Towards Intelligent Migration of User Interfaces

Giuseppe Ghiani[1], Jussi Polet[2], and Ville Antila[2]

[1] ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy
giuseppe.ghiani@isti.cnr.it
[2] VTT Technical Research Centre of Finland, Kaitoväylä 1, 90571 Oulu, Finland
{jussi.polet,ville.antila}@vtt.fi

Abstract. The multitude of interactive devices we use daily has steadily increased since the advent of personal computers. Also, the spreading ubiquitous computing leaves the users with increasing number of device ensembles. There is often a need to change the device in use, which requires moving applications, data, user interfaces or parts of them to other device(s) and back. This is the case, when something is manipulated on one device and a need to migrate it towards another device exists as well. In order to study the problems of such migration functionality, we defined, designed and implemented a proof-of-concept prototype for automatic context-aware migration of Web applications between devices. The prototype was evaluated in three distinct modes (manual, assisted and automatic) with a user study to collect technical data and user feedback. The results highlighted interesting correlations between the system behaviour and user ratings, and statistically relevant differences on how users perceived the proposed modes of the system.

Keywords: Context-Awareness, Device Ensembles, Migration, Migratory User Interfaces, Multi-device Environments.

1 Introduction

People are increasingly accessing applications and services through diverse types of interactive devices, depending on their current location and needs. The users are facing different kinds of device ensembles at home, work as well as on the move. In addition to providing ways and means for moving and synchronising information, whether that is applications, plain data or user interfaces (UI) between all these devices, it is also challenging to *"keep your digital identity in sync"* [20] – contact lists, media libraries, *Facebook* credentials and so forth need to be in sync and accessible on several different devices. The task of keeping each device up to date with the latest applications and data can be sometimes tedious. Therefore, data and applications are increasingly in the cloud, keeping everything in one place, accessible by all devices.

Also, Web applications are a typical example of applications accessed in different places and with heterogeneous devices, as the actual application and user data is always located on the backend server(s) (i.e. cloud). Documents, music, photos etc. are all accessible from the application front-end that is run with each device's Web browser (or with a dedicated native application that displays the Web content).

F. Daniel, G.A. Papadopoulos, P. Thiran (Eds.): MobiWIS 2013, LNCS 8093, pp. 203–217, 2013.

In this paper, we take a look at the possibilities for migrating Web application front-ends (i.e. the Web page and its state information) between devices. The actual application back-end and data will always be accessed from the original application server. Moreover, in this paper we focus on the context-awareness of such system and the possibilities it could give for enhancing the interaction between the system and the user. Specifically, we take a look at three different ways of interacting with the system: *automatic*, *assisted* and *manual*. Two of these are context-aware: automatic and assisted. With automatic mode, when the system senses a predefined trigger for migration, it performs this action automatically. With assisted mode on the other hand, when sensing a trigger for migration, the system notifies the user of this possibility, leaving the final decision to migrate to the user.

The prototype environment for context-aware migration is an extension of a previously developed multi-device and multi-user Web-based *Migration Platform*. Earlier studies with that platform [11] highlighted a need for intelligent, context-aware and more automatic migration of user interfaces. To address these needs we have implemented and evaluated a context-aware system for Web application migration. The solution presented in this paper is especially targeted towards the needs of task continuity and continuous movement, characterising today's technological multi-device environments. In addition, the added context-awareness eases migration triggering and thus minimises user effort. Our early results indicate that adding context-awareness into the migration process is a notable benefit for the user. We have also discovered important correlations between the technical qualities of the system to the perceived usability and user experience.

In the next sections, we will first present some related research on multi-device interaction and interaction techniques and discuss the benefits of context-awareness in multi-device environments. Then, we describe the architecture and implementation of our prototype system and the user study setting with the test scenarios. After that, we present our evaluation methodology, analyse and discuss the user study results and draw some final remarks useful to better identify the need for such a concept and system in the near future.

2 Related Work

Users' behaviour towards the usage of multiple kinds of devices has recently been considered in a *Google* research report [12], which identifies two major ways of interaction with multiple devices: *sequential* and *simultaneous*. Our approach is more sequential, though by migrating only part(s) of the Web application to a target device by keeping them active also into the source device, even simultaneous workflow can be achieved.

Multi-device interaction has also been investigated in the past, focusing on techniques people use to access multiple devices: *Dearman and Pierce* [7] found that one of the main challenges is how to support *seamless device change*. This is the core problem that we are aiming to tackle by providing an integrated, context-aware functionality for device changes.

One such solution for sharing information and session during device changes in multi-device environments is *Pick-and-Drop* [17], which was a first interesting solution to support the dynamic graphical selection of application elements in one device and easily move them to another device. However, that approach was mainly limited to move data across devices, while people would like also to move entire interactive applications. *Berry et al.* [2] deal with view sharing in cooperative work environments, and tackle the privacy issues of a presenter which wants to show/hide parts of the view according to the type of audience. The solution focuses on collaborative tasks, while in this paper we focus on single-user scenarios in which our platform can be useful.

Support for migratory UIs aim to improve user access to information through multiple devices. In this regard, *Wäljas et al.* [24] have also studied how people access various multi-device applications during a few weeks, and have proposed an initial framework for analysing cross-platform service user experience (UX), which relates to three main aspects: *composition*, *continuity* (fluency of the content and task migration), and *consistency*. Our approach pays attention to all these aspects by allowing the user to either migrate the whole UI or just parts of it (composition) between devices, preserving the continuity of the UX by allowing seamless device changes and aiming for a consistent UX by focusing on the Web application domain. The *eLabBench* [23] relies on an infrastructure for distributing data across laboratory devices and biologists personal computers. The focus is on ubiquitous management of digital data, rather than on UI migration. The approach is thus different from ours, but the final aim is still to support task continuity of nomadic users coping with more than one device in different contexts. Their Fluid Computing middleware [3] was also about multi-device interaction. However, it differs from our Platform both on aims and on implementation: Fluid Computing is aimed at interaction synchronization when multiple users share the same interface, or parts of it, while we do not specifically deal with interface replication.

Deep Shot [5] is a solution for automatic migration of UIs across devices with state preservation. Migration of an interface, or part of it, is triggered by "shooting" it with a mobile device camera. In our platform, migration can be also triggered in a fully automated way without requiring explicit user interaction. Also, the authors of Deep Shot state that this approach is compatible even with native applications, but that existing ones need to be modified in order to support the *"deep shooting/posting"*. Our platform for Web migration instead relies on a proxy that injects all the needed support to the navigated pages, and does not require any modification to existing applications. Also, browser plugins are not needed for enabling migration. The only additional software to enable context-based migration is the *Context Monitor* (which is platform-dependent but compatible with consumer devices). The Context Monitor, described in the following sections, is an extension of an already existing platform for *"pushing/pulling"* of Web applications across devices [11]. Pushing and pulling issues were also investigated in [8], whilst our present work considers only the migration pushing i.e. migrations originated from the device in use. Other aspects of the platform that we consider in this paper have been already tackled: security/privacy is discussed in [10].

The *RELATE system* [9] supports user interaction with devices available in the surrounding. Specific hardware equipping the devices in the environment allows them to discover each other and to determine their relative position. Rather than on specific localization capabilities, in this study we aim to investigate how usability perception varies according to the system behavior.

Sensors embedded in the device or deployed in the environment provide low level data. As indicated in [4], such data can be used by means of evolving situation models to infer high level situations in which the users are involved. Our basic idea matches with the author's one, as we believe that context-awareness would enhance nomadic users' multi device experience.

Enabling technologies for communication, such as *near field communication* (NFC), *radio frequency identification* (RFID) and *Bluetooth*, can be used to sense the co-location of target devices in the proximity of the user [6,18]. Moreover, different characteristics of proximity in interacting in ubiquitous computing environments have been identified [13]. There has been research on techniques that facilitate interaction, taking into account the proximity factors between people and computing devices, such as location (e.g. room or building), position (close to a display), movement (coming towards a display) and orientation (facing towards the display). These factors allow better contextual knowledge about the current situation, so that the system can determine, for instance, whether the public display should facilitate interaction with someone or just display information [1,15].

In this paper, we do not tackle issues of difference in interface rendering due to device diversity, nor possible adaptations of the interface to the destination device (e.g., rearranging of presentation components), which are relevant but would require a dedicated paper. We instead demonstrate the ability to use a simple sensing application as an enabler of context-awareness in a multi-device environment. The system we propose is able to automatically detect some context variables, such as user location, device position and information about the user's current activity (e.g. walking or still). We discuss the possibilities of using this context information to allow the system to perform automatic or semi-automatic (assisted) migration of session from a device to another. This type of automatic trigger can be related to the work on implicit *human-computer interaction* (HCI) driven by the context discussed in [21], in which the system acts proactively on the basis of context information.

3 Context Awareness in Migration Systems

While conducting usability tests for our previous prototypes of multi-device UI migration [11], it was noticed that the UX was highly related to the selection of the migration target device (*target acquisition*) and the UI migration triggering (*process initiation*). By making the system context-aware, these two aspects can be enhanced by the system offering/choosing relevant choices for target devices and by suggesting or initiating migration automatically, as discussed by *Schilit et al.* [19]. The offering/choosing of relevant target devices and suggesting/initiating migration is enabled by the system being aware of the surrounding devices, environment, users and social relations. Thus, the main benefits of making a migration system context-aware are:

- *Automatic target acquisition and selection* – with the system knowing and using information of the environment, a suggestion or decision can be made of the migration target device;
- *Suggested or automatic migration* – with the system knowing the current situation, it can either suggest the migration to a nearby device (assisted migration) or trigger the migration automatically (automatic migration).

4 System Architecture and Implementation

The context-aware migration system is based on four main components; 1) devices, 2) *Migration Client* running on each device's Web browser, 3) Context Monitor running on the phone and 4) Migration Platform Web server.

The migration is managed by the Migration Platform that utilises a proxy server. The *Migration Proxy* annotates existing Web pages in real-time, when a user is browsing the Web via Migration Client. The browsed pages are injected with additional scripts that enable the migration from one device to another on request, while the original functionalities of the Web application are preserved. Code injection performed by the proxy and strategies to manage security are fully discussed in [11] and [10], respectively. Context Monitor allows triggering the migration based on sensed values of selected context parameters and applied custom rules, providing increased intelligence into the migration. The Context Monitor utilises the added external triggering functionality of the Migration Platform.

4.1 General Architecture Description

To enable context-aware migration, the originating (source) device and the destination (target) device must run the Migration Client in their Web browsers. The Migration Client is a simple Web application for logging the environment and for browsing Web pages via the Migration Platform proxy server.

Typically, the source device is already running the Migration Client, since it is used to navigate Web pages via the Migration Proxy. However, the user must ensure that the target device also has the Migration Client running in order to manage the incoming migration request when the context-based migration is triggered. It is a reasonable constraint in our system that the Migration Client needs to be up and running in the devices' Web browsers.

An overview of the system architecture is shown in Fig. 1, where the environment and main communications are depicted. In Fig. 1 (left), Devices A (phone) and B (desktop computer) have the Migration Client running and the phone has Context Monitor running as well. When the user opens a Web page on Device A, the request goes through (1) the proxy server, which relays it (2) to the actual application server. The application server then responds back (3) with the resource to the proxy server. The Migration Proxy annotates the received resource (e.g. a *HyperText Markup Language* (HTML) page) by adding additional *JavaScript* code and sends the annotated resource as response to Device A (4). The injected code enables the page to be subsequently migrated, and does not affect its original layout or content.

Depending on the selected context parameters and its configuration, the Context Monitor is aware of the Device A situation. For example, it knows the movement and position of the Device A, nearby devices and the preferences of the user. When the system senses an opportune moment for migration (e.g. Device A is in the close proximity of the Device B and placed display-up on a flat surface, such as a table), the Context Monitor sends an external migration request (5) to the Migration Platform, in which the source and target device IDs are specified. A more complete architecture description of the basic Migration Platform, i.e. without context awareness, can be found in [11].

The Migration Platform relays this request to the Migration Client of the specified source device (6), which then invokes the previously injected JavaScript code for the browsed page. The process serialises the *document object model* (DOM) of the browsed page by including the state of the interaction (e.g. content of page elements) and sends this serialised data back to the Migration Platform (7), which saves it locally. The Migration Client running on the target device is then notified (8), with the address to access the Web page to be migrated and automatically opens the page on the Web browser of Device B.

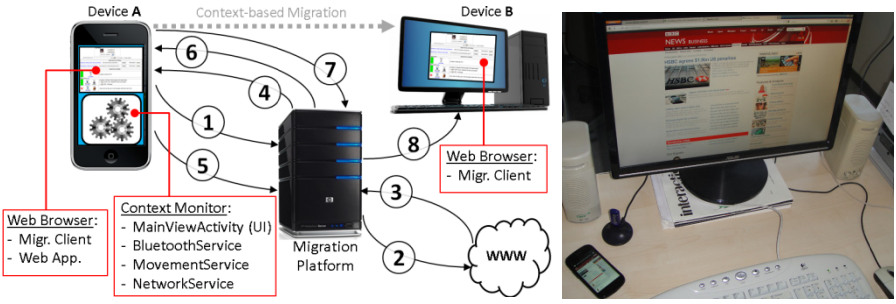

Fig. 1. Overview of the system architecture (left) and typical scenario (right)

It is also to be noted that the Migration Client continuously communicates with the Migration Platform server and provides a unique ID of the devices and user credentials. Therefore, since each instance of the Migration Client is uniquely identified, more than just one potential target device can be running the Migration Client (i.e. be active) in the same environment at the same time.

4.2 Context Monitor Overview

The Context Monitor is a mobile application targeted for *Android* smartphones, which runs on the background and observes the selected context parameters. The context parameters can be e.g. the proximity of other devices (obtained by using the Bluetooth *received signal strength indicator* (RSSI)), the current physical activity of the user (fall detection, still, walking and running activities obtained by using the device accelerometers), the current virtual activity of the user (the currently running applications and currently used applications on the device, call and *short message service*

(SMS) activities etc.), location (*global positioning system* (GPS) coordinates, *wireless local area networks* (WLAN), *base station cell ID* (CID)) and other, such as audio and light etc.

The context parameters can be chosen and configured, so that the user can set when the Web application should be migrated from the smartphone (referred in the following as *phone*) to another device and vice versa. Thus, for example, the user can set the target devices for the migration and also the context-dependent rules, such as *"perform migration from the phone to the office computer when the phone is still after being placed on a table, facing upwards and in the close proximity of the office computer"* and *"perform migration from the office computer to the phone when the phone has been lifted from the table, is moving, and the office computer is not in close proximity"*.

It needs to be noted though, that the Context Monitor is a very limited mobile application developed for the purposes of this study and for the evaluation of the concept of context-aware UI migration. Therefore, the application does not aim to tackle the problems related to the wide field of activity recognition, as those questions are out of the scope of this paper.

5 Evaluation

In this section we present the evaluation of our prototype system to explore how proactive and context-aware UI migration is perceived from user's point-of-view and which technical requirements are needed for. The factors that we were interested in were related to the existence of relevant preferences for a specific migration mode (manual, assisted and automatic) and of possible relationships between individual user factors, interaction behaviour and declared ratings for the tested system.

We recruited a total of 24 persons and as heterogeneous trial group as possible, consisting of both computer science professionals and of people who only use computers and smartphones for basic office worker's purposes. We recruited the trial participants by sending an email within our organisations, asking people matching our recruitment criteria to join study. We personally recruited as many people as possible from within our social working circles matching the recruitment criteria.

We scheduled the user tests for a time span of 1.5 weeks, approximately three to four participants per day, maximum 60 minutes each. Users tested the Context Monitor and the Migration Platform with the simple use case scenarios and answered to the questionnaire formed of the aforementioned interesting factors, such as preferences for specific migration mode and relationships between individual user factors and system behaviour.

The test tasks consisted of Web page migration from a Android phone to a PC and vice-versa using three different interaction modes (manual, assisted and automatic). The diverse available modes to operate an interactive system are discussed in [22], where the authors tackle the prevention of *"mode errors"*, i.e. situations in which users forget the mode in use. The authors discuss how to provide feedback to the users to remind them how the system is operating. It is worth pointing out that in our

test setting the system did not provide any explicit mode-dependent feedback, because the users were provided with instructions for each task and informed about the current mode before starting the interaction. In addition to a background questionnaire, we had the participants fill an in-between questionnaire after each interaction mode.

We also recorded the laboratory studies on video to count the focus shifts between the devices and to catch any freeform comments and suggestions during the laboratory tests. By focus shifts, we loosely mean the *macro attention shifts* described by *Holleis et al.* [14].

5.1 Test Scenarios

The current developed versions of the Migration Platform and Context Monitor were used in the laboratory test. The contextual factors we focused on were the phone stability and the Bluetooth presence/proximity information. The two test scenarios are presented in the following.

Scenario 1: The user is approaching the office while browsing some Web page on her Android phone. As soon as s/he gets to her desk and puts the phone on the table (see Fig. 1 (right)), the browsed Web page is migrated, depending of the test mode, either automatically or assisted, from the phone to the computer.

Scenario 2: The user is accessing some Web application on the desktop computer and suddenly needs to leave his office for a meeting. S/he only takes the phone with her and walks away. The Web application is then migrated, either automatically or after user confirmation (depending on whether the selected mode is automatic or assisted), from the computer to the phone. This way, the workflow is not interrupted by the transition from the office to the meeting room.

5.2 Data Analysis

In this section we provide analysis of the data gathered during the user test. The data sources are the questionnaires the participants filled in and the technical data gathered from the Migration Platform during the laboratory tests.

Demography and Background Information
The 24 test participants were in average 32 years old (median 31 years), youngest participant being 26 years old and the oldest participant 44 years old. Only four female were involved in the study. Most of the people were *Finnish* (eleven) and *Italian* (eleven) by nationality, but also one *German* and one *Portuguese* participated. Education level was high: six *Bachelors*, thirteen *Masters* and five *Doctors*. Two participants were partially impaired (one in wheelchair and one without one hand). Approximately half of the participants were “tech-savvy” (developers, early adopters and highly interested in technology) and the rest had “normal technical skills”.

System Functionality
We logged all relevant event times at the Migration Platform to count the latencies of the system. Latency directly affects UX because it forces the user to wait for the

migration to be carried out. Thus, latency has been considered as indicator of technical quality of the system under investigation and studied in correlation with other variables, as discussed later. We also recorded the user tests on video in order to count the users' focus shifts between the devices during each migration mode and to catch comments. To minimise any learning effects, we shuffled the migration modes for each test participant. By permuting the manual, assisted and automatic modes, we thus obtained six different migration mode orders.

The hypothesis for the focus shifts was that when using the manual mode for migration, there would be more focus shifts between the devices, as the users would check between the devices, and that there would be less focus shifts when moving towards more automatic migration.

In order to get the latencies, we recorded all the relevant events at the Migration Platform server, such as: mobile device placed on the table, migration to PC triggered on the mobile device, Web page opened on PC, mobile device picked at hand, migration to mobile device triggered on the PC and "Web page opened on mobile device.

For the manual mode, we calculated the latencies between the user triggering migration on the mobile device to when the Web page is opened on the computer (A) as well as between the user triggering migration on the computer to when the Web page is opened on the mobile device (B). Focussing on the manual mode, the higher latency of the PC to Mobile migration can be explained by the time taken by the mobile browser to render the migrated page.

For the assisted and automatic modes, we calculated the latencies between the mobile device being placed on the table to when the Web page is opened on the computer (C and E) as well as between the mobile device being picked at hand to when the Web page is opened on the mobile device (D and F). These latencies are depicted respectively in Fig. 2 (left). As it can be seen, in the assisted and automatic modes there was significantly more latency. This is due to the context detection algorithms that take a few seconds to detect the events of *"arrived to the office and placed on the table"* / *"lifted from the table and left the office"*. Indeed, the algorithm continuously monitors acceleration/tilting and needs some time in order to avoid false positives (unwanted migrations). Therefore, the latencies were also higher for the PC to mobile cases, because the context detection algorithms take more time to notice the event of *"leaving the office"* than to detect the *"arrived to the office"* –event.

In case of manual trigger, instead, the time behaviour is mainly due to: a) latency of the page elaboration (client- and server-side) and b) network delays.

Client-side, the page elaboration consists of the serialisation of the page, i.e. the creation of the *eXtensible Markup Language* (XML) string document from the DOM on the browser. Client/side serialisation is done by a JavaScript procedure. Server-side, the elaboration consists of: parsing the XML string to create a document object, filling the document with additional information (e.g. user interaction state) and serialising it into a text/HTML file that will be accessed by the target device browser.

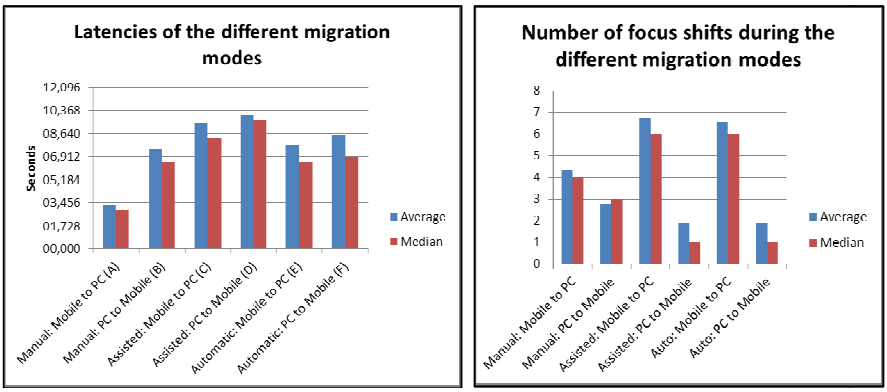

Fig. 2. Latencies (left) and focus shifts (right) of the different migration modes

We argue that, because of the system latencies, assisted and automatic migration led to the highest amount of focus shifts (see Fig. 2). Therefore, we had to claim our initial hypothesis (less focus shifts when moving towards more automatic migration) false for this study. As a future work, the assisted and automatic mode latencies will be reduced as much as possible. The system, as pointed out by many of the test participants, should also give some sort of hint of the migration being in progress, for example, by using a small progress bar over the screen.

Nielsen [16] discusses the acceptable approximate latencies/response times for any application in regards to usability. If the system response time is more than 1 second, feedback during the delay is important and required or otherwise the users do not know whether the system is actually doing anything. With our current prototype the overall response time of the system is more than 1s, therefore it is imperative to give a progress indicator(s) during the migrations. Furthermore, considering the context-detection algorithms and a) the latency of the page elaboration (client- and server-side) and b) the network delays, it will be very challenging to develop the system to perform in the time range of less than a 1 second. However, the participants commented on the free-form sections of the questionnaires that the system already performed surprisingly well, which give us initial direction that a migration latency of few seconds is acceptable, as long as hint(s) of the system progress is displayed. In general, we also gained good insight into the trade-off between the system latency and the detection quality. The participants commented that a slight latency is tolerable, if the context detection does not produce false negatives/positives and system progress is visualised. Half of the participants faced at least one false positive (i.e. an unwanted migration) or negative (i.e. a migration that did not occur when needed). Among such failures, 33 were false negatives and 5 false positives. Most issues affected the mobile-to-PC migration, and thus the “arriving to the office” event was not detected as well as the “leaving the office” event.

Statistical Analysis

Statistical tests have been applied to the data gathered during the trials. The aim was to discover possible correlations between several aspects (i.e. variables), such as

between the number of video-detected focus shifts of the user and her/his rating. We ran *Pearson correlation tests* over the system latency and focus shifts in relation to the average usability, system speed (usability), suitability of the interaction mode and the willingness to use the system (and the selected mode) in the future. The Pearson correlation tests are summarised in Table 1.

Table 1. Summary of the Pearson correlation tests

VARIABLES		MIGRATION TASK					
		Manual		Assisted		Automatic	
		Mobile - PC	PC - Mobile	Mobile-PC	PC-Mobile	Mobile-PC	PC-Mobile
Latency, shifts	Corr.	0,151	-0,042	-0,052	-0,108	0,125	-0,116
	Sig.	0,48	0,845	0,809	0,617	0,561	0,588
Shifts, average usability	Corr.	-0,228	-0,344	-0,066	0,016	0,215	-0,361
	Sig.	0,285	0,1	0,76	0,941	0,313	0,083
Latency, average usability	Corr.	-0,255	0,147	-0,212	-0,013	-0,037	0,019
	Sig.	0,228	0,492	0,32	0,953	0,866	0,931
Shifts, speed usability	Corr.	-0,187	-0,256	-0,054	0,196	-0,041	-0,168
	Sig.	0,383	0,228	0,802	0,359	0,851	0,433
Latency, speed usability	Corr.	-0,351	-0,083	-0,216	0,133	-0,54	0,231
	Sig.	0,092	0,699	0,311	0,535	0,006	0,277
Shifts, suitability	Corr.	0,321	-0,131	0	-0,093	0,107	-0,03
	Sig.	0,126	0,541	1	0,665	0,618	0,891
Latency, suitability	Corr.	-0,473	0,075	0,187	0,079	0,146	-0,201
	Sig.	0,02	0,729	0,381	0,714	0,496	0,347
Shifts, willingness	Corr.	-0,132	-0,391	-0,169	-0,075	-0,114	-0,411
	Sig.	0,539	0,059	0,431	0,726	0,596	0,046
Latency, willingness	Corr.	0,01	0,189	0,056	-0,179	0,081	0,258
	Sig.	0,961	0,376	0,795	0,402	0,707	0,223

The following variables have been considered:

- *Latency:* time needed to perform migration, as previously described in subsection System functionality;
- *Shifts:* number of focus shifts between devices. A focus shift occurs anytime the device the user is gazing changes (i.e. from PC to phone or from phone to PC);
- *Average usability:* mean value among the following usability-related ratings of the user: task easiness, system behaviour understandability, system interaction suitability, speed, reliability, willingness to use the system in the future;
- *Speed usability, Suitability and Willingness:* These refer to system speed, system interaction suitability and willingness to use the system in the future, respectively.

The correlation factor, labelled as *Corr.*, indicates whether the correlation is *moderate* ($0.3 < |\text{Corr.}| \leq 1$), *weak* ($0 < |\text{Corr.}| \leq 0.3$) or *null* ($|\text{Corr.}| = 0$, i.e. variables are independent).

The correlation significance is indicated by *Sig.:* correlation can be *highly significant* ($0 \leq \text{Sig.} < 0.01$) or *statistically significant* ($0.01 \leq \text{Sig.} < 0.05$).

It is worth pointing out that the aim of these tests is mainly exploratory, because they are not actually devoted to accept/reject any predefined hypothesis. Therefore, even if multiple comparison tests involving the same variables were performed, no correction has been applied to the significance intervals.

Such tests show the following statistically interesting aspects:

1. There is highly significant (negative) correlation between latency and system speed usability when using automatic mode from mobile to PC;
2. A statistically significant (negative) correlation exists between latency and suitability when manually migrating from mobile to PC;
3. A statistically significant (negative) correlation for the automatic mode was found between focus shifts and willingness to use the system in the PC to mobile task;
4. A moderate (negative) correlation, although not statistically relevant, was found between latency and speed usability when manually migrating from mobile to PC.

Given these findings, we can argue that they support our notions on how the system was perceived during the tests. Automatic migration took a longer time (due to the context-detection algorithms delay) when migrating from mobile to PC and clearly had the participants question themselves about system usability. Also, when using the manual mode in migrating from mobile to PC, the latency reduced the interaction suitability and had the participants commenting on the manual mode usability.

It is also worthwhile to mention that somewhat surprisingly the results neither show correlation between latency and focus shifts, nor between these two variables and the average usability. This lack of correlation is indicated by the low correlation factor (weak, in most cases) as well as by the significance value, which is much bigger than 0.05. We could argue that the system already performs "good enough" to provide such context-aware migration functionality. As mentioned before, this also shows in the participants' free-form comments that, overall, they already found the system to be reasonably good.

We also ran statistical difference tests on the ratings (*Likert* scale 1-5) the participants gave during the laboratory tests and the system latencies and found out several significant differences. The null hypothesis on the tests was H0: *"The modes perform similarly"* (i.e. the means of the groups are the same).

The results show that we have to reject our null hypothesis (using the conventional 0.01 and 0.05 alphas) on the following cases of the system functionality assessment (numbers in round brackets indicate the p-value):

a) Task easiness, assisted vs. manual, highly significant (0,0001)
b) Task easiness, auto vs. manual, highly significant (0,0008)
c) Interaction suitability, assisted vs. manual, statistically significant (0,0233)
d) Interaction suitability, auto vs. manual, highly significant (0,0059)
e) Willingness to use in the future, assisted vs. manual, highly significant (0,0093)
f) Willingness to use in the future, auto vs. manual, statistically significant (0,0281)
g) Latency, mobile-to-PC, assisted vs. manual, highly significant (< 0,0001)
h) Latency, PC-to-mobile, assisted vs. manual, statistically significant (0,0284)
i) Latency, mobile-to-PC, auto vs. manual, highly significant (< 0,0001)

Such statistics confirm a relevant preference towards automatic and assisted modes (with respect to the manual mode).

These statistics are related to confirmatory data analysis, as they are devoted to reject the null hypothesis. *Bonferroni Correction* has thus been applied with significance level (α/2), aiming to reduce possible family-wise errors. Some variables (i.e. latency in mobile-to-PC manual mode and task easiness manual mode) were indeed involved in two comparisons. The correction lowers the alphas from 0.01 and 0.05 to 0.005 and 0.025, respectively. If such corrected alphas are considered, then some downgrades in the statistical significance occur: d) and e) are re-classified from highly significant to statistically significant, while f) and h) change from statistically significant to insignificant. Nevertheless, even after applying Bonferroni correction, most comparisons are still statistically meaningful. For instance, the e) corrected comparison between willingness to use assisted and manual modes, is also significant, which is interesting as it suggests a preference of the users towards the assisted mode.

Based on these results, we can argue that the assisted and automatic modes were easier to use in the test and suited the given interaction situation better. The participants were also keener to use to the assisted and automatic modes in the future vs. using the manual mode. However, it is worth mentioning that, given further development to system by making the manual mode more easier, the users commented that the manual mode could be useful in some cases as well (e.g. in public spaces when definite explicit control is more desirable).

6 Conclusion and Future Development

We have presented and studied an integrated solution for automatic (implicit) and assisted (suggested) context-aware migration of Web applications, by comparing it with a manual (explicit) one.

The focus has been put on relationships between technically gathered data and users' feedback. We discovered interesting correlations between, for instance, system response time and subjective aspects such as suitability and perceived reactivity of the system. Statistically relevant preferences for automatic and assisted migration modalities with respect to the conventional manual triggering have also been highlighted. Based on the Pearson correlation tests, we claim that 1) even though there was latency in the assisted and automatic modes, the average usability of context-aware migration was still perceived as quite good by the participants and 2) for further development more visual cues of the system progress need to be given. Also, based on the statistical difference tests on the users' ratings, we can argue that the assisted and automatic modes performed better and that the users would prefer those over explicit migration.

According to the study results, we can argue that migratory functionalities for UI and data across different kinds of devices are judged as promising by the users. The migration mode (manual, assisted or automatic) is highly dependent on the task at hand and also impacts on how users might perceive technical parameters such as system latency. We may use an incremental strategy to reduce latency: rather than serializing the full document on the source device and moving it to the target, a document copy could be kept state-persistent in the Migration Platform by means of updates sent from the source. Technical aspects still deserve more investigation. However, if context-based migratory functionalities were available and properly tuned, they would therefore likely to be used by the great public.

Acknowledgements. We gratefully acknowledge support from the EU ARTEMIS SMARCOS Project[1].

References

1. Ballendat, T., Marquardt, N., Greenberg, S.: Proxemic Interaction: Designing for a Proximity and Orientation-aware Environment. In: Proc. ITS 2010, pp. 121–130. ACM Press (2010)
2. Berry, L., Bartram, L., Booth, K.S.: Role-based Control of Shared Application Views. In: Proc. UIST 2005, pp. 23–32. ACM Press (2005)
3. Bourges-Waldegg, D., Duponchel, Y., Graf, M., Moser, M.: The Fluid Computing Middleware: Bringing Application Fluidity to the Mobile Internet. In: Proc. SAINT 2005, pp. 54–63. IEEE (2005)
4. Brdiczka, O.: Integral Framework for Acquiring and Evolving Situations in Smart Environments. Journal of Ambient Intelligence and Smart Environments 2, 91–108 (2010)
5. Chang, T.H., Li, Y.: Deep Shot: A Framework for Migrating Tasks Across Devices Using Mobile Phone Cameras. In: Proc. CHI 2011, pp. 2163–2172. ACM Press (2011)
6. Cheverst, K., Dix, A., Fitton, D., Kray, C., Rouncefield, M., Sas, C., Saslis-Lagoudakis, G., Sheridan, J.G.: Exploring Bluetooth-based Mobile Phone Interaction with the Hermes Photo Display. In: Proc. MobileHCI 2005, pp. 47–54. ACM Press (2005)
7. Dearman, D., Pierce, J.S.: It's on My Other Computer! Computing with Multiple Devices. In: Proc. CHI 2008, pp. 1144–1153. ACM Press (2008)
8. Dees, W.: Usability of Nomadic User Interfaces. In: Jacko, J.A. (ed.) Human-Computer Interaction, Part III, HCII 2011. LNCS, vol. 6763, pp. 195–204. Springer, Heidelberg (2011)
9. Fischer, C., Gellersen, H., Gostner, R., Guinard, D., Kortuem, G., Kray, C., Rukzio, E., Strengin, S.: Supporting Device Discovery and Spontaneous Interaction with Spatial References. Personal and Ubiquitous Computing 13(4) (2009)
10. Ghiani, G., Isoni, L., Paternò, F.: Security in Migratory Interactive Web Applications. In: Proc. MUM 2012. ACM Press (2012)
11. Ghiani, G., Paternò, F., Santoro, C.: Push and Pull of Web User Interfaces in Multi-Device Environments. In: Proc. AVI 2012, pp. 10–17. ACM Press (2012)
12. Google. The New Multi-screen World: Understanding Cross-platform Consumer Behaviour. Research Report (August 2012), http://services.google.com/fh/files/misc/multiscreenworld_final.pdf
13. Greenberg, S., Marquardt, N., Ballendat, T., Diaz-Marino, R., Wang, M.: Proxemic Interactions: The New Ubicomp? Interactions 18, 42–50 (2011)
14. Holleis, P., Otto, F., Hußmann, H., Schmidt, A.: Keystroke-Level Model for Advanced Mobile Phone Interaction. In: Proc. CHI 2007, pp. 1505–1514. ACM Press (2007)
15. Marquardt, N., Diaz-Marino, R., Boring, S., Greenberg, S.: The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. In: Proc. UIST 2011, pp. 315–326. ACM Press (2011)
16. Nielsen, J.: Usability Engineering. Morgan Kaufman Publishers Inc., San Francisco (1993)
17. Rekimoto, J.: Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In: Proc. UIST 1997, pp. 31–39. ACM Press (1997)
18. Rekimoto, J., Ayatsuka, Y., Kohno, M., Oba, H.: Proximal Interactions: A Direct Manipulation Technique for Wireless Networking. In: Proc. Interact 2003, pp. 511–518. IOS Press (2003)

[1] http://www.smarcos-project.eu

19. Schilit, B., Adams, N., Want, R.: Context-Aware Computing Applications. In: Proc. WMCSA 1994, pp. 85–90. IEEE Computer Society Press (1994)
20. Schilit, B.N., Sengupta, U.: Device Ensembles. IEEE Computer 37, 56–64 (2004)
21. Schmidt, A.: Implicit Human Computer Interaction through Context. Personal and Ubiquitous Computing 4(2/3), 191–199 (2000)
22. Sellen, A., Kurtenbach, G., Buxton, W.: The Prevention of Mode Errors through Sensory Feedback. Human Computer Interaction 7(2), 141–164 (1992)
23. Tabard, A., Hincapié-Ramos, J.D., Bardram, J.E.: The eLabBench in the Wild – Supporting Exploration in a Molecular Biology Lab. In: Proc. CHI 2012, pp. 3051–3060. ACM, New York (2012)
24. Wäljas, M., Segerståhl, K., Väänänen-Vainio-Mattila, K., Oinas-Kukkonen, H.: Cross-Platform Service User Experience: A Field Study and an Initial Framework. In: Proc. MobileHCI 2010, pp. 219–228. ACM Press (2010)