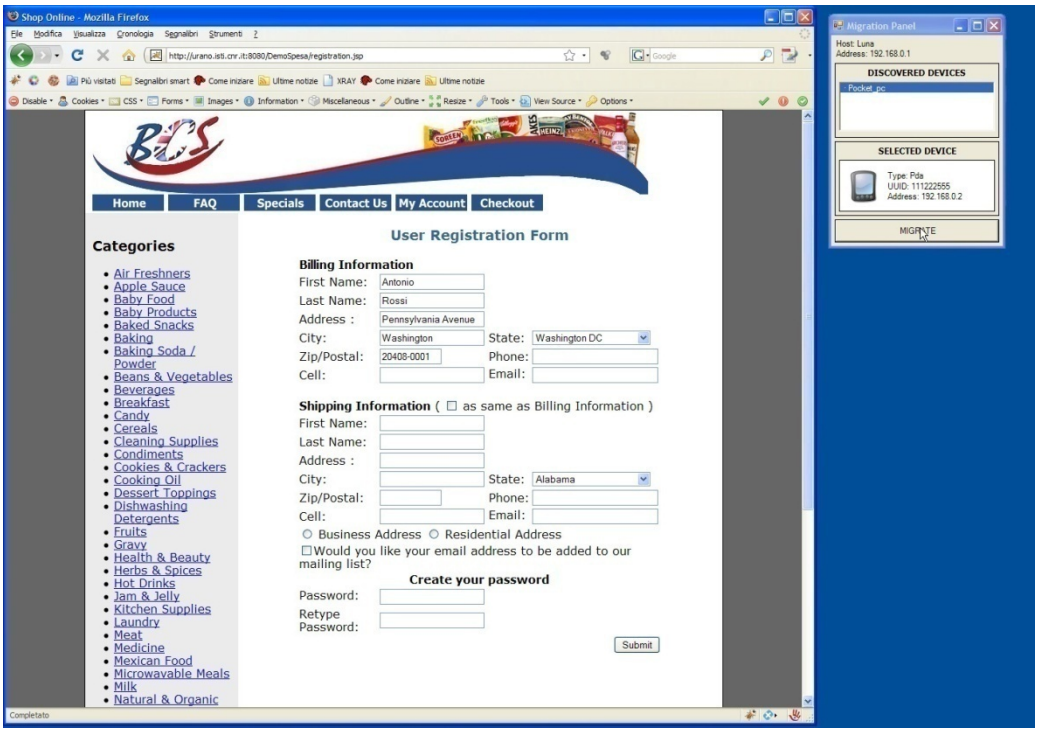


Figure 8. The desktop form at the beginning

On the mobile device she completes the registration form and logs in the application. Figure 9 shows the form on the mobile device after migration: it is adapted to the device with smaller screen and immediately shows the input that was already entered while interacting with the desktop system. Regarding

adaptation, you can note from Figure 9 that the form fields that in the desktop platform were originally rendered exploiting a two-column alignment, on the mobile platform they are rendered following a single column alignment.

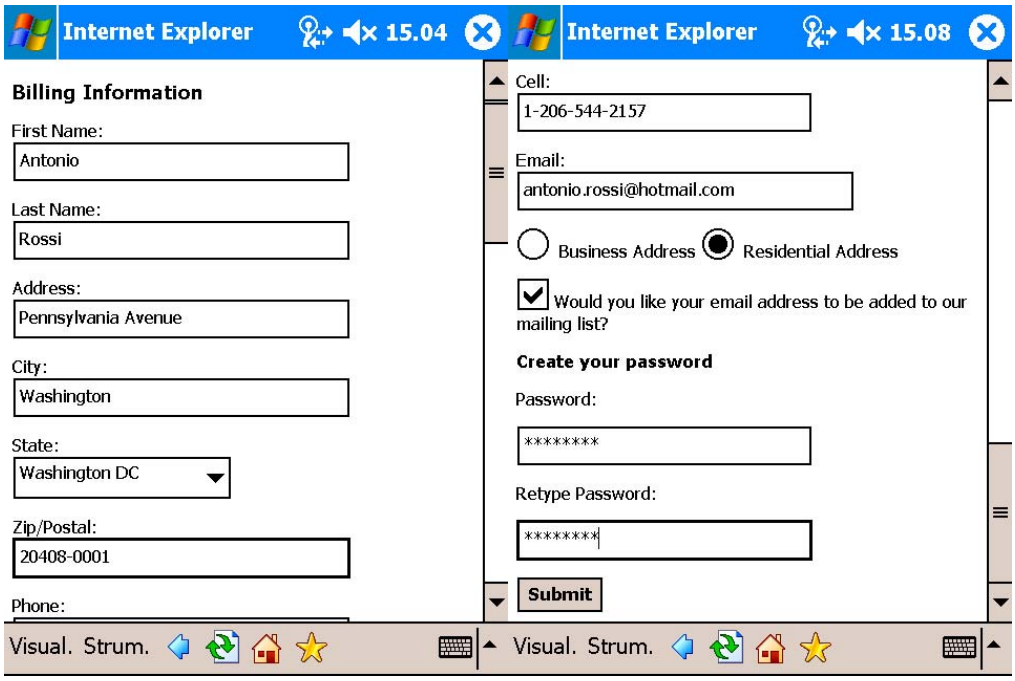


Figure 9. The mobile form after migration

Then, the user starts to enter a selection of some types of food and the associated quantities. Figure 10 shows the selected items and the associated quantities (left side) and the corresponding shopping cart (right side). Then, after having checked the fridge, the user decides to complete the shopping list comfortably by interacting with the desktop system available in the living room, and therefore she activates a second migration.

Figure 10. The mobile shopping form and cart

Thus, without having to re-login again, she migrates to the desktop system and immediately finds all the elements entered through the mobile device in the shopping cart (this is obtained by the migration platform support for session persistence through various devices) and she can complete the list and perform the payment. Figure 11 shows (right part, window at the bottom) the modal window that appears when the user interface is going to appear in the target device. This is an example of how the system can highlight to the user the fact that a transition phase is currently going on. Therefore, it shows how the incoming transition of the application to the target device is highlighted to the user. After the user accepts the migration request, then the application appears at the point it left off in the source device (in the example considered this means showing the current state of the shopping cart).

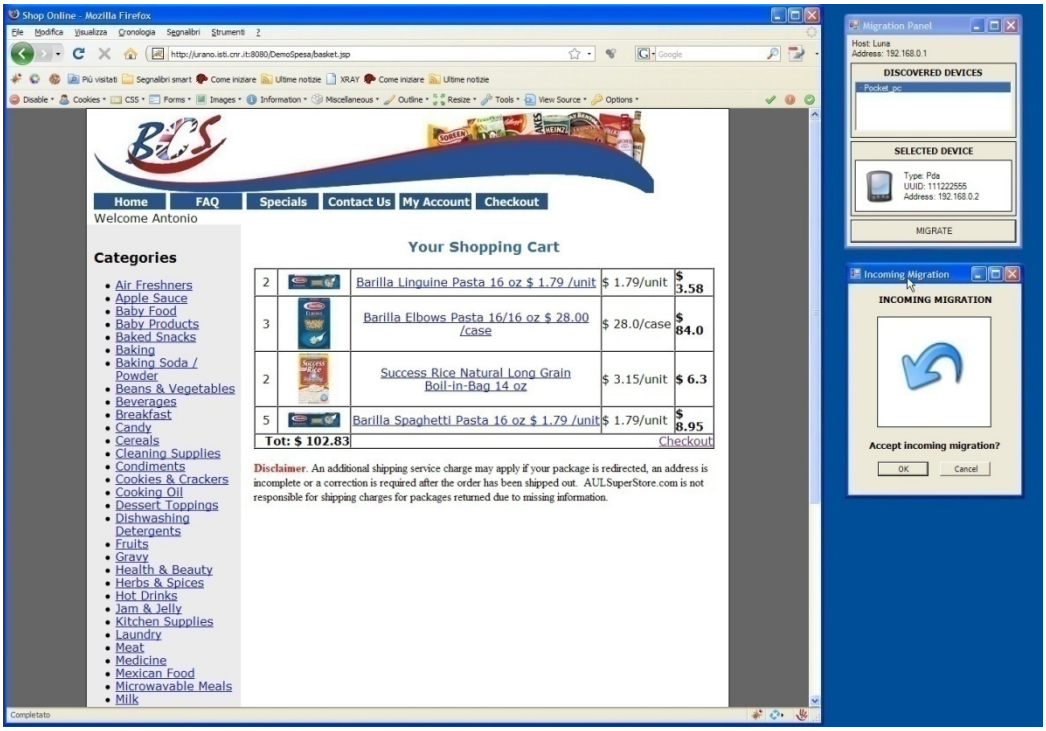


Figure 11. The desktop interface after the second migration

4.2 Guidelines for User Interfaces in Graphical / Vocal Platforms

4.2.1 Adaptation

When adapting the user interface from a mainly graphical platform (like the desktop one, or the PDA) onto vocal platforms, some aspects have to be taken into account. Indeed, in vocal UIs there are some guidelines that should be followed, which have to be taken into account when migration involves such a platform. For instance, in vocal platform, it is important that the system always provides feedback when it correctly interprets a vocal input. In addition, it is also advisable

to provide meaningful error feedback in case of poor recognition of the user's vocal input.

In addition, with vocal platforms, users should be able to interrupt at any time the system with vocal keywords (for example "menu") to access other vocal sections/presentations or to activate particular features (for instance, when the system is reading a long text). An important aspect to consider when interacting with a vocal platform is that sometimes users do not have an overall understanding of the system state. Indeed, differently from graphical user interfaces the state of the UI could not be easily available because the interaction is not persistent, and therefore the state of the system should be mentally recorded by the user, whose short term memory can be easily disturbed by some distraction. Thus, a useful technique to cope with this issue is to provide some indication about the interface state in the application after a period of silence (timeout). Other useful support for this problem can be the use of speech titles and welcome or location sentences in each vocal presentation to allow users to understand their position and the subject of the current presentation and what input the system needs at that point.

Another important difference between speech and graphical interfaces is that the vocal platform supports only sequential presentations/interactions, while the graphical one allows for concurrent interactions. Thus, in vocal interfaces we have to find the right balance between the logical information structure and the length of presentations. For example, a large desktop presentation with some meaningful parts (i.e., some top level grouping) should not be transformed into a single vocal presentation containing a long sequential and heterogeneous vocal output. On the one hand it is better to split desktop presentations by inserting elements associated with grouping composition operators in different vocal presentations so that the main vocal presentation is structured to access the various logical parts. On the other hand, newly created vocal presentations containing grouping content should not be further split in order to avoid deep navigation levels, which could be a negative factor for user orientation in speech interfaces. Moreover, the generated vocal presentations representing the elements of each desktop grouping should produce the proper amount of homogeneous vocal flow. Migration involving platforms that exploit two different modalities as in this case (vocal and graphical) can be called "trans-modal" migration (from one modality to another one). Examples of guidelines that can be followed in this case are reported in (Bandelloni et al., 2005). In particular, in this work a Restaurant application is considered. It allows users to select a restaurant, access general information and make a reservation. The user starts using a graphical PDA; when the application is activated in the vocal device, a spoken summary of the interactions that have been done before through the PDA is provided to the user (see Figure 12, top-right-part). Therefore, apart from adapting the user interface for the characteristics of the new device, in order to support an easy continuation of the task that was started in the previous device, the applied guideline was to provide an initial audio feedback summarizing the information already entered in the previous device.

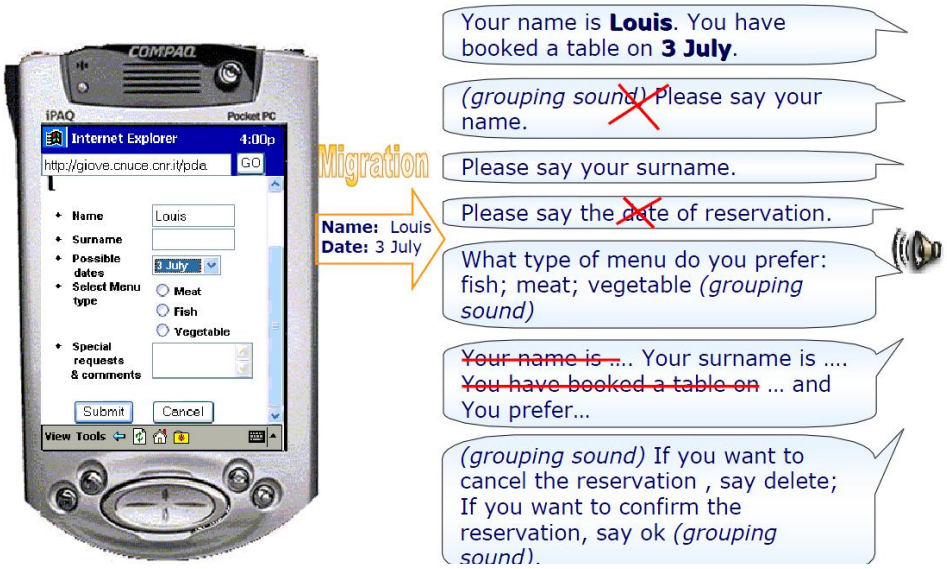


Figure 12: Support for continuation and adaptation in transmodal (PDA to vocal) migration

Another aspect that should be addressed when considering a trans-modal migration is the fact that the user interfaces for the graphical platform (e.g.: desktop/PDA) and vocal platforms might differ in the *tasks supported* (apart from the type of concrete objects used, which are platform-specific). Indeed, there can be some tasks that are not judged relevant for one platform and then they are removed in the new platform. For example, in the Restaurant reservation example previously mentioned, the user can provide special requests/comments through the PDA, while this feature is not offered on the vocal interface as it was judged not relevant for the considered platform (see Figure 13). It is worth pointing out that the latter technique should not be applied if we consider e.g. a multimodal user interface like a graphical+voice desktop (the "special requests" field would be maintained), since the capabilities of the new platform allows usable support for that (text editing) task.

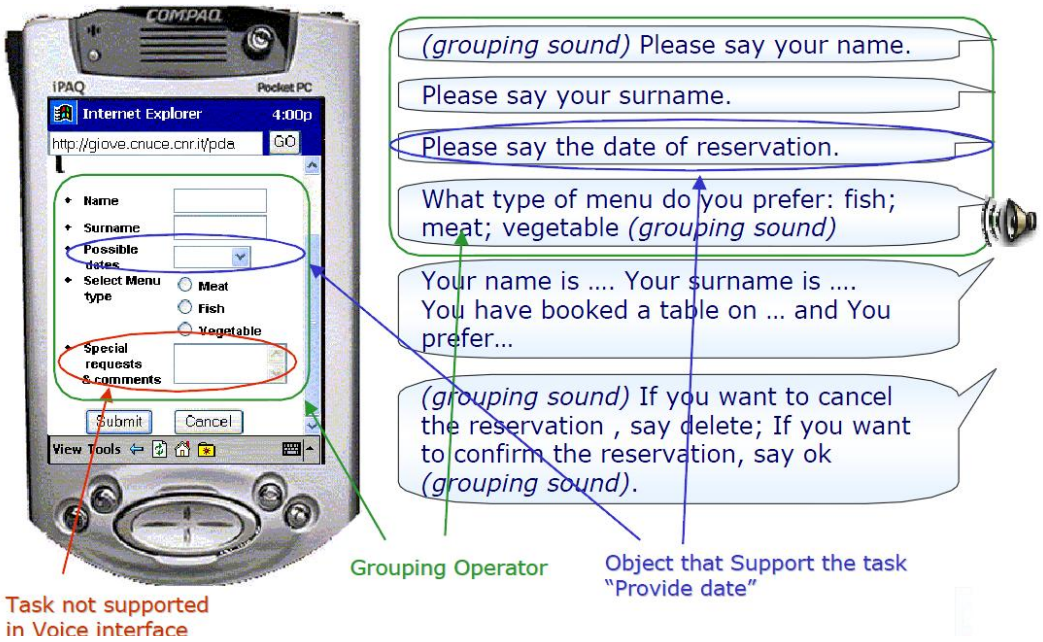


Figure 13: Graphical-to-vocal Adaptation

4.2.2 Continuity

When the migration is from desktop to vocal, in order to effectively support continuity, a vocal feedback should be provided via an initial message, recalling the information inserted before migration, through the source platform. In addition, a final message at the end of the session about the information inserted after migration might be also provided, in order to reduce user's memory load. A further, potential source of problems when migration involves graphical and vocal platforms is the fact that sometimes the tasks that are supported by such two platforms may differ (e.g. in the number and in the way they are supported). Therefore, a task that is supported in the source platform might not be provided in the target platform.

4.3 Guidelines for User Interfaces in Desktop / Digital TV Platforms

Regarding this kind of migration, the two platforms involved are quite similar in the screen size capability. Differences in such two platforms are associated with the different environments they could support. For instance Xlets (namely, the applications that can be loaded on DTV) are java-based applications, therefore, there are some features that they might not be directly supported (e.g. javascript, the existence of a Web browser,..). A particular characteristic of the DigitalTV is the fact that the control of the application is performed through a dedicated remote handset, while with desktop platforms the interaction is generally carried out through a mouse and a keyboard. This implies that, for instance, tasks like entering/editing a text, and also navigating within the same presentation (which is generally performed on a Digital TV through a remote control by giving focus, in

rotation to the different elements of the presentation) is much more easy/direct with desktop platforms.

In addition, since the user is generally quite distant from the DTV set, a reasonably big font size (between 24–36pt) and font style are generally used in order to guarantee a good readability of the text.

Regarding *continuity* aspects one aspect that should be taken into account is the fact that the some user might do some assumptions regarding the features that will be available on the DigitalTV with respect to the desktop platform. For instance, when use interacts with a Web application, she may save in the history a number of sites already accessed, which, after migration, she may expect to find on the other platform (DigitalTV). However, the "history" of e.g. Web pages already visited on the desktop platform might not be always available after migrating on the DTV platform (which basically support java-based applications). In this case the user might feel disappointed in discovering this lack after migration. Therefore, a guideline could be available that informs the user about the features that will be provided (or not) on the various devices.

4.4 Guidelines for User Interfaces in TV-based / Mobile Platforms

4.4.1 Guidelines for User Interfaces in Digital TV / Mobile Platforms

Digital TV and mobile Platforms are two radically different platforms, generally associated with two very different contexts. Indeed, DigitalTV is found at home, while most of the use of mobile device is generally *outside* home. With respect to the dimension of the screen the adaptation guidelines for this case can be quite similar to the case involving desktop and mobile platforms. Another aspect to be taken into account is the fact that such two platforms might suffer of some limitations (e.g. no javascript support). Examples of rules/guidelines that can be followed in this case have been already discussed in the work of (Paternò et al., 2008a). For instance, as Figure 14 shows, when moving/migrating from a PDA (source device) to a Digital TV (target device) for a shopping application, some guidelines have been applied, leading to an adapted version on the target device. As you can see from Figure 14, on the PDA, due to the smaller capability in screen size, the selection about the different types of products is supported by a combo-box (which visualises only a choice at a time). Instead, the larger screen size of the Digital TV allows for supporting the same task through a radio-button (for which more screen space is needed because all the choices are shown at the same time).

Another difference between the user interface of the source device (PDA) and the user interface of the target device (digital TV) is the fact that the target platform supports an additional task, that is the possibility for getting further details on the different products (see "Details" button in Figure 14, bottom part). As you can note, this task is not supported on the PDA device.

In addition another rule that has been followed when adapting from PDA was to modify not only the type of elementary objects of the user interface but also in structuring its layout. Indeed, as you can see comparing top part and bottom part of Figure 14, the larger screen space available on the Digital TV allows for arranging buttons in a single-row, which is not the case for the PDA device.

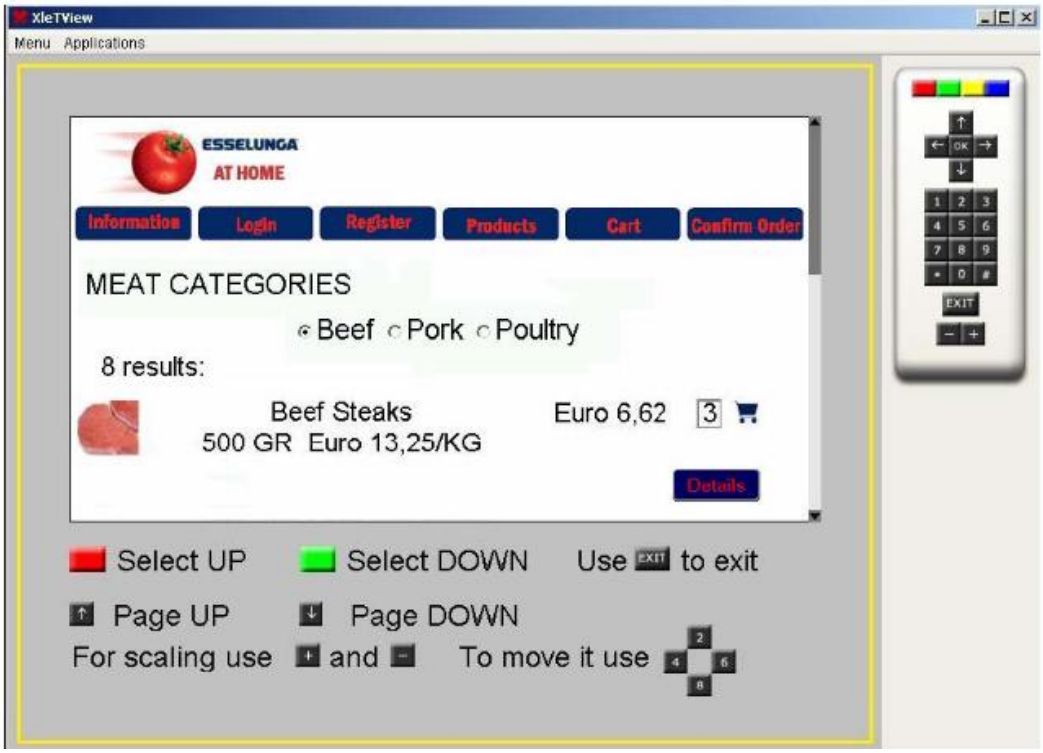


Figure 14: PDA to DTV Migration/Adaptation.

Top part: PDA version; Bottom part: DTV version after migration

4.4.2 Guidelines for User Interfaces in Migration between IPTV / Mobile Platforms

4.4.2.1 Adaptation Strategies

Moving from IPTV to mobile platforms (and vice versa) means to adapt the user interface to radically different devices and contexts. A lot of aspects must be taken in consideration in the adaptation in order to have a smooth migration between the two environments. In the next paragraphs there are some suggestions to migrate from IPTV to mobile, for the reverse migration the hints could be applied in reverse (e.g. reduce font size moving to mobile become increase font size going back to IPTV).

First list of hints are related to the context adaptation:

- Aspect adaptation, IPTV is typical connected to a horizontal display while a mobile platform has a vertical one (that in same case could be used in horizontal). The hint is to change controls pagination from horizontal to vertical (e.g. placing labels describing the control on the top instead on the left of the control) and avoid horizontal usage that can be uncomfortable if the user is required to use hard controls (e.g. arrows and soft keys can become unnatural).
- Screen density, mobile devices typical have a lower colour range and a higher pixel density, these aspects should be taken in consideration in quality optimization, e.g. anti aliasing quality can be decreased

(compensate by smaller pixels), but dither should be increased in order to compensate lower colours especially in situation where the artefacts will be more visible (e.g. fading effects).

- Aspect ratio adaptation, IPTV display has a 16:9 ratio while mobile platform has typical a 3:4 ratio. The hint is to have different sets of images in order to avoid distortions (or at least a cropping/resizing policy that take in consideration the aspect ratio).
- Brightness, generally a user uses the mobile platform in very different light situations and despite IPTV scenario controls to regulate contrast/brightness are not of easily use. The hint is to provide more contrast/sharpness on the interface artefacts also reducing space between controls (that instead could be very useful on big screen scenarios).
- Font size, mobile user is typical a stand up people with the equipment display at 50-70cm distance, these means that text font size can be drastically reduced from 26-36pt to 10-12pt without sacrifice a good readability of the text.

Second list of hints are related to modification that can be applied to gain a more mobile oriented user interface

- Iconography, mobile interface, to gain usable space, can replace common space consuming control with a well accepted iconography, e.g. replacing a "help" button with a more compact "?" icon.
- Compact information, on mobile to keep the interface optimized secondary information can be removed, moved in different pages or showed only on needs, e.g. if on IPTV platform each control has a brief text description of the control (e.g. "According to the [long terms and conditions] insert here the money you want to bet") the description could be shrink (e.g. "bet (+) (?)"), the description show only if the user selects the *more* options and the terms and conditions moved to a secondary paged reachable by a iconography link.
- Shrinking, some rules can be used for modifying some attributes of the user interface elements in order to reduce the space consumed. A typical example is to reduce a large image to a smaller one that fits on the device screen. Another example could be to cut long textual string in order to be shown on a platform with more limited resources.

Next section is more related to try to adapt the user interaction with the interface:

- Thumb(s), fortunately there is not a big difference between a remote control and a mobile keypad, the hint is to keep a one thumb fallback also when more powerful controls are available (e.g. gesture recognition, touch screen, …) in order to allow the user to use the application also with one hand.
- Access keys, the association of action to access keys should be revisited in the mobile migration due to the typical lack of four colour button on the mobile, some sort of emulation should be present (e.g. soft key mapping)

trying to keep the same experience (e.g. using the same colour for the label background).

- Control order, like IPTV scenario also in mobile usage the switch between controls is not easy and some tricks should be used to avoid the lack of a pointing device. E.g. in mobile the features that move the focus to the next control should be present when the user finishes to use the control (e.g. has picked a value from a combo box); due screen restriction an higher diligence is due to keep all the logical grouped component on the same page, a useful way could be to move all the optional fields into a separate page.
- Control changes, like IPTV platform the lack of a classic pointing device make some controls complex to use (especially combined to screen restriction) so some transformation rule can be applied e.g. changing radio buttons into compact combo box, change from a "tab" approach to a "menu" structure, … Also transformation on how the data will be presented should be evaluated in order to increase the readability, e.g. maybe an outline/tree view control could be more effectiveness than a table to present crucial information, however user can get further information opening a tree node or selecting an outline link.
- Splitting rules, such rules enable splitting large bodies of content into multiple sections which are semantically and logically related. Each of such section is rendered in a separate presentation. Therefore, with such rules one single presentation is transformed into multiple presentations and in each of the new presentation further navigation elements are added. The objective of applying such rules might be for instance the need of reducing the horizontal and vertical scrolling, as well as better highlight the relationships occurring among different parts of a large presentation and which might be difficult to follow using a small screen device
- Due to the single user nature of the mobile platform some acoustic and vibration feedbacks could be taken in consideration in order to migrate/enrich the experience, e.g. a button that visually change his state or a status indicator (e.g. the presence of a device that support the migration) could be translated to acoustic messages. The drawback is that user could not desire such approach (e.g. (s)he is without the headset in a public space) and should be used only in accordance of his/her preferences.

A consideration about "hidden" events, on IPTV some events of interest could be collected by the users also if they are not explicitly presented by the platform, e.g. a user can see end of laps of the F1 show also when is chatting (and live TV is in background). After the migration these kind of events could be lost (e.g. as the background application, due to screen size limitation, will be not visible), so an analysis should be done in order to understand the real value for the user and to provide a fallback feedback (e.g. according to user profile, vibrating the device at each lap end, activate the live TV application when the last lap is starting, …).

Last considerations are related to the *migration* controls. The first one is probably the asymmetrical migration; if it is probable that a user wants to migrate to an IPTV platform from his mobile as soon as the large display is available the only presence of a mobile platform is, instead, generally not enough to infer the intention to migrate the application.

This asymmetry should be taken in consideration during the design of the user interface, but also the context should be used to change the experience, for example, the detection of a "home" IPTV should have different policy compared to a guest one.

Migration opportunity can be represented adding avatars iconography on IPTV when one or more mobile platforms are detected (and collapsing migration controls on the avatar context menu); a bit of more logic can be applied to detect some events for which is possible to deduct an intention to migrate, e.g. blinking the associated avatar when a Bluetooth connection is switched on.

Instead on the mobile, the detection of the "home" IPTV could be an event so important to justify an interruption of the normal user experience adding an explicit control that can trigger the migration (e.g. an on screen warning and a temporary migration action mapped into a soft key), leaving to a less invasive advice (e.g. an handset vibration) the detection of a possible IPTV platform migration candidate.

For both migrations the "back" task could be an issue as both platforms concentrate their actuators on a single hand (remote control for IPTV, the device itself in case of the mobile platform); a way to mitigate the effect could be to present the back action also on the other device, e.g. after a IPTV to mobile migration is completed for a short time the IPTV/remote control propose a easy way to go back.

As any machine smart adaptation can be user boring, all the automatism should be user controllable through user preference settings (and due to asymmetry in both directions) including the timing to avoid continuous switching warnings (e.g. changed the room, but still in a proximity range of a IPTV).

In conclusion a simple and limited example of the application of the below rules is reported. In this example the IPTV application is something that looks like as in Figure 15 below:

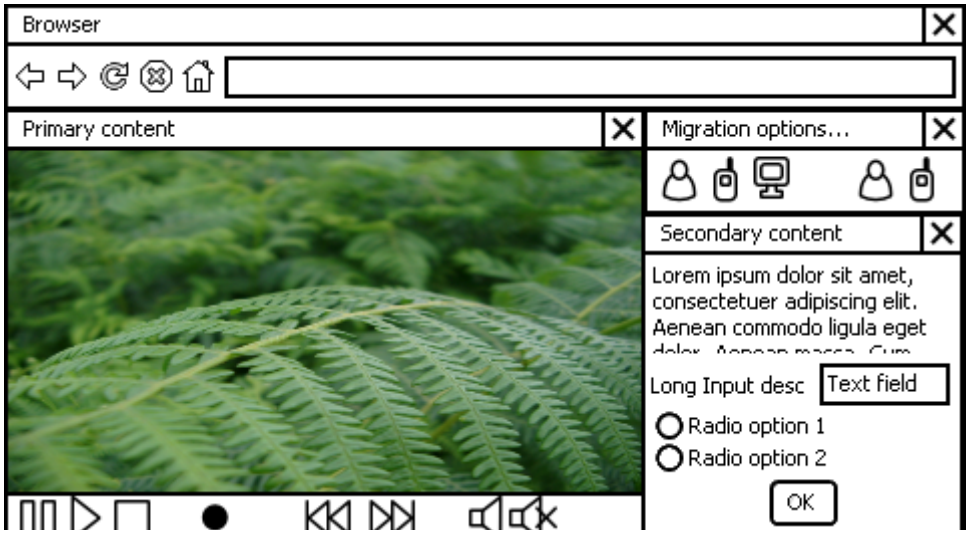


Figure 15: The application visualised on the IPTV

Where there are two contents: the primary has the bigger space in the top-left corner, while the second is placed in the less important lower right corner.

The application when showed in the mobile context, thanks to the transformation rules, could look like:

Figure 16: The application shown in a mobile context

The first difference is that the screen is not big enough to collect all the application outputs; in this case a vertical (re)layout with scrolling solution is adopted. The primary content has been placed on the top and the secondary is kept smaller then the screen size in order to maintain at least a part of the primary content also when the secondary has the focus (see Figure 17, darker is the hidden part).

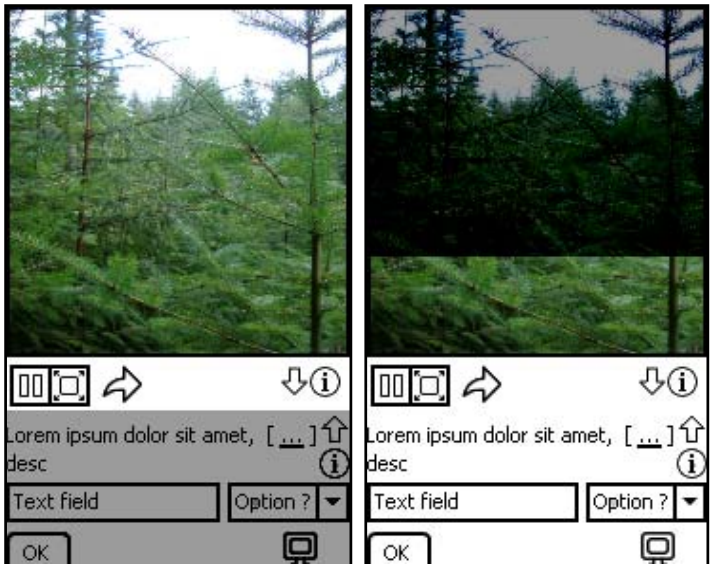


Figure 17: Managing primary and secondary content on a mobile platform

Note that a lot of IPTV secondary controls are removed, e.g. the browsing controls are removed as the user is assumed enough skilled to remember phone keys mapping, *window* controls have been stripped as not so useful in mobile context, spacing are reduced and important icons have a visibility frame, … Media controls instead are collapsed in few most useful ones with the opportunity to show more (right arrows). Instead some new controls are introduced, like an icon to have a maximized horizontal view of the primary only content, more help icons and icons that allow to switch between secondary and primary part of the screen (up and down arrows).

The media was also changed from a high quality 16:9 format to one more mobile optimized. Note that also changes the media itself should be desirable, e.g. an one hour show is perfectly suitable for IPTV scenario, but when ported to mobile a slip into a summary and 5-15minutes episodes will increase user usage experience.

Also the second content was subjected to transformations, e.g. the texts were shortened (keeping a way, with a […] link, to read relevant ones), input controls are re-layouted and some of them are substituted with more compact version (e.g. the radio buttons list was replaced with a compact combo box).

Last words on the migration, the controls in mobile were striped out as an event approach is assumed (e.g. when the IPTV is discovered a pop-up will appear). Just one migration (computer in the picture) icon was present in order to grant the user a fallback way to start the migration when the primary way does not work.

4.4.2.2 Continuity

In addition to the adaptation hints given in the previous paragraphs, some further consideration could be added for the IPTV/Mobile scenario.

The first consideration is that probably the mobile platform display will be not big enough to accommodate all the applications running on the IPTV screen. In order

to grant the continuity the migration should also migrate application focus, e.g. if the last action on the IPTV platform was a chat message the mobile platform should start with the chat page, with the focus on the last control used.

In order to reduce the discontinuity introduced by the migration some tricks could be introduced to simplify the change, one of them could be to migrate also the history and not only the session context. E.g. if in a chat session context migration grants the user to still have a open session with the conversation group, a trick to reduce the discontinuity could be also to port the last messages exchanged.

4.5 Guidelines for Multicore Platforms

The adaptation process on multi-core platforms mainly has to deal with differences in hardware capabilities such as speed, screen size and energy consumption.

Regarding adaptation to device performance, the GUI toolkit library may exchange intricate animations and effects by less demanding versions or completely disable them when desired.

In order to adapt to varying screen sizes, the GUI toolkit can for example use different buttons or fonts and adjust font style or spacing. More extensive changes can also be made to the overall GUI layout by re-arranging widget items or by replacing them with simpler or more complex variations (parts of this functionality will be provided by other modules of the OPEN platform).

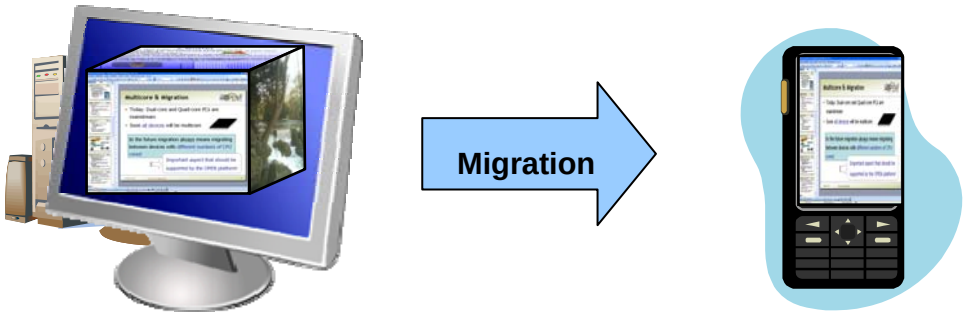


Figure 18: Sketch of the GUI adaptation process caused by migration

In Figure 18 the basic idea of adapting the GUI to the hardware capabilities is illustrated. Besides taking the screen size into account, the GUI library adapts its computational requirements according to the number and speed of the device's computing entities. On a desktop PC one might use 3D effects as shown on the left side of Figure 18. After migrating to a less powerful device such as a smartphone, expensive effects are disabled.

Furthermore, the functionality of the multi-core GUI toolkit could be extended by retrieving status and context information from other OPEN run-time modules. E. g., according to the battery status information, the power consumption could be controlled by enabling and disabling CPU cores at run-time.

In our implementation the dynamic choice of CPU cores is supported by a multi-core task scheduling library that supports dynamic load balancing (called TPI - Task Processing Interface). TPI can also adapt the number of utilized cores during the migration between devices with a different number of cores. Consequently, as TPI will detect the available cores and distribute the workload evenly among them, the GUI performance and appearance is automatically adapted to the capabilities of the hardware.

In general, the adaptation concept of the GUI toolkit and the TPI library help to reduce the complexity of parallel programming for the application developer: TPI hides many of the difficult low-level details by providing an easy to understand and unified programming interface. Furthermore, as the GUI toolkit already makes use of TPI internally, the developer does not need to implement multi-core code for most application parts dealing with graphical output.

5 Guidelines for Partial Migration

Partial migration is the ability to migrate a portion of the user interface onto a new device, while the remaining portion remains in the source device. One of the typical examples of partial migration is splitting the application between the *control part* and the part dedicated to rendering the data (*visualisation* part, in graphical platforms).
In partial migration, one of the objectives of the system is to make clear to the user *what* they should expect to find in one device and what in the other(s) device(s) involved in the migration. Then, one of the key aspects will be to avoid users believing that they will find the *same* functionalities on every device. Another key aspect is to avoid that the user looks in a device for a functionality that has been placed onto another device. In other words, in partial migration, we should try to maximize the user's *awareness* and *control* of what is happening with migration. To this regard, in partial migration, the transition phase is even very relevant in its goal of providing key information to the user about what is currently happening.

Another aspect to be considered in partial migration is *task continuity*. If we consider one of the classical cases of partial migration in which the control part remains on the source device and the data presentation is transferred on the target device, from the user's point of view this case should provide a better feeling of continuity in the execution of task (provided that the state has been preserved in the migrated application). In fact, differently from what happens in total migration, in this case the users basically *continue* to use the same interaction device for controlling the application, and therefore this should ameliorate possible discontinuity effects caused by changing the involved devices.

Another key point in partial migration is the fact that the attention of the user will be split, after migration, onto two devices. Then, suitable techniques should be identified in order to cope with this potential issue of user's division of attention, and then identify the best manner to support partial migration.
The use of logical user interface descriptions can be useful for this purpose because they provide the possibility for automatically identifying the logical structure of a user interface (e.g. to identify parts dedicated to specific purposes such as the navigation bar, the banner, the content area, …). Thus, they can provide useful suggestions for indicating which part should be migrated. The typical example is a user interface on a mobile device. The device starts the migration when it is close to a large screen. It migrates the main content area in order to take advantage of the larger screen and in such a way that it can be controlled through the mobile device.
Even with this approach based on logical UI descriptions, our previous experiences already showed that it is not always possible to split the application into exactly two parts (Bandelloni & Paternò, 2004). For instance, problems arise when the presentation resulting from a control action contains some additional control objects. In this case partial migration could lead to rather confusing split interface, since control objects might appear on both the source and the target platforms.

Figure 19: An example of Partial Migration (from PDA to a PDA+ desktop platforms)

Figure 19 (top and bottom part) shows a sample tourism application for PDA: in this example a partial migration from PDA towards a desktop PC is applied (see also (Bandelloni & Paternò, 2004)). Figure 19 (top part) illustrates the result on the PDA, where only the control part remains and the objects on the page have been rearranged in order to provide a pleasant and usable interface. The user can now select images using the handheld device and look at them on the desktop PC as shown in Figure 19 (bottom part). In this case, we have partial migration from the device maintaining the control part to the one that will visualize the result of the user interactions.

Another example of partial migration can also be seen in a museum guide application developed at ISTI Lab (see Figure 20). Every visitor of the museum has a mobile PDA guide which also provides some quiz games that have as a subject the museum artworks. In the considered case, there is a game representation on a large screen, which is used in order to enable multiple users to share the game and discuss together about it. Differently from the representation on the mobile platform (see part A of Figure 20), after partial migration the interactive controls are available only on the PDA interface (of the user who is the actual controller, see Figure 20, part B1), while the question, the higher resolution images, and the game results are shown on the larger screen (see Figure 20, part B2).

Figure 20: An example of Partial Migration (from a PDA to a PDA+ large shared screen-based device)

6 Examples of Guidelines Application in Project Prototypes

In this section we provide some examples of guidelines that can be derived from the analysis of the prototypes implemented by industrial project partners.

6.1 Examples from The Social Game

The Social Game is a complex web application that includes several elements: Chat, Betting, IPTV, Additional Content, Racing Game, which have been described in details in D5.1 - Initial application requirements and design. Figure 21 shows a screenshot of the PC version of the Social Game. In this paragraph, an example of a possible adaptation of the PC version of the UI to a mobile phone is described.

Figure 21: Social Game, PC version

Different migration cases have been taken into consideration, most of them related to a partial migration from PC to mobile phone. In particular, since the Social Game is a very complex application, only some elements can be run on a mobile device. For this reason the application has been split into separate elements and the UI

design reflects the same concept, in a way that the single elements can be easily isolated and migrated separately.

The cases described in this paragraph should be considered as preliminary examples and the choices that will be made in the final prototype may differ from the considerations made so far, depending on the final project requirements.

In order to cope with the reduced size of the mobile display, an entire screen has been dedicated to each migrated element. The user can navigate through the different elements of the application through tabs. If a single element needs more than one screen, it can be further split in different screens, each one related to a different functionality (this can be seen as an example of application of the guidelines provided in Section 4.1). The screens are accessible through additional tabs (lower tabs) located under the main tabs (upper tabs).

In case of migration of more than one element from PC to mobile phone, to preserve the continuity of a task, the migration platform must consider the current focus of the user, and show on the mobile display the element which the user was interacting with before the migration.

Figure 22 shows the PC version and a possible mobile version of the Additional Info element, on the upper and lower part, respectively.

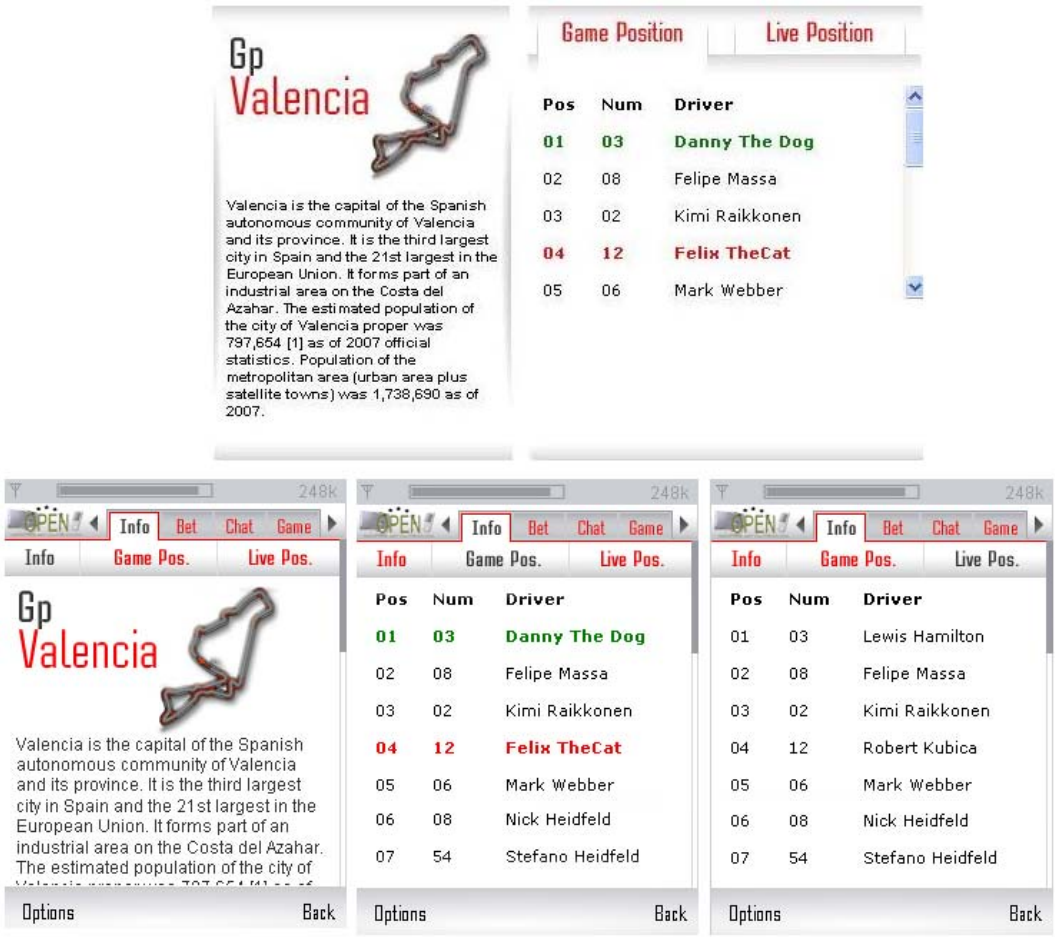


Figure 22: Additional Info element

In the PC version, the element is split into two main areas: the description of the circuit and the drivers' positions. The last area is organized in two tabs, containing the real live positions and the racing game positions merged with the real live positions. In the mobile version, three different tabs for Info, Live Position and Game Position have been used, due to the reduced size of the mobile phone display.

Figure 23 shows the PC version and a possible mobile version of the Chat element, on the left and right part, respectively.

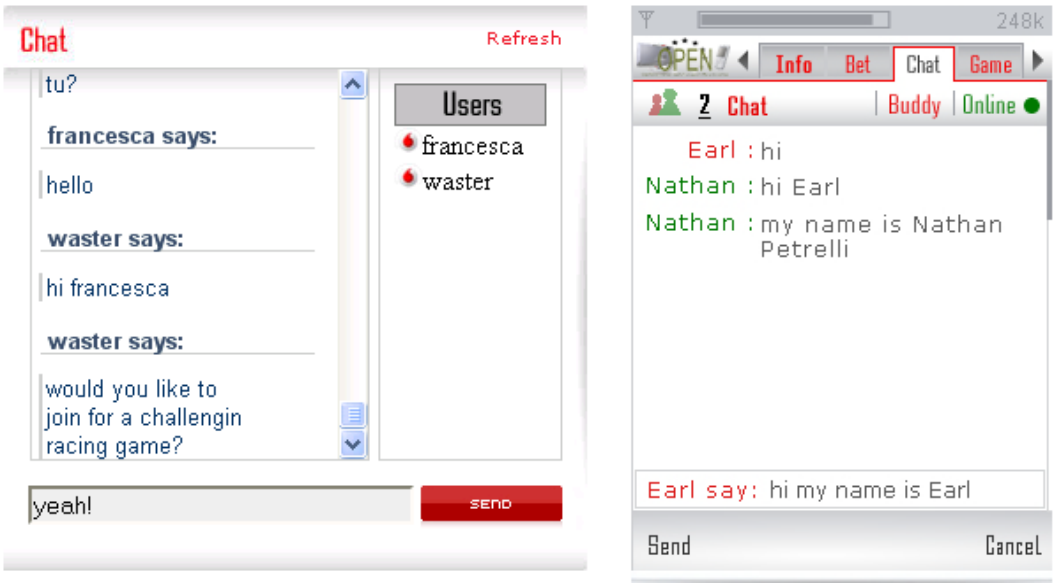


Figure 23: Chat element

Most of the screen in the mobile version is dedicated to the message area, which is the most important element of the GUI. As mentioned before, the user can navigate among the various elements through the upper tabs, and she/he can control the various chat functionalities using the lower tabs. While in the PC version the buddy list is always visible, in the mobile version the user can see the friends by selecting the Buddy tab and switching among them through a drop-down list. Moreover, the user can change her/his state by selecting the rightmost tab, whereas in the PC version she/he can change state by clicking on her/his name in the users list.

Figure 24 Shows the PC version and the mobile version of the main Betting area.

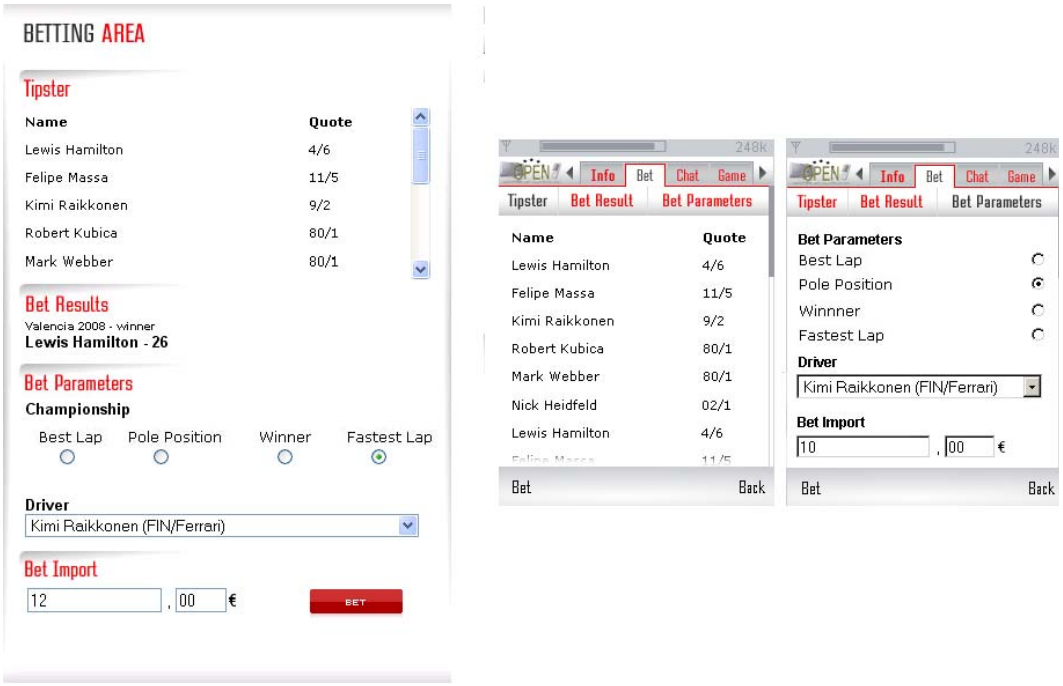


Figure 24: Bet element

While in the PC version all the functionalities are available to the user, in the mobile version the functionalities have been split in two screens, where the first one contains the information about the drivers and the second one allows the user to make choices and to submit the bet. In order to cope with the reduced screen size, the Radio Button layout for the Bet Parameters has been changed from horizontal to vertical.

Once the user submits her/his bet, in the PC version the various steps of payment selection and data confirmations are shown through a set of subsequent pop-ups. Figure 25 and Figure 26 show how the pop-ups are handled in the mobile phone. Specifically, for each pop-up there is a dedicated screen where the user confirms or cancels choices with the softkeys.

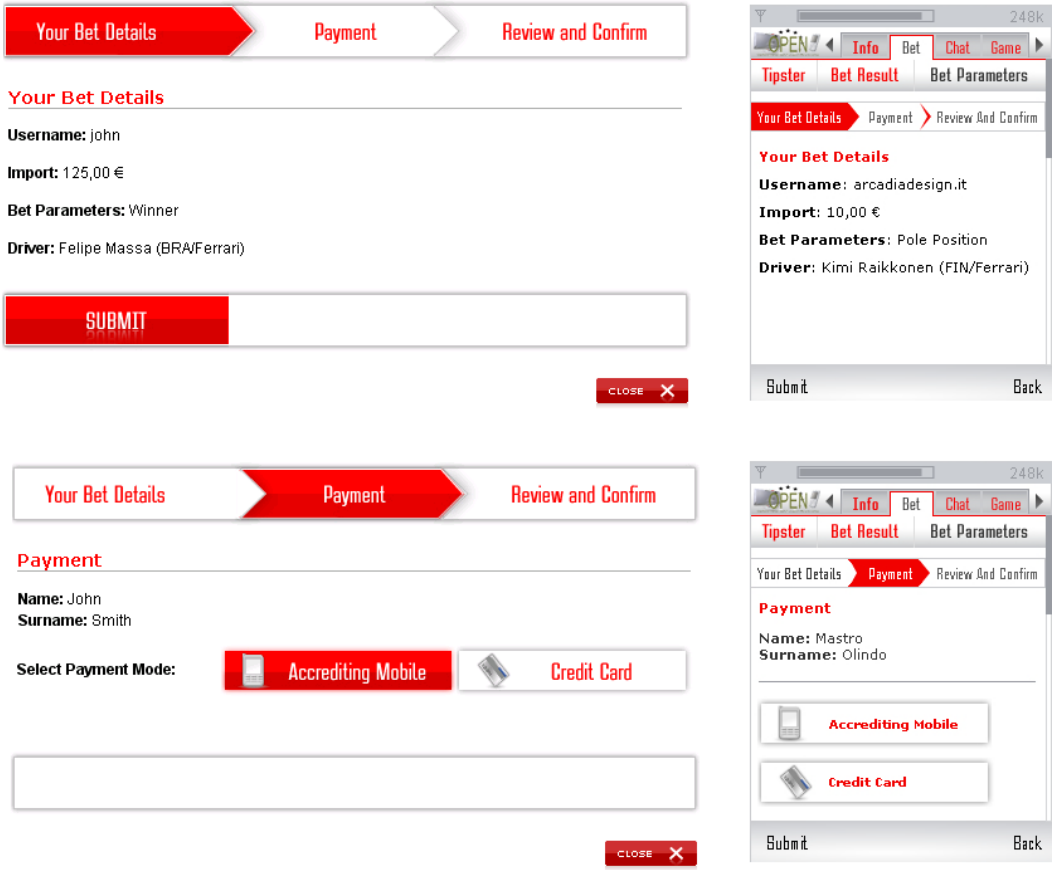


Figure 25: Bet details and payment modality selection

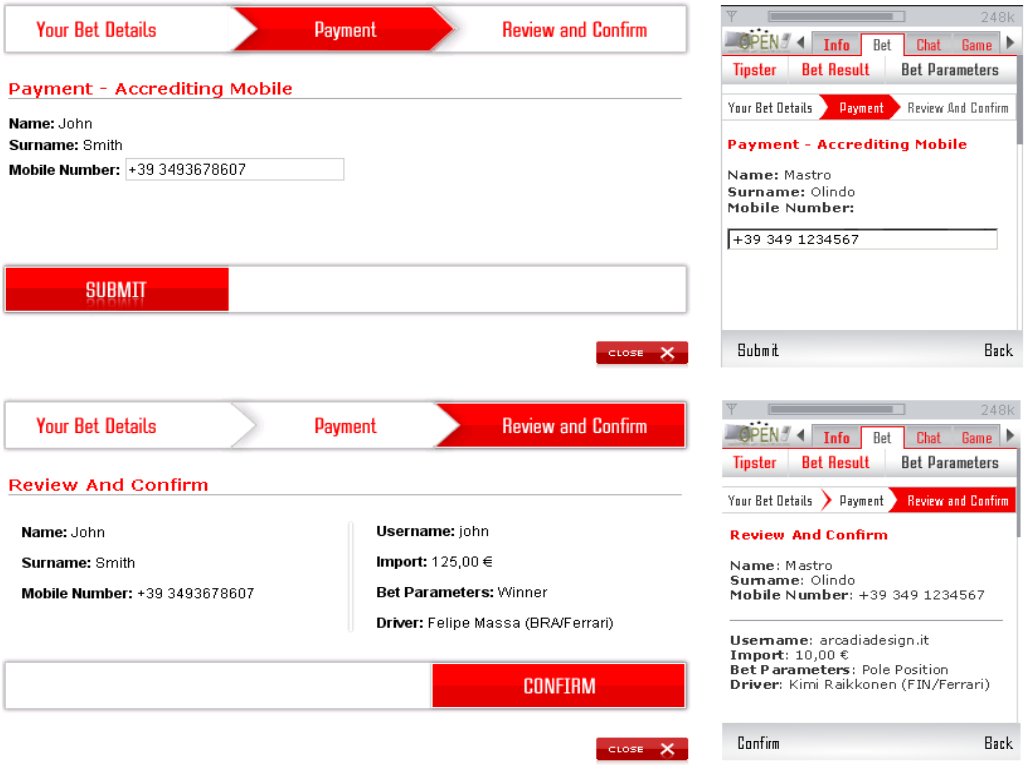


Figure 26: Accrediting and confirmation

6.2 Examples from the Emergency Scenario

The Emergency Scenario is an aggregating scenario in which applications belonging to various experts are merged onto a high-resolution multi-touch, multi-user wall-sized device (hereafter "smartwall"). This scenario, where multiple users use the same device, is different from the other scenarios where one user might use multiple devices.

Some usability aspects that are specific to this scenario and which need to be considered are the allocation of screen real estate to experts, the control of who is in control and the clarity of which controls and display elements belong to which experts. These aspects are closely tied to the *transition phase* when the application is being migrated from one device to another. In addition *task continuity* for all the experts needs to be assured as experts join and leave the smartwall. Furthermore, adaptations to the smartwall characteristics like high-resolution and multi-touch need to be considered.

Guidelines aimed at assuring usability and task continuity for this scenario are presented by means of an example involving two experts. It is assumed in this example that the source platforms run the same application as the target platform. The only difference being that the target platform has much higher resolution, greater size, and is also multi-user, multi-touch. In general, both source and target would probably rely on the same technology, e.g., Microsoft Silverlight.

Note that the example screen shots shown in this section should be considered as preliminary ideas, since the final prototype may differ from them, depending on the final project requirements.

The context of the emergency scenario is that a number of different experts assemble at an emergency control center to plan the response to a flood. To do this they share a smartwall. Figure 27 shows the initial state of the smartwall before the arrival of any experts.

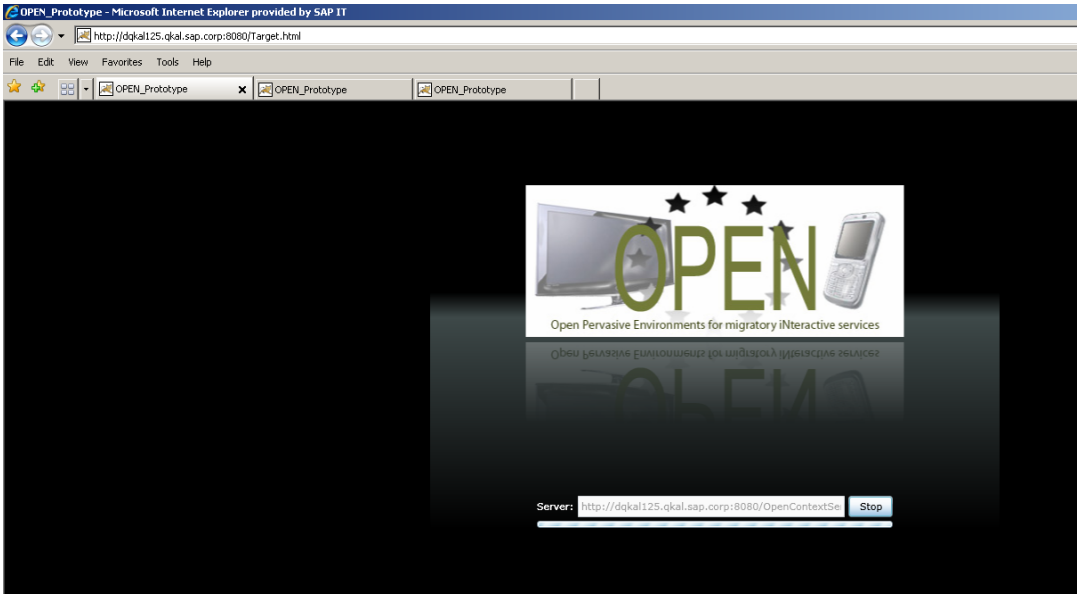


Figure 27: Smartwall at the Emergency Planning Center : Initial State

In this example, two experts arrive at the planning center, one for flood management, the other for traffic management.

The first usability issue is how to coordinate migration in a way consistent with the wishes of the manager at the control center. One guideline is that the migration should give him the option of retaining or regaining control of migration or, alternatively, of allowing experts to initiate migration. In the first case the manager would decide which applications to migrate to the smartwall and when. In the second case each expert would migrate on their own initiative.

In either case another guideline is that in this situation the experts are known by name, so rather than display device names the migration client should display person names. A further guideline is that it should be easy to tell from looking at the smartwall which experts are represented on it (which means: which experts have an application currently shown on the smartwall). This guideline might be implemented by various means. One possibility is to display a count of experts prominently and to also mark the controls of each expert, e.g., with a personal icon. The personal icons and associated names might also be displayed in a box. If screen space is an issue, the application could display just the number of experts and on click or mouse-over display the names.

In case the migration is initiated by the experts, the smartwall should be offered as a preferred choice to make migration more seamless. This might be done by putting it first in the list and making it more pronounced, e.g., larger text or more pronounced character style or color. Also, the other devices present could be represented by expert name rather than device name. Alternatively, in the situation of the planning room, the smartwall might be the only option offered for migration.

In case the manager retains control of migration, one guideline is that experts should be informed of the pending migration and asked to consent to it. This avoids feelings of loss of control.

Figure 28 shows the situation where the manager has opted for expert-initiated migration. So the expert for flood management sees the smartwall as a migration target in the Migration Control area.

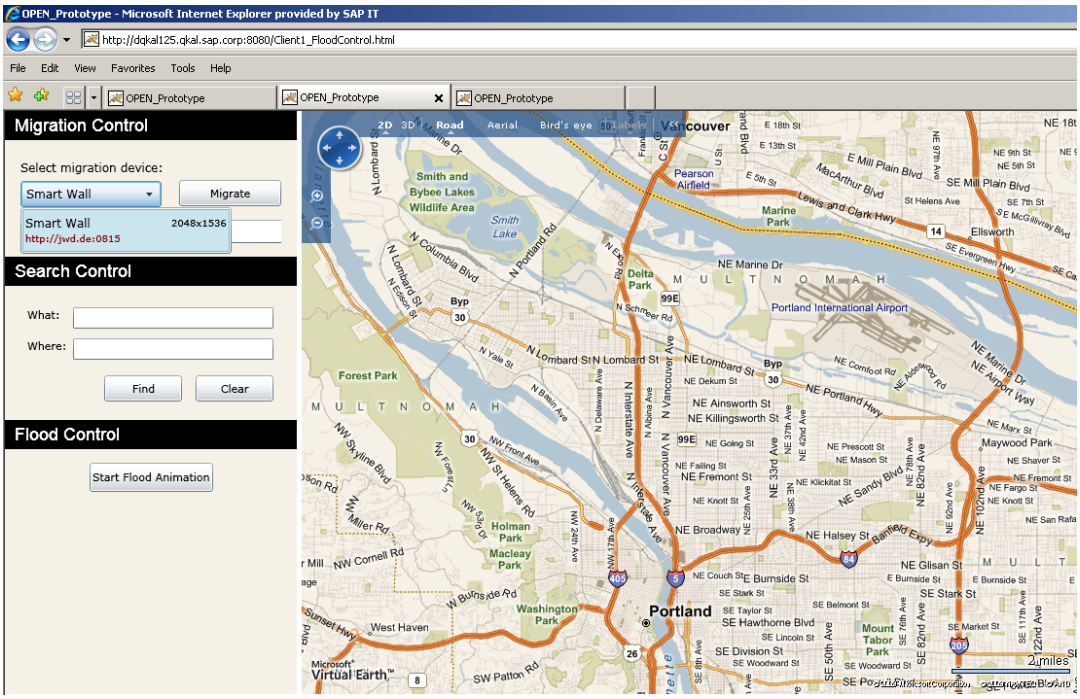


Figure 28: Flood Management Application running on the Expert's Device

The previous screenshot shows the application – *Flood Management*. Each client has a *Migration Control* to manage migration to individual target devices and a *Search Control* to search for specific points of interest, which are to be found and displayed on the map as pins. Specific to the *Flood Management* client is the *Flood Control* panel which allows control over the flood animation.

The expert who is working on flood management moves to a point of interest on the map and starts the flood animation. Figure 29 shows the results of these actions.

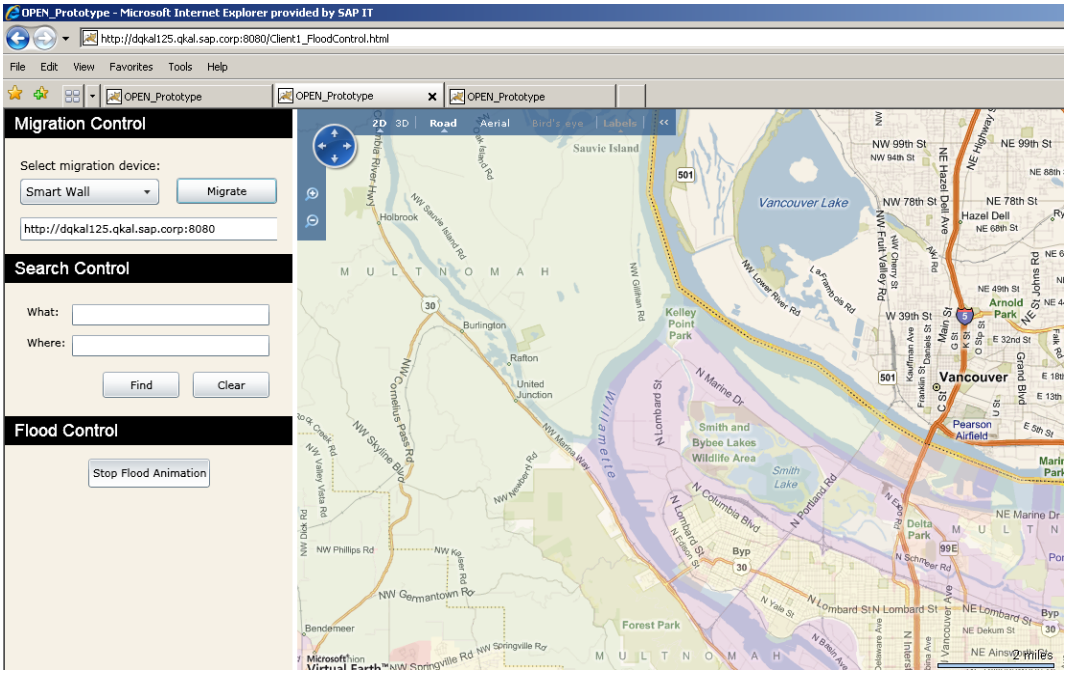


Figure 29: Results of running the Flood Animation at a particular Point on the Map

In order to migrate the Flood Management Application to the smartwall, the expert chooses the smartwall and clicks the *Migrate* button. As a result the smartwall will display this application. All of the following are transferred to the smartwall: map position, zoom level, flood animation status, any values in input fields and the cursor location. In addition the *Flood Control Panel* is transferred to the smartwall. The *Flood Control Panel* includes the user interface and the logic behind it, thus transferring not only the look but also the functionality of the application to the smartwall. So the flood management expert can continue working on the smartwall, and can also take advantage of its multi-touch screen. Figure 30 shows the smartwall after migration of the flood management application to it.

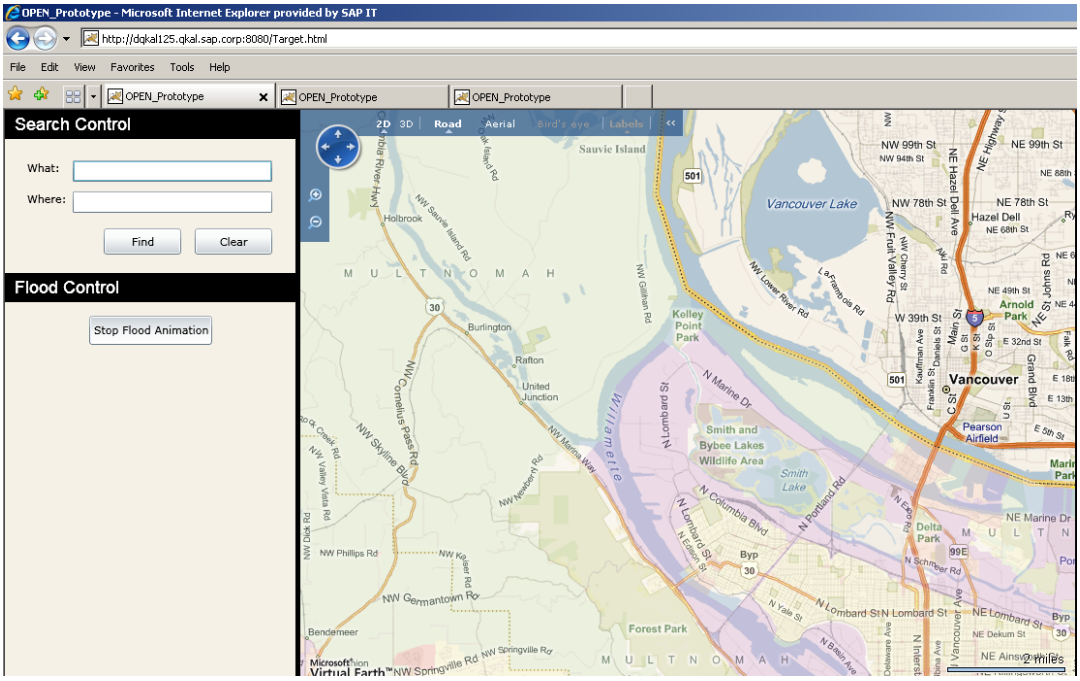


Figure 30 Smartwall after the Flood Management Expert has migrated to it

Two guidelines are relevant to adapting the user interface to the smartwall. One is that high-resolution images should be substituted for lower resolution, if the look of the application is important. Another is that if some participants will be far from the smartwall, larger font sizes (between 24–36pt) and more readable font styles should be used in order to guarantee the readability of any text. Note that adaptation of interactors to take advantage of multi-touch capabilities is another option that needs to be considered.

Another aspect of the scenario that needs to be considered is the potential need for experts who leave to synchronize their device with the smartwall before they go. This option enables task continuity for departing experts, so it should be offered to them as a choice.

A second client application in the example being used is the *Traffic Management* application which also has the usual *Migration and Search Control* plus the individual *Traffic Control* to display the current traffic situation (how slowly or quickly traffic is moving on the roads) and/or the driving times from the center of the map. Figure 31 shows this application.

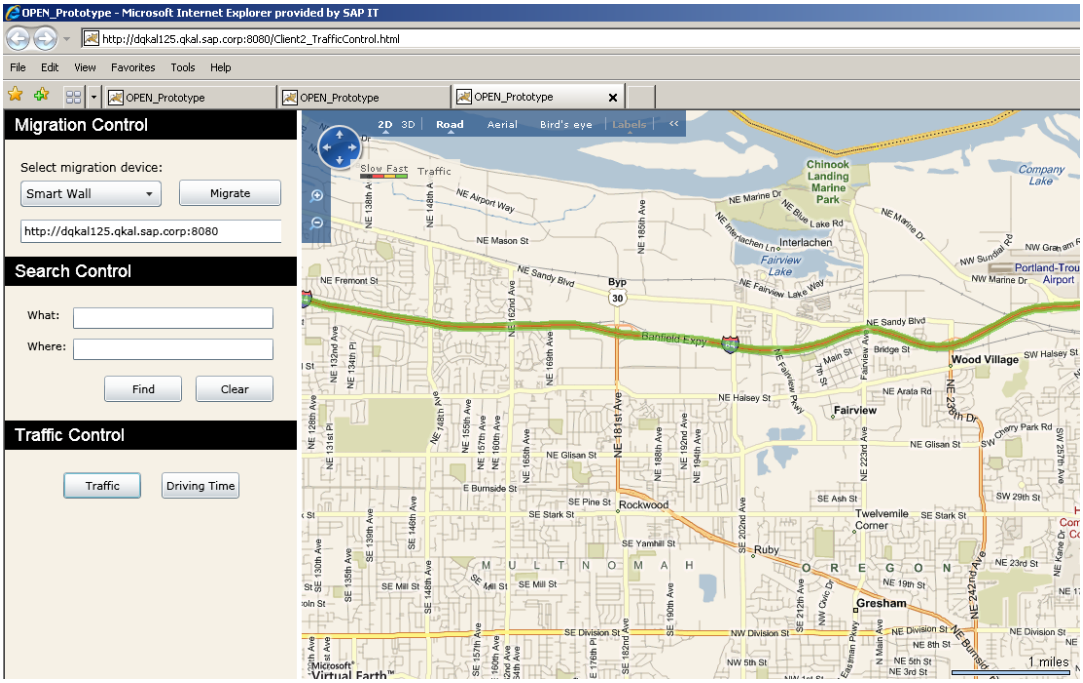


Figure 31 Traffic Management Application

When the traffic expert migrates to the smartwall his application will be merged with the other applications (in this case just one) on the wall. Some guidelines relate to this merging. First, as with migration, the manager should have the option of controlling how the merge happens. Some examples of choices he should have are leave the map position and zoom level as it is on the smartwall, move it to the new expert's values or automatically calculate new values that make it possible to display the positions of all experts' activities on the smartwall.

To continue with the example, assume the manger choose the automatic option. Assume further that the traffic expert is interested in the traffic situation at a location other than the location the flood expert is inspecting. Thus, after pressing the "Migrate" button in the *Traffic Management* client the smartwall needs to not only merge and display the controls on the left but also to update the map on the right.

To optimally display the center points of both experts, the application needs to calculate a bounding box that contains these center points. It also needs to set the zoom level and center point of the updated map accordingly. Figure 32 shows such an updated map.

To enable the two experts, Flood expert and Traffic expert, to proceed with working on their individual problems on the smartwall (task continuity) it is necessary to identify the center points of each expert on the map. In the example in Figure 33, this is done by so called push-pins (in red), which can hold meta-information.

For identification the push-pin meta-information contains the unique ID number which identifies the expert. One guideline for making this more usable is to use the person's name instead of an alphanumeric identifier as in Figure 32. Another

guideline is to visually distinguish the push-pins in such a way that each can be associated with the corresponding expert.

Another usability guideline is to make the association between controls (on the left) and experts clear. One way to implement this would be to assign icons to experts, and to include the icons in the experts' controls. The same or similar icon might be used for the push-pin. As to focus, probably it is best if focus is not updated when a new expert joins the wall.

A further guideline relates to preserving input when merging common elements like the *search control, what field*. Input in a field should not be lost when migration occurs and a user at the smartwall has just entered data in that same field. One solution to this problem would be to add the migrating data to a field-specific list of values that can be retrieved on request.

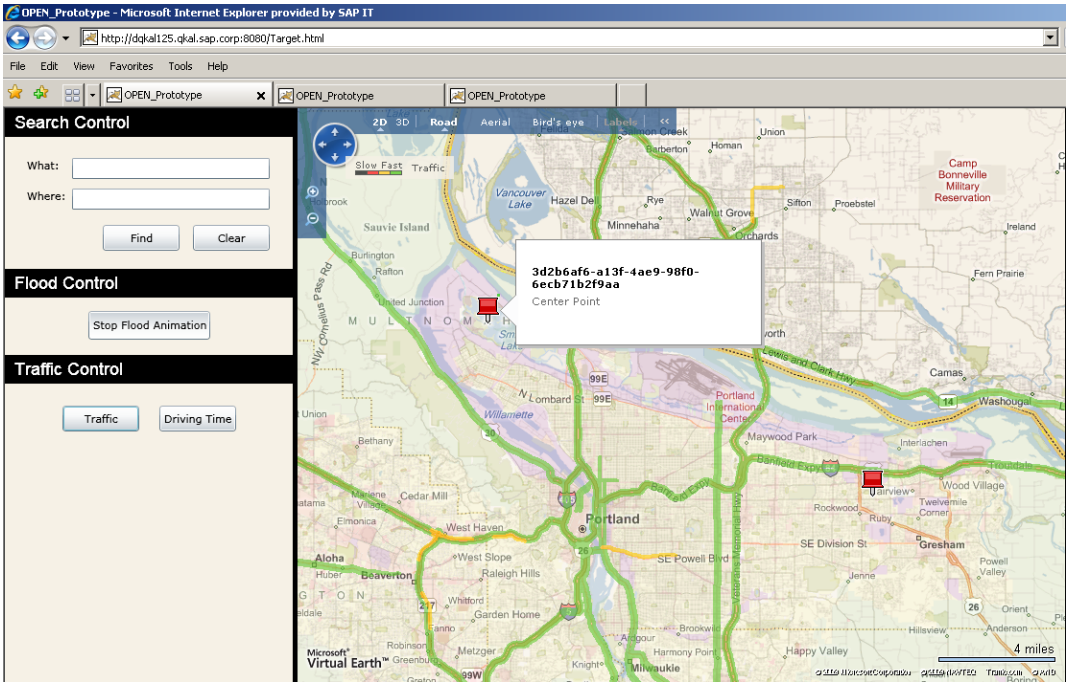


Figure 32 The Smartwall after the Traffic Expert has joined

Since not only the static information like map position and zoom level, but also the functionality of the individual controls (Flood- and Traffic Control) are available after migration, the experts can continue working. They can use the functionality of the individual controls to manage the flood animation, traffic and driving time calculation and adapt the map. The experts can combine their results and go on with further tasks like searching for the most appropriate fire department by displaying the individual driving times as shown in Figure 33.

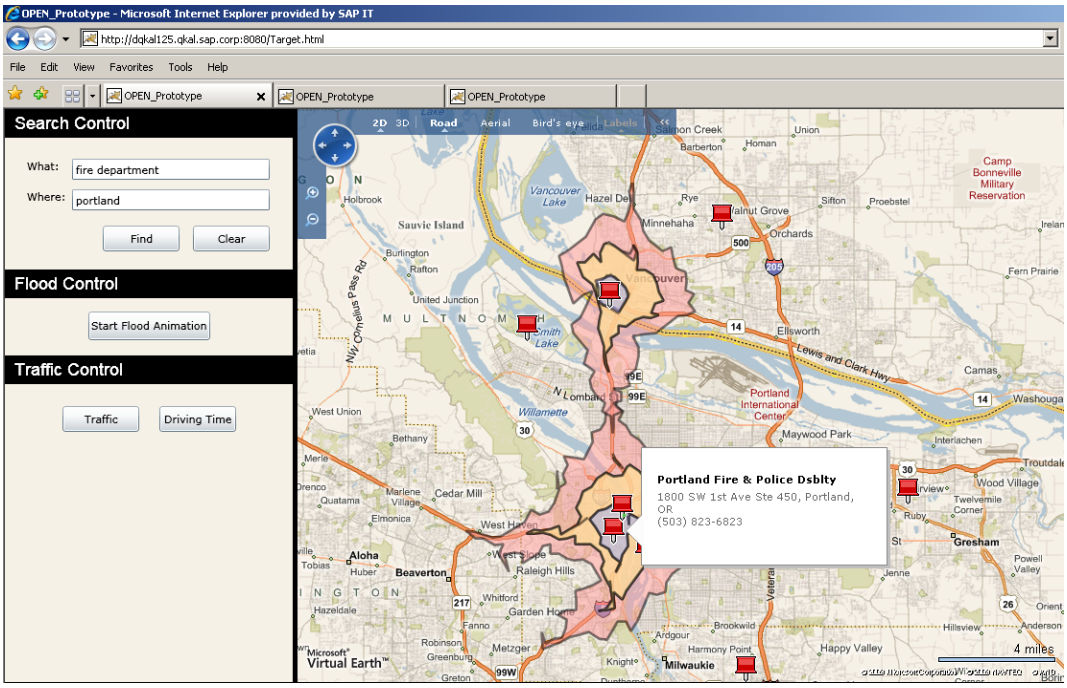


Figure 33 The Smartwall after both Experts have been working at it

This example has considered the case of two experts, but there could be several more. Thus, guidelines about sharing the screen space become important.

One guideline is that when more than two experts are involved, the application needs to provide ways to move and/or hide and unhide the controls for the different experts.

In summary, this example has demonstrated the need for a range of usability guidelines for an aggregating scenario like the Emergency Scenario, which brings together the applications of multiple experts onto one shared smartwall. These guidelines can be categorized into control issues, the clarity of which controls and display elements belong to which experts, adaptation to the smartwall and, finally, allocation of screen real estate to experts.

Control issues include who is in control: of the migration process and the map merging options. The migration process should enable the central planner to retain or regain control of the migrations, while at the same time ensuring that the experts do not experience loss of control. One aspect of this is preserving field input when merging. Migration should also be made more seamless by adapting it to the situation of the central planning room and using person names rather than device names. Moreover, departing experts should be empowered to continue their tasks by synchronizing with the smartwall before they leave.

Clarity of what belongs to whom on the smartwall is essential, and might be achieved by means of person names and icons to identify which experts are present on the smartwall with which controls.

Potential adaptations to the smartwall: include changing image resolutions, fonts and taking advantage of multi-touch multi-user capabilities. Additionally, when a number of experts use the wall simultaneously it is necessary to add options to hide or unhide controls for the different experts.

7 Conclusions

The aim of this deliverable is to discuss some rules and techniques (for both total and partial migrations), which can be exploited for delivering effective user interfaces that are able to preserve task continuity and are adapted to the target device in various scenarios that have been identified (desktop/mobile, tv-based/mobile, etc.) and judged relevant by the OPEN partners. We have shown various examples taken from prototypes developed by the OPEN partners available at May 2009.

The set of adaptation rules are continuously updated depending on the experiences gathered and we plan to further refine them after the evaluation phase in the project.

8 References

(Bandelloni & Paternò, 2004) Bandelloni, R., Paternò, F.: Flexible interface migration. IUI 2004: 148-155

(Bandelloni et al., 2005) Bandelloni, R., Berti, S., Paternò, F.: Analysing Trans-Modal Interface Migration. INTERACT 2005: 1071-1074

(Berti, et al., 2005) Berti, S., Paternò, F., Santoro, C.,: A Taxonomy for Migratory User Interfaces. DSV-IS 2005: 149-160

(Denis & Karsenty, 2004) Denis, C., Karsenty, L., Inter-Usability of Multi-Device Systems - A Conceptual Framework. Edited by: Ahmed Seffah, Homa Javahery. (2004), pp. 373-384.

(Florins et al., 2004) Florins, M., Trevisan, D.G., Vanderdonckt, J. The Continuity Property in Mixed Reality and Multiplatform Systems: A Comparative Study. Available at http://www.isys.ucl.ac.be/bchi/publications/2004/Florins-CADUI2004.pdf

(OPEN D5.2) Cherchi, G., Mureddu, F., Initial prototype applications. January 2009. OPEN Project.

(OPEN D6.1) Stecca, M., Marzorati, S., Grasselli, A., Piunti M., Usability criteria for project phases: use cases selection, design, development, test and deployment, November 2008. OPEN Project.

(Paternò et al., 2008a) Paternò, F., Santoro, C., Scorcia, A.: User Interface Migration between Mobile Devices and Digital TV. TAMODIA/HCSE 2008: 287-292

(Paternò, et al., 2008b) Paternò, F., Santoro, C., Scorcia, A., Automatically adapting web sites for mobile access through logical descriptions and dynamic analysis of interaction resources. AVI 2008: 260-267

(Pyla et al., 2006) Pyla, P.S., Tungare, M. and Pérez-Quiñones, M.A., Multiple User Interfaces: Why Consistency is Not Everything, and Seamless Task Migration is Key. Proc. of the CHI 06 Workshop on The Many Faces of Consistency in Cross-Platform Design, 2006, 55-60.

(Richter et al., 2006) Richter, K., Nichols, J., Gajos, K., Seffah, A. (editors) The Many Faces of Consistency in Cross-Platform Design. Montreal, Canada, April 22-23, 2006, CHI Workshop. Available at: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-198/

(W3C, 2004) Authoring Techniques for Device Independence. W3C Working Group Note, 2004. Available at: http://www.w3.org/TR/di-atdi/

(W3C, 2008) Mobile Web Best Practices 1.0, Basic Guidelines, W3C Recommendation 29 July 2008. Available at http://www.w3.org/TR/mobile-bp/