Distributing Web Components in a Display Ecosystem Using Proxywork

Pedro G. Villanueva, Ricardo Tesoriero, Jose A. Gallud
University of Castilla-La Mancha, ISE Research Group, Information System Department
Campus Universitario de Albacete, (02071) Albacete, Spain
{pedro.gonzalez; ricardo.tesoriero; jose.gallud}@uclm.es

Proxywork is a system that allows users to distribute user interface components of Web applications among a set of devices. The distribution is controlled by the user through a set of primitives (i.e. show, hide, copy, move, etc.) attached to Web page components. As these operations are automatically attached to Web page components on runtime by the Proxywork, Web pages do not require any extra information to be distributed among different devices. To illustrate how to distribute Web application user interface components, this paper presents a Web site as a study case showing the results of performing user interface distribution operations among devices running on different platforms. This paper also presents the evaluation of the Proxywork system.

Distributed User Interfaces, Web, Proxywork.

1. INTRODUCTION

In a distributed user interface (DUI) scenario, users distribute one or many elements/s of one or many user interface/s to support one or many user/s to carry out one or many task/s on one or many domain/s in one or many context/s of use [9].

Web applications do not offer the possibility to distribute UI components from one device to another one. For instance, suppose that you are viewing a Web site using a Smartphone, and you want to read an article in a bigger display such as your desktop computer.

In an ideal situation, you should be able to "select" the article from your Smartphone, and "migrate" it to the Web browser running in the desktop computer. However, in the real life, it is not as simple as it seems, because Web browsers do not support this feature.

In this paper, we present the Proxywork system which offers the ability to transform Web applications designed to run on a single display into Web Applications running on a DUI. This transformation is performed at runtime by the means of a Web proxy which is able to distribute a Web application UI across different platforms.

In order to carry out this task, Web browsers connected to the Proxywork proxy receive a modified version of Web pages they have requested to the proxy. This modification attaches a menu on each UI component to display, hide, copy, or distribute the UI component to the rest of the browsers that are connected to the Proxywork proxy. Therefore, the Proxywork proxy is also in charge of orchestrating how UI components are displayed on devices running on different platforms.

This article is organized as follows: Section 2 presents the most relevant related work regarding DUIs. A description of the Proxywork system is presented in Section 3. Section 4 presents a case of study that shows how Proxywork supports the Web site scenario we have mentioned. Later on, the Section 5 shows a quantitaive evaluation of the Proxywork system. Finally, Section 6 presents conclusions and future work.

2. RELATED WORK

DUIs are relatively recent and are continuously evolving; therefore, there are few frameworks that support the distribution of, elements or the whole, user interface. Moreover, most frameworks do not support full-fledged DUIs.

From the common programming language perspective, UI development toolkits such as Java Swing or Windows Presentation Foundation (WPF) do not support DUI concepts. They just allow developers to assign UI components to a single container/context. However, once these components are assigned to their container, they cannot be reassigned or redistributed to another container or context residing on different runtime platforms. An example of a fixed distribution of UI elements between smartphones and TVs is presented in [8]. Note that the UI component

distribution is predefined because no reconfiguration is allowed at runtime.

Some approaches allow the distribution of UI components at runtime with limitations. For instance, Web browsers allow users to open a new window to display images that are embedded into a Web page. However, the distribution is limited to the same runtime platform. In addition, the distribution is limited to images and a couple of HTML elements, such as links. You cannot distribute a DIV tag defining a panel on a form.

The granularity of UI components to be distributed is fixed, and often limited to coarse-grained UI components such as dialogs or windows. It leads to the fact that is not possible to distribute or replicate widgets.

Han et al. [3] presented a work where a Web page is split into several partial pages which are distributed to all the users. Our approach supports multi-device and multi-user Web browsing where clients are connected to the server that delivers the page. In this work, the proxy splits pages according to device and user constraints. Therefore, each Web page is represented by a XML file containing specific tags to configure how the Web page is split among users and devices. Although, this work is similar to our proposal, the advantage of our approach lays on the lack of the definition of configuration tags at design time to distribute the UI. These tags are replaced by the enrichment of the Web page with distribution primitives at runtime.

A similar work, implemented by Luyten and Coninx in [4], shows how an interactive system can be distributed among several peer devices. Our approach is significantly different from this work because it allows all Web applications be distributed independently of the computing resources and the application design.

Luyten et al. [5] limit the granularity of the UI component distribution to design time.

A toolkit supporting DUIs was proposed in [6]. It is based on a widget distributed structure composed by 2 main parts. While the first part (the widget 'proxy') remains stationary within the process that created the widget; the other part (the renderer) is distributed to where the user can interact with it. This solution requires that the user interface to be implemented as an extension of the TCL/TK toolkit. The main difference regarding to our approach is the Web as a platform, we are focusing on migrating parts of Web applications using XHTML, CSS and Javascript. Moreover, contrary to [6], our solution does not impose any authoring technique to deploy applications (the proxy modifies pages automatically to support the distribution).

Another solution regarding partial Web migrations is presented in [2]. This approach allows users to select parts of existing interfaces interactively, and migrate them to different target devices. To achieve this goal, this approach uses a native application that allows users to select the parts of the web application UI to migrate. The main difference between this work presented and ours is the lack of a native application to distribute UI components, the interface to distribute UI components is embedded into UI components, instead.

Another way to achieve the distribution of UIs is presented in [1]. In this work, Bandelloni el al. claim that a Web UI can be partially or completely migrated. A partial migration of the UI implies the splitting the UI in two or more parts that run on separate devices. To achieve this goal, extra information should be added to the UI definition using a flexible language to describe the UI presentation. Again, this approach requires extra information that should be added to Web applications beforehand in order to distribute the UI.

Melchior et al. [7] presents a catalogue of distribution operations, and a toolkit to build applications based on this catalogue. The catalogue of distribution operations defines the following set of primitives: SET, DISPLAY, UNDISPLAY, COPY, MOVE, REPLACE, MERGE, SEPARATE, SWITCH and DISTRIBUTE. The toolkit provides a native command line interface to allow manual redistribution of UI components at runtime. The approach we are presenting in this paper does not depend on a native application to distribute UI components, and distribution operations are attached to components, instead. It allows users to manipulate components directly from the UI.

3. PROXYWORK: DISTRIBUTING WEB APPLICATIONS

Proxywork acts as a proxy where a set of devices are registered. These devices send all their Web requests to the proxy in order to get a response enriched with distribution operations. The Figure 1 shows the overview of the Proxywork architecture.

Let suppose that Proxywork proxy is hosted on the x.x.x.x IP address listening on port 80. Devices that are part of the same display ecosystem should set the Web browser proxy IP address to x.x.x.x and the port to 80.

Once the device Web browser is registered, the request follows these steps to display the Web page enriched with distribution operations: First, the request for the page http://www.yyy.com departs from device browser, and arrives to the Proxywork proxy. The Proxywork system requests the Web resource to the Web server where the

application is hosted (http://www.yyy.com). The Web server returns the Web resource to the Proxywork. Proxywork modifies the resource by inserting HTML, CSS and JavaScript extra code in the page in order to add distribution primitives, and provide a list of devices where users are able to distribute UI components. Proxywork returns the modified Web page supporting distribution primitives to the device Web browser that sent the request. Finally, the Web browser shows the distributable Web page (http://www.yyy.com).

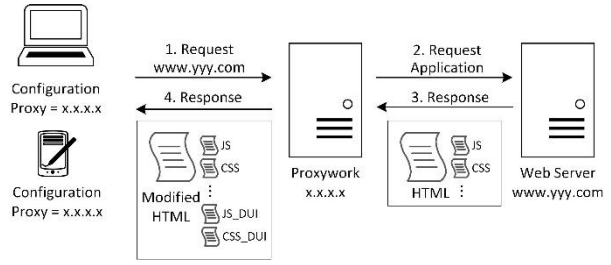

Figure 1: *Overview of the Proxywork architecture*

To carry out the transformation of a simple Web page into a distributed Web page, Proxywork defines 5 modules to process Web pages.

The **Code Manager** module is responsible for inserting the code containing the distribution operations into the Web pages requested by Web browser devices.

The **Device Manager** module keeps the status of the Web browsers connected to the distribution environment managed by the proxy.

The **Link Manager** module is responsible for translating Web page links to suit distributed user interface behaviour.

The **Granularity Manager** module sets which parts of the Web page can be distributed, and which cannot.

The **Distribution Manager** module is responsible for showing or hiding UI components of the UI. It keeps the distribution state for each device.

3.1. The granularity

The granularity of a Web page defines which parts of the Web page users are able to distribute (or make sense to be distributed) across devices, and which are not.

These parts are identified in terms of HTML tags. Therefore, Proxywork sets the granularity to the <DIV> HTML tag level by default because this tag represents groups of graphically related tags. However, users are able to set other HTML tags to make the distribution more flexible.

3.2. The Proxywork distribution process

This section describes the workflow process that the Proxywork system follows to transform a Web Application into a distributed Web Application.

The process begins when a Web request arrives to the Proxywork. If the device is not registered, the request is forwarded to a registration page to request the device name. Once the device is registered, Proxywork forwards the request to the Web page that was initially requested. If the device is already registered, the proxy checks if any content was distributed to this device. If it is so, the device request is forwarded to the content assigned to this device.

If the device browser does not need to reload its contents (no changes), the proxy checks if the request is a distribution operation. If the URL matches to a distribution operation, the device and operation IDs are extracted from the URL and the proxy updates the devices affected by the distribution operation in the registration table.

If the request is not a distribution operation, the proxy checks whether the request is related to an “internal navigation link”. If the request is related to an “internal navigation link”, the proxy extracts the link distribution parameters (i.e. the HTML tag ID and the target URL) to update the device Web browser contents accordingly. However, if the request is not an “internal navigation link”, the proxy sends the request to the Web server that hosts the requested resource.

When the proxy receives the resource, it checks if it is a HTML page. If is not a HTML page, the resource is returned to the device without any modification. However, if the requested resource is a HTML page, the proxy modifies the page to transform it into a distributable Web page.

To carry out all this process, the Code Manager delegates to the Link Manager the modification of the navigation links of the Web page. The Code Manager also delegates to the Granularity Manager the selection of the HTML tags that are enriched with distribution capabilities. Finally, the Code Manager sends the modified page back to the browser that made the request.

3.3. Distribution operations

This section describes the set of distribution operations supported by Proxywork.

The **Connect** operation associates the IP address of a device Web browser to a device name. The connection occurs when the Web browser request a Web resource for the first time. As result the user sends the device name to the proxy through a form. Once the device is registered, the name is used to parameterize the distribution operations that require a target device Web browser (i.e. Copy and Distribute).

The **Disconnect** operation releases a device Web browser from the distribution environment. Once the device is disconnected, the device name is removed from the list of parameters that are set to distribution operations.

The **Rename** operation allows users to change the registered name of a device Web browser.

The **Display/Hide** operation allows users to display/hide UI components.

The **Copy** operation allows users to copy UI components from one device Web browser to another one connected to the same distribution environment. This operation requires the target device Web browser as parameter.

The **Distribute** operation sends UI components from one device Web browser to another one connected to the same distribution environment. This operation requires the target device Web browser as parameter.

To illustrate the difference between Copy and Distribute suppose that A and B are two device Web browsers. If a user performs a Copy operation on a UI component, called X, running on A to affect B; all subsequent operations performed on X do not affect B in any way. However, if a user performs a Distribute operation on a UI component, called Y, running on A to affect B, the Y UI component disappears from A and appears on B. Note that all operations performed on Y are targeted to A.

Finally, if two device Web browsers A and B perform the Distribute operation on the same UI component Z of the same Web page to a third device Web browser C, operation performed on Z affects both, A and B. That is to say that while the Copy operation "copy" UI component instances, the Distribute use the "reference" of UI components which is defined by the ID attribute of the tag.

4. CASE OF STUDY

This section presents how Proxywork is employed to support DUI for Web applications in a real case of study[1].

The case of study describes how the navigation bar of a Web application can be distributed from a desktop or laptop computer to a smartphone. The idea is the creation of a "remote control" of the Web application using the smartphone touch screen.

The distribution is carried out on 2 devices: Dell XPS M1530 laptop computer running the Microsoft Windows 8 operating system and Chrome browser; and Nokia Lumia 900 smartphone running the Microsoft Windows Phone 8 operating system and Internet Explorer 9 browser.

[1] A demo is available at http://youtu.be/MEC2Y5rVGXQ

The proxy of both Web browsers points to the IP address and port where Proxywork is running. We register the laptop as the "DellXPS" device and the smartphone as the "Lumia" device. Then, we set the URL of the laptop web browser to http://www.uclm.es. As soon as the page loads, users are able to see the context menus that show distribution operations.

The **Figure 2** (a) shows the laptop display when the user cursor is over the navigation menu and select the device to distribute the UI. In this case, the "Lumia" device (see zoom on **Figure 2** (a)).

When users click on the device, the navigation bar disappears from the "DellXPS" display (see **Figure 2** (b)) and appears on the "Lumia" device (see **Figure 2** (c)). Thus, when users click on an anchor of the navigation bar that was distributed from the "DellXPS" to the "Lumia" device, the target of this content is set to the "DellXPS" (see **Figure 2** (d))

5. USABILITY EVALUATION

To validate the Proxywork proposal presented in this work, we have performed a preliminary quantitative evaluation with users within a possible application scenario.

The set of Proxywork functions to test are: (a) configure a device to use Proxywork, (b) distribute parts of a web page between devices and (c) share parts of a web page between devices. To perform the evaluation, we will focus on the effectiveness, efficiency and satisfaction.

5.1. Goal and apparatus

The goal of the quantitative evaluation is to compare Proxywork against a set of systems that users currently employ to transfer parts of web pages among devices. Thus, the main goal is to prove the hypothesis: "Users are more efficient, using Proxywork to perform the task of copying and distributing parts of a web page between two devices, than employing the user's choice system". In addition, we measured users' satisfaction after using Proxywork.

The hardware employed in the test was the same used in the case study plus the user's device.

5.2. Participants

The profile of the participants potentially interested in the application's functionality is broad. We selected a user population of 6 users. All of them have an advanced technological knowledge in the specific technique that they have selected to realize the tasks' test, and have previously transferred information among computers using alternative methods. In all cases, this was the first time that they have used Proxywork.

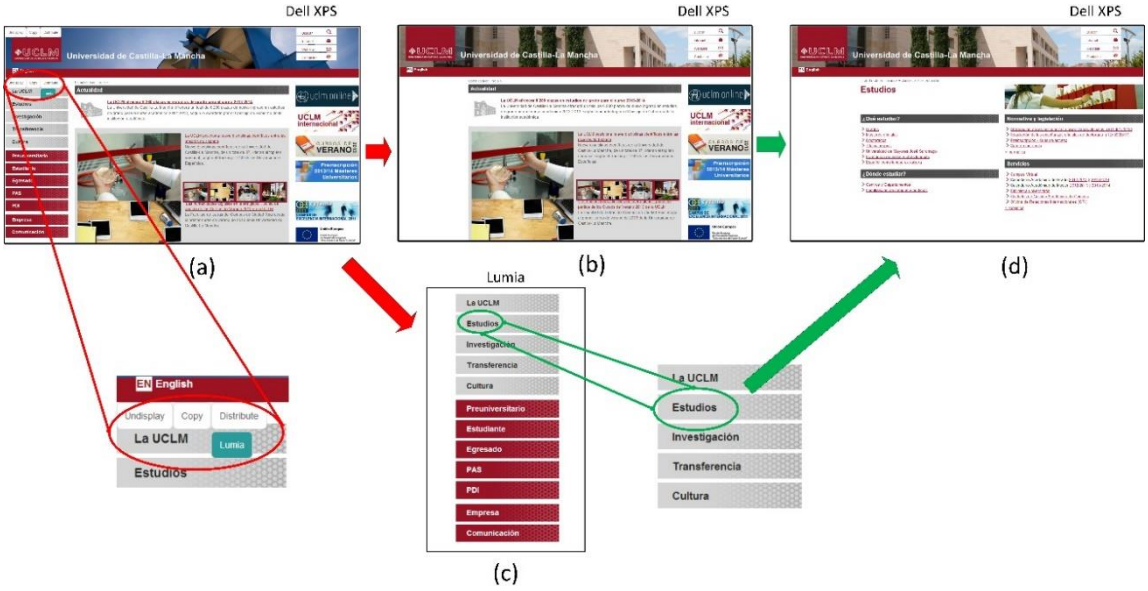

Figure 2: *Distributing the navigation menu. The user decides to move the main menu (a) and is removed (b). The main menu appears in the Lumia device (c) and change the content in the first device (d).*

5.3. Definition of tasks

All participants were asked to perform the same 3 tasks in the same conditions. These tasks are defined as follows:

Task 1: The user must configure the devices that he/she wants to use so that they can be used on the raised scenario. In the case of Proxywork, the browsers must be configured with the IP and port of Proxywork, in addition the devices must be registered.

Task 2: On her device, the user has a web page with information relevant to the meeting. He/she has to show some of that information in the projector of the room.

Task 3: On her device, the user has a web page with an image that must copy to the organizer of the meeting.

5.4. Procedure

The testing period was a week from the beginning to the end of the experiment. Before the test, users were introduced with an explanation of the tasks to be performed. Later, they were asked to choose the system they wanted to use to transfer the information as the “competitor” system.

In addition, the laboratory was setup with users’ selected system and the Proxywork. Users performed the three tasks with the system they have chosen, and later, the same three tasks were performed using the Proxywork (within-subjects test). After each testing session, users fulfilled the System Usability Scale (SUS) [2] questionnaire.

5.5. Dependent variables

The efficiency evaluation is measured by the task completion time metric. The satisfaction evaluation is based on the analysis of the SUS questionnaire where each question was valued in a scale from the range of 1 to 5 (1 strongly disagree, 5 strongly agree).

5.6. Results and discussion

As result of the test session we collected the information shown in Table 1. The Table 1 shows users’ time to complete each task. Note that all users completed all tasks with both systems.

Table 1: *Completion time for each task (in sec.)*

ID	*Chosen system*	*Chosen system*			*Proxywork*		
		T_1	T_2	T_3	T_1	T_2	T_3
1	Hotmail	41	37	42	70	14	15
2	Gmail	53	41	33	71	15	12
3	Gmail	43	60	64	121	9	16
4	Gmail	70	47	32	83	12	10
5	Pen Drive	15	82	64	68	14	13
6	Pen Drive	17	81	60	105	10	14

The collected data reveals that using the Proxywork system, both task 2 and task 3, were performed on an average time that is less than a quarter the time employed to perform the same task with the Email technique and less half the time employed to perform the same task with the Pendrive technique. Figure 3 shows that users are more efficient when they use the Proxywork instead of the system they have chosen.

On the other hand, the SUS score is 86.8 regarding a maximum of 100. This score indicates that users

have reached an acceptable level of satisfaction with Proxywork.

A detected usability problem is that Proxywork configuration (Task 1) is too complicated for novice users (see Figure 3).

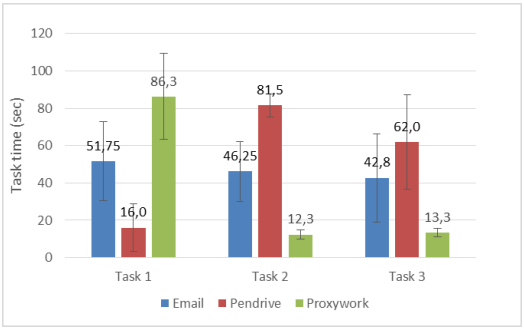

Figure 3: *The confidence intervals around the average time for each task in sec.*

6. CONCLUSION AND FUTURE WORK

This paper presents Proxywork, a novel system to support multi-platform distributed user interfaces. It was implemented as a proxy that transforms traditional Web pages into DUI Web pages dynamically on runtime. It was developed using Web standards such as XHTML, CSS and JavaScript in order to be supported by most platforms and not to depend on native applications.

This proposal implements 7 distribution operations or primitives: Connect, Disconnect, Rename, Display, Hide, Copy and Distribute. However, it is not difficult to add new operations due to the flexibility of the implementation.

To show the capabilities of Proxywork, a case of study was presented, which shows how to deal with the distribution of Web application navigation bars where users distribute the navigation bar from a laptop or desktop computer to a smartphone in order to use the smartphone as a “remote control” of the computer display.

The results collected in the Evaluation section have validated the hypothesis: “Users are more efficient using Proxywork to perform the task of copying and distributing parts of a web page between two devices, than employing user’s choice system”.

As future work, we are working on a new version of the environment that implements new distribution operations such as Clone, Replace, Merge, Switch and so on.

7. ACKNOWLEDGMENTS

We thank all, the CICYT-TIN 2011-27767-C02-01 Spanish project, the PPII10-0300-4174 and the PII2C09-0185-1030 JCCM Projects for supporting this research. We also would like to thank to the “Programa de Potenciación de Recursos Humanos” from the Scientic Research, Technological Development and Innovation Regional Plan 2011-2015(PRINCET).

8. REFERENCES

[1] Bandelloni, R. and Paternò, F. Flexible Interface Migration. In Proceedings of Intelligent User Interface 2004 (IUI 04), pages 148–155, 2004.

[2] Ghiani, G., Paternò F., Santoro C. On-demand CrossDevice Interface Components Migration, Proceedings Mobile HCI 2010, pp. 299 – 308, 2010, ACM Press.

[3] Han, R., Perret, V., and Naghsineh, M. 2000. WebSplitter: A Unified XML Framework for Multi-Device Collaborative Web Browsing. In Proc. of the ACM Conf. on Computer Supported Cooperative Work, pp. 221-230.

[4] Luyten, K. and Coninx, K. 2005. Distributed User Interface Elements to support Smart Interaction Spaces. In Proc. of the 7th IEEE Int. Symposium on Multimedia, IEEE Comp. Society, Washington, DC, pp. 277-286.

[5] Luyten, K., Van den Bergh, J., Vandervelpen, Ch., and Coninx, K. Designing distributed user interfaces for ambient intelligent environments using models and simulations. Computers & Graphics 30, 5 (2006), 702–713.

[6] Melchior, J., Grolaux, D., Vanderdonckt, J., and Van Roy, P. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications, in Proceedings of EICS 2009, ACM, 2009, 69-78.

[7] Melchior, J., Vanderdonckt, J. and Van Roy. Distribution Primitives for Distributed User Interfaces. Distributed User Interfaces Workshop, ACM CHI 2011, Vancouver, BC, May 7th, 2011. ISBN 978-84-693-9829-6. Pag 29-32.

[8] Sjölund, M., Larsson, A., Berglund, E. Smartphone Views: Building Multi-Device Distributed User Interfaces. In: Proc. Of MobileHCI’2004. LNCS, Springer (2004), 507-511.

[9] Vanderdonckt, J. Distributed User Interfaces: How to Distribute User Interface Elements across Users, Platforms, and Environments. Proc. of XI Interacción, 20-32, 2010.