Chapter 5
User Interface Migration Based on the Use of Logical Descriptions

Giuseppe Ghiani, Fabio Paternò and Carmen Santoro

5.1 Introduction

Nowadays people are ever more exposed to ubiquitous environments, which are characterized by the availability of various interactive devices with different interaction resources. Thus, the possibility to opportunistically exploit the resources that such contexts offer (e.g., moving from stationary to mobile devices) is invaluable for providing an effective interaction experience to the user. In this context, interactive migratory user interfaces offer the added value of enabling users to migrate across various types of devices while preserving the task continuity. This implies that there should be the possibility to select a target device and activate on it a version of the user interface adapted to its features with the same state as on the source device. The state of a user interface includes the values entered or selected by the user, the content, the cookies, etc.

Various types of migration can be identified depending on the number of source and target devices or whether the entire user interface or only a part of it migrates. In particular, partial migration means moving only a portion of the interactive application (namely: some components) to another device in order to better exploit its interactive resources. The typical scenario is a user who is interacting with a desktop or large screen system and then for some reason has to leave, but wants to continue the interaction through a mobile device with only a part of the application. This can be either because of its complexity or because of some limitations in the mobile device (e.g. iPhones do not support Flash applications). This is particularly important with mashup-like applications, which tend to be particularly complex and made up of various perceivable components.

Model-based approaches (see for example Eisenstein et al. 2001; Paternò et al. 2009a) have shown good potential in managing the complexity of multi-device environments. In such approaches there is usually a distinction between abstract (modality independent) and concrete (modality dependent) logical descriptions,

F. Paternò (✉)
CNR-ISTI, HIIS Laboratory, Via G. Moruzzi 1, 56124 Pisa, Italy
e-mail: fabio.paterno@isti.cnr.it

F. Paternò (ed.), *Migratory Interactive Applications for Ubiquitous Environments,*
Human-Computer Interaction Series, DOI 10.1007/978-0-85729-250-6_5,

which makes it possible to better support interoperability across various types of devices and implementation languages. Indeed, such approaches generally work by progressively refining these descriptions, from higher, to more concrete ones, till the implementation level. Abstract descriptions enable the designer to specify user interfaces using a platform-independent vocabulary of UI objects (also called "*interactors*"). Therefore, at this level the UI specification includes generic *selection* objects, *edit* objects, … etc. Concrete UI descriptions further refine abstract objects by adding platform-dependent details. So, an abstract selection object can be refined into various concrete interactors, such as a pull-down menu, a radio button, a check-box. It is worth noting that the concrete UI objects still do not refer to any specific implementation language: such details are added by a final refinement step.

Our approach aims to provide a general solution for Web applications implemented using (X)HTML, CSS, and Javascript. It can also support applications based on languages such as JSP, PHP, ASP because it considers one page at a time on the client side. Thus, it adapts only what is actually accessed by the user. Another advantage of the solution proposed is that it makes Web applications migratory regardless of the authoring environments used by the developers. Without requiring the use of any specific tool in the development phase, it enables the applications to migrate, even if the developers never considered migration. This is obtained through the use of reverse engineering techniques that create the logical descriptions of the Web pages accessed on the fly, which are then adapted for the target device. Lastly, an implementation with the state of the source version is dynamically generated.

The subject of partial migration raises a number of issues, which have been addressed from different viewpoints in other work by the research community. Partial migration can be related, to some extent, to the issues connected with Distributed User Interfaces (DUI). To this regard, in (Melchior et al. 2009) a toolkit for deploying distributed graphical UIs is presented. In our solution we opted for a distribution down to the granularity of the single interactor but no deeper, since we judged such fine granularity unimportant for our goals. In addition, this solution requires that the user interface be implemented using an extension of the Tcl/Tk toolkit, while we are interested in solutions that allow to partially migrating any Web application developed with standard W3C languages (XHTML, CSS) and JavaScript.

The issue of distributing a user interface onto multiple devices is also analysed in (Pierce and Nichols 2008), with particular attention to how to leverage legacy applications to attain the new distributable features easily. However, it is worth pointing out that the relations on which this infrastructure is based includes a strong limitation that narrows the set of devices that can be interconnected to each other (only the personal devices of a user). Instead, fully migrable applications should be able to opportunistically exploit the devices in the environment (even the devices not owned by the users but accessible to them).

The study in (Edwards et al. 2009) describes an infrastructure (Obje) that supports building interoperable systems without having prior knowledge about them ahead of time. While the motivation of the Obje architecture was to provide an infrastructure for opportunistic interoperation in device-rich environments (as in migration) this approach basically addresses problems of interoperation rather than migration. In

general, we can notice that while a number of model-based approaches have been put forward for the design of multi-device interfaces, and in particular for mobile applications (see for example Eisenstein et al. 2001), none of them has shown a general solution able to work on any Web application implemented according to the W3C standards for supporting partial user interface migration from desktop to mobile systems.

In this chapter we present a solution supporting UI migration (both total and partial), its main characteristics, the architecture of the migration platform supporting it, and also provide examples of migration for a Web application, in order to show its use and potentialities.

5.2 Architecture

The starting point for the work presented in this paper was the solution described in (Paternò et al. 2009a), which supports migration of only entire user interfaces (total migration), without providing any possibility to migrate only parts of them. In that work an architecture based on a number of modules was proposed supporting dynamic reverse and forward engineering. The modules identified were: the Reverse Engineering module, which builds the logical description of the source page considered; the Semantic Redesign, which transforms the source logical concrete description into another one, tailored for the target platform; the State Mapper associates the state of the current Web page to the logical description automatically generated for the target device; the Generator generates the corresponding implementation. Such implementation is then sent to the target device so that the user can immediately find the adapted page, with the state resulting from the interactions already carried out with the source device; the Proxy Server is an intermediate layer: as a proxy, it captures all the communications occurring from the user browser to the application server and vice versa; the Migration Orchestrator (as we will see in Chap. 6) handles the communications with the different modules involved in the migration.

The Reverse Engineering part is able to build corresponding logical descriptions from (X)HTML, CSS and Javascript implementations. If the Web application contains Flash or Java applets, then the Reverse Engineering is not able to analyse its code. In this case, the applets are either replaced with alternative content provided by the application developers (such as images) or passed to the target device "as they are", if the target browser is able to execute them.

The Semantic Redesign module transforms the concrete description (specific for the source platform) to the one that refers to the target platform. The concrete descriptions are independent of the implementation language, while the abstract descriptions are even independent of the interaction modalities. In general, concrete descriptions assume the existence of some interaction modalities but are independent of the implementation language. At the abstract level there are, for example, concepts such as selection, edit, activate, while at the concrete level for a graphical device for example, the selection object can be refined into a list or a radio-button or a pull-down menu or other similar techniques. Such elements can be implemented

in different languages. The abstract and concrete vocabularies contain concepts for structuring the user interface as well, such as grouping and relations. The semantic redesign transformation aims to map source concrete interface elements into ones that are more suitable for the interaction resources of the target device. The semantic redesign uses the abstract level to identify the type of interaction to support and then identify suitable, concrete refinements for them for the target platform. Thus, for example, in a desktop-to-mobile transformation the possible target concrete elements will be characterized by a more limited usage of screen space while preserving their semantics (i.e.: the effect that they have on the interactive application).

The objective of the State Mapper is to update the concrete user interface for the target device (and which has been delivered by the semantic redesign module) with latest information regarding the state of the UI contained in the DOM file of the source page just before migration. After having obtained the new concrete user interface description for the target device (updated with information about the state), the Generator module builds the final user interface specified in an implementation language supported by the target device considered. The Proxy Server module plays a role whenever a browser on a client device requires access to a certain Web page. Indeed, every request to the application server is filtered by this module, which accesses the application server to obtain the page and also annotates it by including scripts, which will enable capturing the UI state.

The previously described solution was not able to support partial migration because this feature implies the ability to select a subset of features and then migrate them to the target device. In order to obtain this we have again exploited the possibilities offered by the use of logical user interface descriptions. Indeed, in the MARIA language (Paternò et al. 2009b) we use, it is possible to describe the logical structure of a user interface through interactor composition operators that indicate groups of logically connected elements or relations among such groups (e.g. a set of controls are associated with a certain form).

When the user selects the migration, the migration server Orchestrator communicates with the Reverse module, which produces the Concrete User Interface (CUI) description associated to the current page and passes it to the Migration Orchestrator. If the user triggers a migration request when a subset of the components is selected in the page, then the migration is considered to be partial. In this case, the Migration Orchestrator requests a subset of the CUI from the Partial Migration module, according to the sub list of components selected by the user, and forwards it to the Semantic Redesign, thus skipping the Reverse phase (which, however, had been executed previously to create the original CUI of the entire desktop interface).

The web migration support has been designed to be run as integrated with the OPEN Migration Service Platform or as a standalone support. In the following, more details are provided about the two possible modalities.

5.2.1 OPEN Platform Integrated Orchestration

When the web migration support is run as part of the integrated OPEN platform, the migration orchestration is mainly provided by a dedicated module of the OPEN

platform. Such module provides the migration support with the list of migratory components of the OPEN-aware web application. This is possible because the application had previously registered them and the Migration Orchestration had bound them with unique ids. The web migration support also notifies the platform orchestration about the components for which the migration has been requested and about the result of the migration (completed/failed). All the communications with the OPEN platform are made via XML-RPC according to the OPEN specifications.

The advantage of such strategy is to exploit the features of the OPEN-awareness offered by the web application: for example, when the migration of a subset of components is completed and the application is notified by the Migration Orchestration, the components in the source device are deactivated.

5.2.2 Stand-Alone Web Migration Orchestration

When run as stand-alone, the web migration is able to autonomously generate the list of migratory components. This is done by the reverse module that takes a web page and creates a logical description for it. Every page navigated via the web migration support is annotated by the migration proxy, which fills each of the main page components (such as DIV, FORM, TABLE, etc.) with a unique id (if it was not originally present). Such information is fundamental for the platform to identify the components selected by the user for the partial migration.

The stand-alone web migration is then able to entirely perform the migration process. The advantage of this solution is that every valid web application can be migrated, even if it is not OPEN-aware (i.e.: even if it was not developed according to the OPEN specifications).

5.3 An Application Example of Total Web Migration

In this section we present an example of total migration involving a web site of an airline (http://www.ba.co.uk), through which the user is able to perform a search in order to get available flights for a business trip.

Figure 5.1 shows a Web page for the considered example, visualised on a desktop device. As you can see, it is structured in different sections, in particular in the left hand side there is a form for arranging a trip. In the considered scenario, the user interacts with this site and at a certain point s/he decides to migrate to a mobile device since s/he has to leave. Then, s/he activates a migration to a mobile device. As a consequence, the current page is analysed by the OPEN Migration Platform, which will derive (through a reverse engineering step) a logical description of this page. This description will be used as input for another module that adapts it for a mobile device (this will be carried out by the Semantic Redesign module). From this new logical description, another UI specification for the mobile device will be derived, with the updated state included in it.

Fig. 5.1 The desktop page of the airline company, considered in the total migration example

Figure 5.2 shows an example of the results of the transformations that the Semantic Redesign module performs. While in the desktop device the selection regarding the ticket type was implemented through a radio button, in the mobile device a pull-down menu has been used, since it supports the same activity while 'consuming' less screen space. Other modifications done by the Semantic Redesign module are, for instance, the resizing of the images, as well as the splitting of the original desktop page in more than one mobile page (if needed). As a consequence of the splitting, some additional links can be added in the split pages (see top-left part of Fig. 5.2, where a "back" link was added by this module in order to navigate back to the referring page).

5.4 An Application Example of Partial Web Migration

In this section an example of partial Web migration is presented. The example considers a stand-alone web migration (which means that it can be carried out even on non OPEN-aware web applications). The application (called 'Social Game') is

Fig. 5.2 The pages generated on the mobile device after total migration

Fig. 5.3 The social game web application in the desktop device

a game that includes different parts: there is an IPTV in the top-left part, some additional info just beside the IPTV (e.g. live race positions), information on game positions, and a chatting area displaying the buddy list where users can connect and talk. In the bottom (left) part there is a betting area for selecting the driver to bet on as well as the desired amount, while the bottom right part displays the racing game. The application also involves interaction among multiple users (see Chap. 8). An example user interface can be seen in Fig. 5.3: The goal of the game is to finish a lap in the shortest time. Since the considered application is composed of several modules, it was judged as a good case study for assessing the features of the partial migration support.

From the desktop device, the user migrates (see Fig. 5.4) both the betting area and the chat area to a mobile device (an iPhone) (partial migration desktop→mobile). The interactive selection has been implemented with a mechanism that is supported directly in the concerned web page. Whenever the user clicks on a certain component or a part of the page ("onclick" event), this part of the page (or elementary component) will be automatically highlighted with a green background color within the page (see Fig. 5.4), in order to make the user aware of the selection currently done. It is worth pointing out that, in order to unselect a previously selected region/component in the page, the user has just to click again on the same region/component, which will be removed from the list of components to be migrated.

The resulting UIs on the mobile device are shown in Fig. 5.5. In this case, a partial migration will be done since the user interactively selects the parts of the UI that are of interest for him/her.

In Fig. 5.5 you can see the resulting presentation that is displayed on the mobile device after partial migration. As you can see only the parts that have been

Fig. 5.4 The selection of the chat and the betting area components

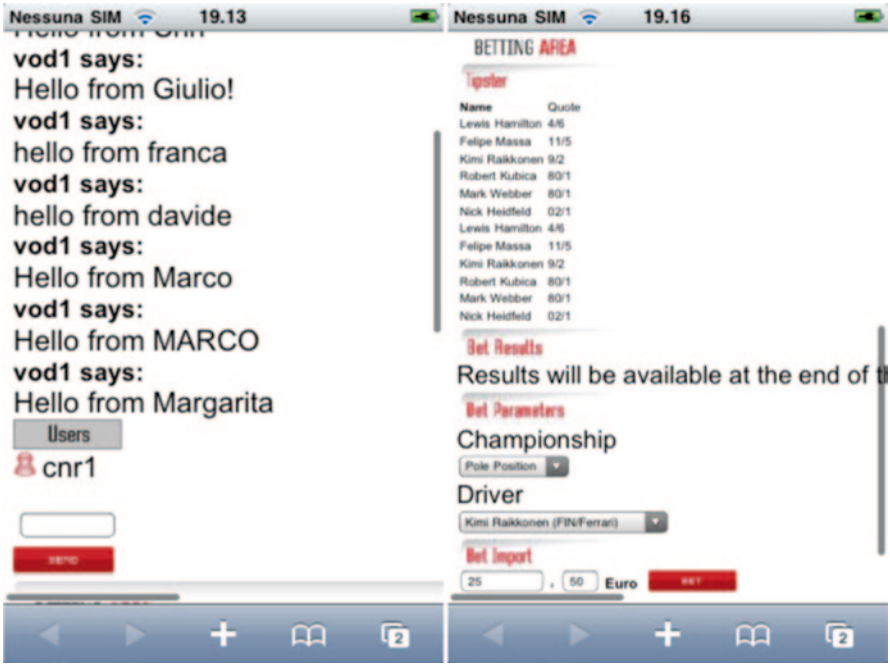


Fig. 5.5 Two migrated components displayed in the target device (iPhone)

previously selected by the user are available on the mobile device, with the state preserved (you can note that all the selections that were done in the desktop component are still maintained in the mobile device after migration).

5.5 Usability Evaluation

We conducted a usability evaluation test of our environment. The users involved were 8 male and 2 female, with average age of 30. Regarding the education level, 2 have bachelor degree, 5 had a Master degree, 3 had a PhD. All used internet desktop daily; as for the use of internet through a mobile device, 4 used it daily, 3 monthly, 1 yearly, 2 never accessed Internet through a mobile device. Furthermore, users were asked the frequency with which they played a game using a desktop device: 1 user played it on a weekly basis, 1 user reported a monthly frequency, 8 yearly. Finally, users were asked to report the frequency with which they played a game using a mobile device: 2 reported they had a monthly frequency, 4 yearly, 4 never played a game using a mobile device.

The users were also asked to describe any other application/system in which they already found concepts similar to migration. The vast majority of participants (9 out of 10) answered that they never found such concepts in other systems. Only one person reported to have found some similarities in multiplayer games on desktop.

Another question regarded which kinds of application the users believed the migration could be particularly useful for. One user declared that the migration could be useful especially for large web pages, and for any kind of "shareable" application (shared document editing, presentation viewing, …). More than one user mentioned the fact that business people, who generally have multiple devices, could mostly benefit from this system. Also, the migration support was found promising for communicating with friends/colleagues without interruptions.

We also asked participants for quantitative evaluations, using a [1..5] scale where 1 is the worst score and 5 is the best one. Then, the user interface for selecting the target device for migration was assessed by them (Average value: 4.2; Standard deviation: 0.92). Two users pointed out that a major integration between the panel for selecting the device and the panel in which the application is shown would have been appreciated (in the tested version the panels were displayed in two different tabbed panels within the browser). Such users did not like the fact that the two interfaces were shown in two different tabs and then the relationship existing between them was not immediately evident. Another person suggested to use a photo of the device in order to let the user recognize more easily the target device for migration (this could be especially useful in case of multiple devices of the same category as in this case they would have the same icon). The user interface tested contained an icon and a name to identify each device, together with a link for accessing further information.

Moreover, users had to evaluate the easiness in continuing the interaction in the target device, from the point they left in the source device (Average value: 4.1; Standard deviation: 0.73). Users said that they did not experience any problem in

the desktop device. However, some of them pointed out that not having much familiarity with the mobile device could represent a detrimental factor regarding this aspect. However, this does not seem a big issue since people interested in migration to mobile devices are obviously familiar with such devices. One user pointed out that on mobile device the easiness could decrease because the page after migration might be very different from the original one. Connected to this aspect, another user appreciated that the migration platform was able to handle the information about the last element that received the focus, and preserve it when the resulting migrated UI was presented to the user: he said that this represents a sort of reference point within the pages resulting from the splitting in the new mobile device.

Regarding the mechanism supporting the interactive selection of the UI portion to migrate, the users judged it as intuitive (Average value: 3.8; Standard deviation: 0.63). Nevertheless, several users gave recommendations for improving it. One suggestion was about the possibility for the user to enable/disable the automatic highlighting of the parts to be migrated, because some users found it a bit annoying when it occurred during normal navigation. It is worth noting that, after the evaluation exercise, we exploited this suggestion and implemented such feature and enhanced our support accordingly with the enable/disable option for the partial migration, as it can be seen in the initial part of the paper (see Fig. 5.2).

All the users agreed on the usefulness of the partial migration, and thus on the possibility of selecting only the information they are really interested in. They highly appreciated its flexibility and capability in better coping with the interaction resources of devices with less capabilities than the desktop platform. Therefore, they highlighted its usefulness especially in cases when the migration is carried out from desktop to mobile.

We also conducted further evaluations on more quantitative aspects of the migration. In particular, for the *effectiveness* of our migration platform we considered task *failures* (i.e.: when actions different from the ones required by the task list were executed by users). We conducted this test by directly observing the user while carrying out the activity. If we consider as 100 the number of all tasks executed by the entire user sample we obtained that no task caused the test abandonment and a very small percentage of tasks were not successfully completed (1.11%), all due to application errors. In particular, the deterioration of user experience was due to rare prototype bugs (which have since been corrected) regarding technological aspects and not to problems in the interface or presentation layer.

Efficiency was the second quantitative indicator about usability related to the task execution time. This parameter was measured by recording the time that was needed for carrying out the tasks. In Fig. 5.6 the execution time distribution for the Social Game is reported. The plot considers the task list execution classes on x-axis (the classes depend on the performance time, which is reported in seconds) and the number of people that fall in each range on y-axis.

Observing the distribution histogram we can see that all the data fall into two classes, which is a good result because it is a signal of the fact that different users carried out the task list more or less in a close time, and it is a positive indicator of the usability of the platform because it is perceived in the same way by different users.

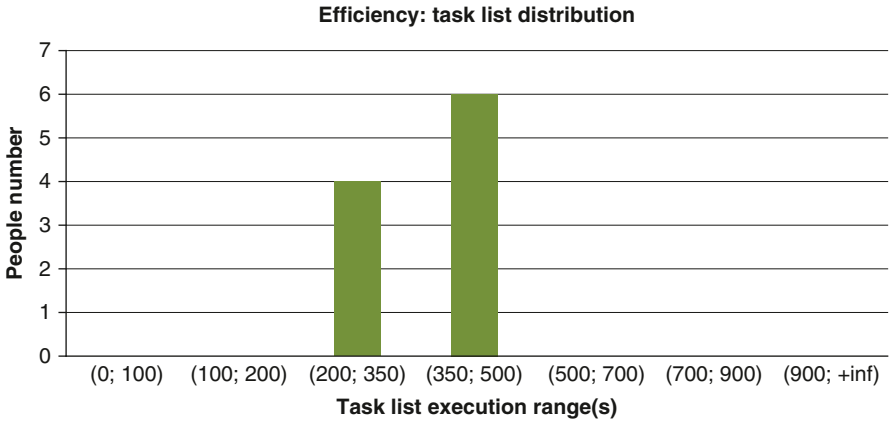
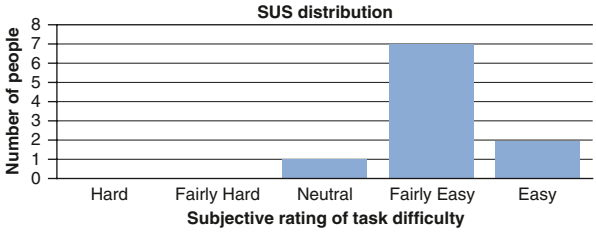


Fig. 5.6 Efficiency results

Fig. 5.7 Satisfaction results



Regarding the satisfaction, we used the SUS questionnaire (Brooke 1986) to evaluate it. The System Usability Scale (SUS) approach is a simple, ten-item scale giving a global view of the experience proven by the user during the testing execution. The ten questions are standard and appears in a predefined order that alternates positive and negative question to mitigate the effects of undecided testers. A score is assigned to each answer, and a scoring algorithm is applied to obtain the overall usability value within the range of 0–100% (where 0% means "hard to use" while 100% means "easy to use"). Figure 5.7 shows the results we obtained to evaluate the user satisfaction about the partial migration experience. During this test the average SUS score obtained is 75% with a standard deviation of 11%, which is a very good result (especially if we consider the standard deviation associated to it).

5.6 Technical Migration Evaluation

Apart from usability evaluation, some technical evaluations have been also carried out in order to understand the performance of the migration platform. In particular, a technical analysis was conducted in order to gain data on the performance of the

overall migration platform and, down to a finer granularity, the performance of the various modules constituting the migration platform (e.g., the Proxy, the Semantic Redesign module, the Generator, etc.), in order to evaluate their impact on the overall performance. In order to do this, not only the times for migration were recorded for different web pages, but also the characteristics of the web pages themselves were analysed in order to understand to what extent some characteristics affect the different sub-modules.

For instance, for the various modules of the migration platform the following parameters have been considered: number and size of CSS files, number and size of javascript files associated with the web page, number of images, size of the considered web page, number of nested nodes in the tree corresponding to the considered page, number of links existing within the page etc. Of course, some parameters are more relevant for some submodules, while other parameters are relevant for other submodules. For instance, the number of images included in the page is a parameter that is especially relevant for the Semantic Redesign module: indeed, for every image included in the page, the Semantic Redesign module has to evaluate if a resizing process is needed (if the size of the image is beyond a certain threshold), therefore, the time that the Semantic Redesign will need for this process will be proportional to this number. In addition, the number of images that have been actually resized by the Semantic Redesign is another parameter that has an impact on the performance of this module, since for each of such images the module has to perform the resizing action.

For other submodules, the characteristics analysed might change. For instance, for the Proxy module, the time requested by this module on a page will be affected by the number of links that belong to the page, since the Proxy module has to change every link included in the page in such a way that all the links will pass through the Proxy.

5.7 Considerations and Open Issues

There are some aspects that are not currently managed by the migration platform and others that highlighted the need of further improvements, especially after evaluation tests that were performed within the OPEN Project.

One point is that the reverse engineering part is able to build corresponding logical descriptions from (X)HTML and JavaScript implementations. However, if the Web application contains Flash or Java applets, then this module is not able to analyse their code and correspondingly generate the logical description. Thus, in this case, such pieces of code are either replaced with alternative content provided by the application developers (such as images) or they are passed as they are to the target device, if it is able to execute them.

Other aspects are connected with the preservation of the javaScript state of the web page. In some cases it revealed to be a bit hard to be managed. One example of this is the use of lexical closures (that can be found in some javaScript functions

existing in a web page). Closures are a mechanism that is supported by javascript language and it is often exploited to deliberately hide the state of some variables (sometimes for security reasons). Due to the specificities of this mechanism, it was difficult to preserve the state enclosed in this kind of constructs.

Moreover, one aspect that was highlighted as needing improvement in the migration platform was the meaningfulness of the labels of the links that are automatically added by the Semantic Redesign module when a web page splitting is carried out. For the moment, the labels for such links are built by directly getting them from the element names/identificators that are included in the original page for those elements. More intelligent techniques (e.g. like the analysis of additional data like for instance the words contained in the referring page) should aim to identify more meaningful names for such links in order to enable the user to better understand which content the linked page will actually contain.

Other aspects that deserve further attention in the near future are those related to privacy and security in such migration environments.

5.8 Conclusions

In this chapter, we presented our solution for supporting both total and partial migration of Web applications from desktop to mobile systems. Examples of web application were described to show how we support the possibility of totally migrating a page, or migrating a subset of the elements displayed and running in the source device. In the latter case users can directly select what parts should be migrated from the original Web page in an easy and intuitive manner. We also presented the results that we gained during an user evaluation where specific aspects were assessed (usability, effectiveness, efficiency, …).

References

Brooke, J.: SUS—A quick and dirty usability scale. http://usabilitynet.net/trump/documents/Suschapt.doc (1986)

Edwards, W.K., Newman, M.W., Sedivy, J.Z., Smith, T.S.: Experiences with recombinant computing: exploring ad hoc interoperability in evolving digital networks. ACM Trans. Comput. Hum. Interact. 16(1), 1–44, Article 3 (2009)

Eisenstein, J., Vanderdonckt, J., Puerta, A.R.: Applying model-based techniques to the development of UIs for mobile computers. In: Lester, J. (ed.) Proceedings of 5th ACM International Conference on Intelligent User Interfaces IUI 2001, pp. 69–76 (14–17 Jan 2001), ACM Press, New York (2001)

Melchior, J., Grolaux, D., Vanderdonckt, J., Van Roy, P.: A toolkit for peer-to-peer distributed user interfaces: Concepts, implementation, and applications, EICS'09, pp. 69–78, Pittsburgh, 15–17 July 2009

Paternò, F., Santoro, C., Scorcia, A.: Ambient intelligence for supporting task continuity across multiple devices and implementation languages. Comput. J. 53(8), 1210–1228 (2009a)

Paternò, F., Santoro, C., Spano, L.D.: MARIA: A universal language for service-oriented applications in ubiquitous environment. ACM Trans. Comput. Hum. Interact. 16(4), 19:1–19:30 Nov (2009b)

Pierce, J.S., Nichols, J.: An infrastructure for extending applications' user experiences across multiple personal devices. In: Proceedings of UIST 2008, pp. 101–110, Monterey, 19–22 Oct 2008