OPEN Project

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

Title of Document:	Engineered infrastructure for migratory interfaces
Editor(s):	G.Ghiani, F.Paternò, C.Santoro, A.Scorcia
Affiliation(s):	CNR-ISTI
Contributor(s):	
Affiliation(s):	
Date of Document:	November 2009
OPEN Document:	D2.5
Distribution:	EU
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)

ABSTRACT

This deliverable describes the software architecture of the updated prototype under implementation for supporting user interface migration. It is a middleware aiming to support automatically the main functionalities, adaptation and state persistence, across multiple devices with various interaction resources.

TABLE OF CONTENTS

1. INTRODUCTION

This deliverable is the updated version of the software prototype that was presented in D2.1. This new version of the prototype shares the basic architecture of the previous one but provides more refined versions of the various modules and adds the possibility to support partial migration. Partial migration is moving only a portion of the interactive application (namely: some components) to another device in order to better exploit its interactive resources. This new version was also presented at the review meeting at M18 (in one of the integrated prototypes) applied to the Arcadia Social Game and using the Vodafone Orchestrator.

Our approach aims to provide a general solution for Web applications implemented using (X)HTML, CSS, and Javascripts. 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. Thus, 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.

Our approach aims to provide a general solution for Web applications implemented using (X)HTML, CSS, and Javascripts. 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. Thus, 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.

In this deliverable we present a solution supporting partial migration, its main characteristics, the architecture of the migration platform supporting it, and also provide an example of a partial migration for a Web application in the game domain, in order to show its use and potentialities.

2. BACKGROUND

The starting point for the prototype presented in this deliverable was the solution presented in D2.1, which supports migration of only entire user interfaces (total migration) without providing any possibility to migrate only parts of them.

In that deliverable an architecture based on a number of modules was adopted supporting dynamic reverse and forward engineering. The module identified were:

- the **Reverse Engineering** module builds the logical description of the source page considered;
- the **Semantic Redesign** transforms the source logical concrete description into another one tailored for the target platform;
- the **State Mapper** associates the state of the cur-rent 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 in charge of serving as an intermediate layer capturing the interactions between the user browser and the original web site;
- The **Migration Orchestrator** 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 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). Another aspect supported by the semantic redesign phase in the case of desktop-to-mobile transformation is the possibility of splitting the presentations into smaller ones that are more sustainable for the resources of the target device.

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 of the UI state.

3. THE ARCHITECTURE OF THE NEW SOLUTION

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 language that 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). Thus, we want to make this logical structure accessible to the users so that they can interactively select what parts they want to migrate and to which device.

Our approach assumes the existence of a Web desktop version of the application, which does not seem a particular limitation given the wide availability of this type of application. This version provides the basic content for automatically creating the versions adapted for the other platforms.

The software architecture of the new prototype supporting user interface partial migration is illustrated in Figure 1. All the devices that can be involved in the migration are detected by the device discovery protocol, which is continuously active to detect changes in device availability. Such devices should be able to run Web applications, therefore they should be web browser-enhanced. In addition, each device involved in migration should run a thin software (the migration client), which allows users to know what devices are available, select the one that should be the migration target, and trigger the migration. Currently, this software is implemented in C#.

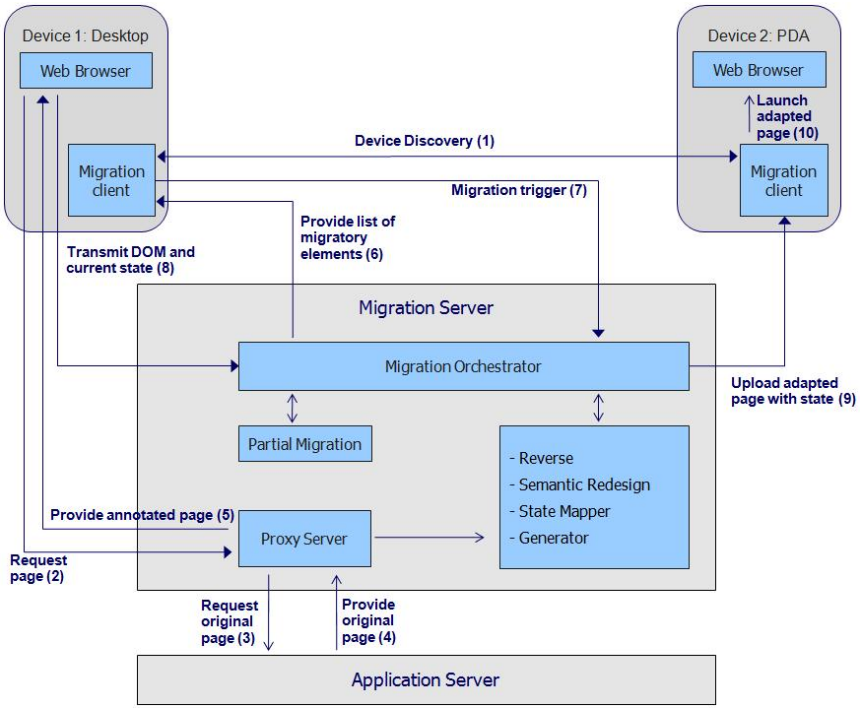

Figure 1 The architecture of our prototype for partial migration

The device discovery support is based on the exchange of presence information among devices. The communication is enabled by sockets, which can be accessed through various implementation languages. The device discovery has to notify (see the Device Discovery phase, labelled (1) in Figure 1) the presence of the associated device to a known multicast group. The group consists of all those devices that are running the device discovery. Such devices are available and could be involved in the migration. In order to carry out this notification, the Device Discovery module in the Migration Client multicasts "hello" messages using UDP/IP protocol. Each device compiles the list of available devices according to the received "hello" messages.

Afterwards, the user interacts through the Web Browser on the current source device to request a page (2) from the concerned application server. The request is captured by the migration server (which includes a proxy server), which asks for the original page from the application server (3). The latter provides the migration server with the original requested page (4). The migration server annotates the page and returns it to the Web Browser on the source device (5). It is worth pointing out that this process is repeated for every page the client device requires access to.

When the user selects the migration options on the client (see Figure 4, left), the migration server Orchestrator communicates with the Reverse module. This is done in order to call the Reverse module functionalities for the analysis of the page. The Reverse produces the Concrete User Interface (CUI) description associated to the current page and passes it on to the Orchestrator. One specific functionality of the Partial Migration module is to parse the CUI provided by the Orchestrator. The result is a list of user interface components according to the hierarchical relations among them. Such list, organised according to the hierarchy of the

interactor compositions of the main presentation, is sent by the Orchestrator to the source device (6), in order to let the migration client display the tree-like view of the page structure. The user is then able to specify which page components to migrate by selecting them in the migration options form (see Figure 4, right).

Automatically generating the list of Web page components names and sending them to the

Figure 2 A screenshot of the Web gaming application considered in the example highlighting the groupings

migration client (in the background, since the user is interacting with the browser), allows masking the computation and communication latency. When the user pops-up the migration client, s/he will already have the tree view of the page structure in the migration options, thus speeding up the selection of the components to migrate.

Let us consider what happens when migration is triggered (7) after the list is provided. If the user triggers a migration request when a subset of the components is selected on the list, then the migration is considered to be partial. In this case, the 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 state resulting from the user interactions (elements selected, text entered, …) is automatically transmitted through an Ajax script to the migration server (8). Such state transmission is supported by the annotation, provided by the migration server, which includes in the page some scripts that take the Web page state through its DOM and provide it on request. The state information also contains the last page element that had the input focus in the source version. The version for the target device is generated with the same focus, thus allowing the users to continue the interaction from precisely the same point they left off.

4. AN APPLICATION EXAMPLE OF PARTIAL WEB MIGRATION

In this section an example of partial Web migration is presented. The example considered deals with a gaming application that includes different parts: a chat, a betting part, an IPTV, a racing game, etc. (http://www.arcadiadesign.it/SocialGameIntegrated/), and also involves interaction among multiple users (social game). An example user interface can be seen in Figure 2: 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 goal of the game is to finish a lap in the shortest time. In order to be able to interact with the various elements of the social game, users need to login with password in the fields on the top right of the page. As soon as a user connects, a car is assigned to him. The arrow keys can be used to control the car (up for accelerating, left and right to steer, down to brake). In the current scenario various users are already connected.

At a certain moment the user decides to partially migrate the application to a mobile device, since s/he has to go out.

Figure 3 shows the tree that summarizes the logical structure of the CUI of the current Web page created by the reverse engineering module. Note that the colours of the nodes in Figure 3 refer to the colours of the squares that highlight the sections in Figure 2. *Grouping* and *Relation* nodes are specific types of interactor compositions. For example, the HTML login form (pink nodes) is identified by a *relation* containing two input interactors and the submit image.

The HTML tag div containing the Betting area composed of the table, section and form at the bottom left is identified by the purple nodes in Figure 3 (which refer to the first yellow grouping node in Figure 2).

The server carries out the migration of the selected components to the new device (PDA). In this case (see Figure 4) the partial migration is applied to the chatting, betting and gaming parts.

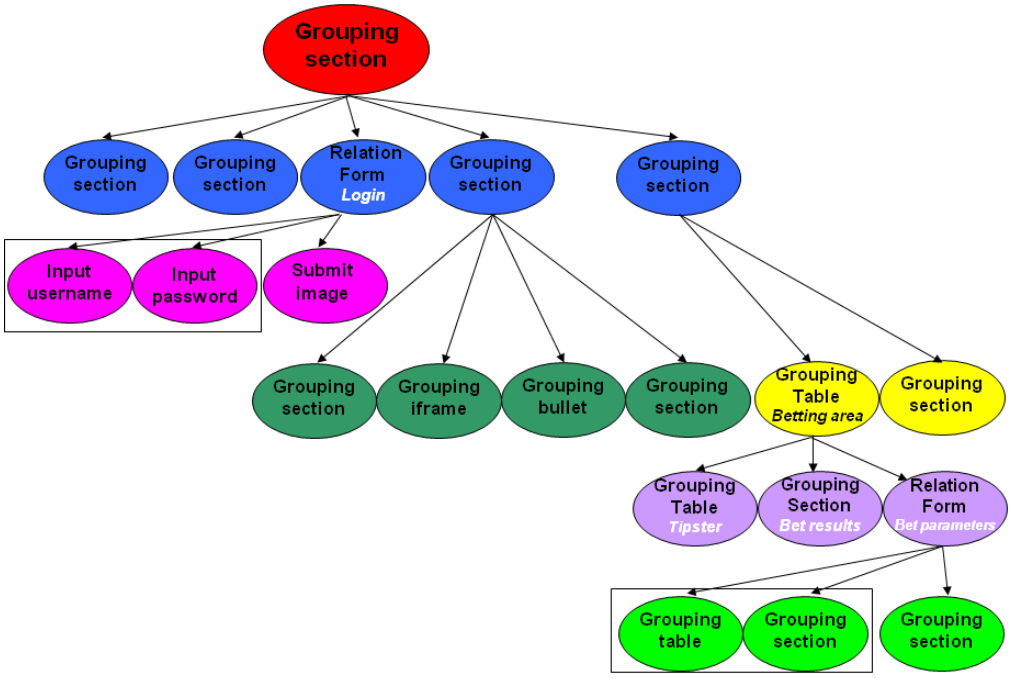

Figure 3 The tree-structure of the Web interface components resulting from the Reverse

Figure 4 shows the migration client interface for selecting both the target device and which components to migrate. In particular, for a desktop-to-pda migration, it is possible to select a PDA as the target device, as in Figure 4. More specifically, on the left side you can see that four devices are available in the example scenario. Depending on the device currently selected further information is presented in the bottom part of the user interface.

As soon as the user has selected a particular device s/he can trigger the migration. In the case of partial migration, an additional window shows the list of components that can be migrated. The list is displayed in a tree-like way, reflecting the structure of the original page: groups of components and/or single components can be selected for migration (see Figure 4, right). The representation of the user interface components is a simplified version of the logical structure of the application interface in order to facilitate the choice by the users. For this purpose our platform does not show the groups that correspond to elements without associated functionality. In the example considered some groups just correspond to decorative images and they do not appear in the tree-like list of Figure 4. The names that appear in the selectable list are automatically generated from the name attributes used in the corresponding tags at implementation level.

It is important to note that the tool supports also a different solution than that in which the list of components is obtained with an automatic procedure through the use of logical user interface descriptions. Indeed, in the prototype that we showed at the M18 review, the list of components was obtained by the Migration Orchestration implemented by Vodafone, which sent to the CNR tool the list of components previously registered by the Arcadia application. This is an important

demonstration of how this prototype has been integrated with the other components of the OPEN platform.

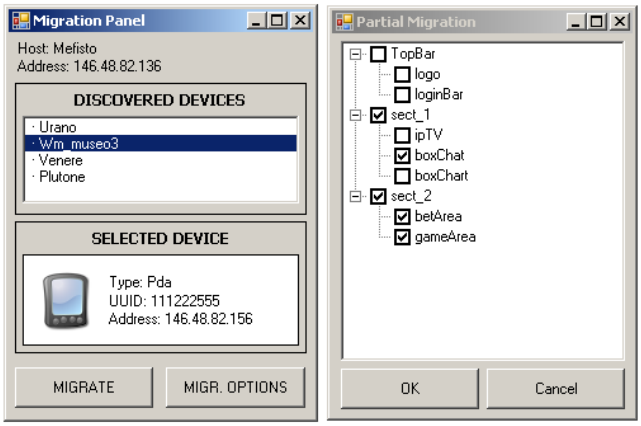

Figure 4 The window displayed to the user for selecting target device (left part) and the components to migrate (right part)

Then, once the user has selected the components that s/he wants to migrate (which, in this case, are three), a new presentation is generated by the migration support. This is done by creating a new presentation consisting of a grouping composition of the selected components at the concrete description level.

Such a newly created concrete presentation has to be analysed in order to be semantically redesigned and adapted for the new target device. In this case, since the target device is a mobile one, each of the three selected components (bet, chat and game positions) will be displayed on a separate page.

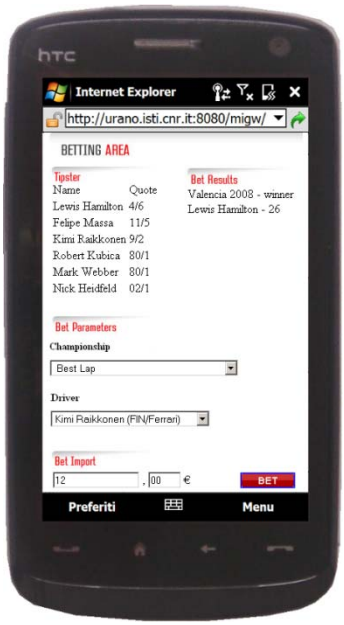

Figure 5 The Betting area visualized on the PDA after partial migration

Figure 6 The Game position part after partial migration

Figures 5, 6, and 7 show the various presentations that the user can see after migration. In the adaptation process some interactors can be replaced by others that preserve the same semantics but are better adapted to the current device. For example, in Figure 5 the Championship component is a pull-down menu while it was a radio button on the desktop version.

Figure 7 Chatting part

5. CONCLUSIONS

This deliverable describes the current state of the prototype supporting the migration of the user interface software.