Proxywork: Framework to transform from Web Application to Distributable User Interface Web Application

Pedro G. Villanueva, Ricardo Tesoriero, José A. Gallud
Computer System Department. University of Castilla-La Mancha
Campus Universitario de Albacete
(02071) Albacete, Spain
[pedro.gonzalez, ricardo.tesoriero, jose.gallud]@uclm.es

ABSTRACT
The growth of cloud services and new tools for web development is greatly favoring the growth of the number of Web Application that we use today. Furthermore, is becoming more common that we have a variety of interconnected devices able to visualize any type of Web Application. In general, today Web Applications do not offer the possibility to distribute parts of the interface from one device to another. For example that the navigation menu, from a Web Application displayed on a PC, can be sent to a mobile device thus allowing navigation from your mobile device to your PC. In this paper, we present a Framework in the form of proxy that transforms any Web Application into a Web Application with distributable user interface, at the time in which is requested through the browser. In this way, any Web Application offers the possibility to distribute parts of your interface between different devices. The Framework includes a set of menus on each UI component that allows users to perform certain actions (display/undisplay, copy and distribute) with each component and orchestrating the parts that are migrated on different devices involved in the migration. Throughout the work presents the implementation of the Framework and demonstrates its functionality on a particular Web Application (University of Castilla - La Mancha) using multiple devices with different platforms.

Categories and Subject Descriptors
H.5.m [Information Interfaces and Presentation (e.g., HCI)]: Miscellaneous; D.3.3 [Programming Languages]: Language Constructs and Features

General Terms
Design, Human Factors, Languages.

Keywords
Distributed User Interfaces, Web, Proxywork.

1. INTRODUCTION
Could computing offers many advantages such as the centralization of data, security, scalability, and saving on maintenance and cost, etc. These advantages are favoring that every year have a greater number of companies and individuals who migrate their applications to this type of architectures.

According to a study carried out by Mimecast[1]. Currently, 70% of companies use cloud services. This has driven the increase in Web Applications that provide functionality to manage all information and presenting data to end users quickly and accessible from anywhere.

Another aspect that has greatly influenced the proliferation of Web Applications are the advances in new tools of web development. These tools provide great power and speed at the time of development and allow you to build rich user interfaces.

On the other hand, is increasingly more common to see users who have and do use in their every day from a variety of personal devices (Desktop, Laptop, Smartfone, Tablet, etc.) connected to Internet through Wi-Fi connection or data connection rates. In addition, all of them equipped with a browser capable of rendering any type of Web Application.

Clearly identified these two trends such as the rise of Web Applications and the diversity of interconnected personal devices, we must ask if we exploit the advantages offered by the paradigm of Distributed and Distributable User Interfaces.

Today Web Applications do not offer the possibility to distribute parts of a user interface from one device to another. For example, imagine that we are viewing a Web Application of a newspaper in our Smartphone. At a certain point, we want to read some of the news that we have found, but in a bigger screen as our desktop computer. It would be ideal to bring news directly from our Smartphone to desktop screen directly and transparently.

This is the problem that we have detected and solved with the work presented here. In this paper, we present a Framework, implemented as a proxy, which offers the ability to transform any Web Application into a Web Application with distributable user interface. This transformation is performed at runtime when the application is requested by the browser from any device and platform. In this way, any Web Application offers the possibility to distribute parts of the interface between different devices.

The Framework we propose makes certain changes in the behavior of the Web Application to insert a set of menus on each UI component. These menus allow users to perform certain actions (display/undisplay, copy and distribute) on the UI components. Besides, the Framework is responsible for orchestrating the parts that are migrated on different devices involved in the migration.

Throughout the paper presents the implementation of the Framework called *Proxywork* and demonstrates its functionality on a specific Web Application using multiple devices with different platforms.

[1] Taklin'clud.com: URL=http://talkincloud.com/cloud-computing-research/survey-71-organizations-using-unsanctioned-cloud-apps

The document is structured in the following way: in the current section, we present a gentle introduction to the motivation that has come to do the work done, we present the problem and we are moving forward with the proposed solution. Section 2 is a review of some work from the point of view of multiple display systems and Frameworks focused on DUIs, besides, section 2 compares the advantages of our proposal with related works. Later, in section 3, we describe in detail the proposal submitted and finally, in section 4 are broken a number of conclusions.

2. RELATED WORK

In this section, we compare the advantages and disadvantages of the major related work in order to identify the specific aspects of our proposal. Works related to our proposal can be grouped into three main lines: Multiple Display System, Distributed User Interfaces and Framework to support Distributed User Interfaces.

2.1 Multiple Display Systems and Distributed User Interfaces

Many research papers present environments with multiple displays as well as design of distributed user interfaces that exploit the advantages offered by these systems.

A typical example is *Office of the Future* [14], envision a workplace in which every surface serves as a high-resolution projected display. In their system, they modify images projected onto particular surfaces so that they appear correctly to observers at known locations.

In the work called *i-LAND* [18], the authors have also worked on integrating real architectural and virtual information spaces. They have populated this environment with various physical components, each with its own associated display device. Together these displays provide physical affordances that aid in content organization and work process control. Another similar work is *Augmented Surfaces* [15] implemented by Rekimoto and Saitoh, but have focused on interaction techniques, as the technique called hyperdragging, users utilize the physical relationship between devices to transfer information between them.

InfoCockpit [19] improves human memory for information viewed. Authors not only use multiple monitors to spatially distribute information and engage human memory for location, but also present synthetically created visual context on large ambient projection displays to leverage human memory for place. In a similar work called *Kimura* [11], utilizes projected peripheral displays to support the perusal, manipulation, and awareness of background activities in order to manage multitasking between multiple working contexts.

Finally, other works that show multiple display systems are *Group scribbles* [16], projects supporting brainstorming in face-to-face as *WeSpace* [22], or digitalized rooms such as *Connectable* [20].

Furthermore, the design of user interfaces for this type of system is widely. Probably the first DUI ever was developed as a system that distributed a UI over many workstations connected to the same network and running the same operating system [3] thanks a to a connector mechanism.

In [1] and [21], a part or whole of a DUI can be migrated from one platform to another at run-time. The underlying architecture is a client-server architecture that maintains in a central position the internal state of the DUI.

WallShare [6] is a collaborative system that can distribute the interfaces between different devices such as mobile phones, PDAs, laptops, etc., and a shared space to be displayed via a projector on a surface such as a wall. Another similar work is *Dynamo* [8].

CSchool [4] is system to support the administration of schools educational process in the cloud by applying DUI.

The Framework that will be presented in this paper is significantly different form this previous work in that it is not a distributed user interface system, but it is a Framework or tool to transform Web Applications to Distributed User Interfaces Web Application in real time. Previous work demonstrate the variety of multiple displays system and our proposal to provide a Framework to cover this scenarios with Distributed User Interfaces Web Application.

2.2 Framework to support DUIs

The domain of Distributed User Interfaces (DUI) is still in evolution and there are few Frameworks allowing the creation of DUIs, moreover, in most pieces of work, there is almost no genuine DUI.

There are toolkits to develop UI such as Java Swing or Windows Presentation Foundation (WPF), but they do not support DUIs. The UI elements simply remain in their initial context, while communicating with each other, but without redistribution. There is some distribution of UI elements, but it is mainly predefined and opportunistic: no configuration of the distribution at run-time. Sjölund presented in [17] a work that the repartition of UI elements across the Smartphone and the TV is fixed. It is not possible to rearrange their distribution. Some works allow distribution at run-time but with some limitations. The UI elements subject to this redistribution are mainly containers, such as windows or dialog boxes. The problem is that the granularity of UI distributed elements is often coarse-grained; it is not possible to distribute at the widget level.

In addition, they do not support replicability, i.e. when another platform comes in the context of use, it is hard to migrate on this platform parts that have already been transferred to other platforms.

In [7], a web page is split in partial pages which will be replicated to all the users. The framework supports multi-device and multi-user Web browsing where clients connect to a server which delivers the page. A proxy split the pages in respect to the device and user constraints. Each page is in a XML file with specific tags to configure how the Web page will be split among the different users and devices. This work is very similar our proposal, but our proposal allows distributing parts of UI from the Web Application in runtime.

A similar work implemented by Luyten and Coninx [9], shown how an interactive system can be distributed among several peer devices. Their approach relies on the fact that nowadays most computing resources are network-enabled and publish their device profile like in UAProf or CC/PP. It raises the opportunity for supporting collaborative tasks with the same user interface with little or no extra effort from the user interface designer. Our approach is significantly different from this work in that our approach provides a Framework to all Web Applications. This work only presents an example to support the design, development and deployment of DUI.

In [10], there are already attempts to model the distribution. The granularity is however limited to tasks that are predefined before the application starts.

A toolkit for Distributed User Interfaces was proposed in [12]. It is based on a widget distributed structure composed of two main parts: one part (the 'proxy' of the widget) remains stationary within the process that created the widget; the other part (the renderer) is distributed and migratable and the user can interact with it. The toolkit is based on a peer-to-peer architecture in which a multi-purpose proxy is connected to one or more rendering engines able to render (partially or entirely) a graphical user interface. 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 partially migrating any Web application developed with the standard Web languages (XHTML, CSS) and Javascript, moreover, our solution differs in that Web applications can be migrated without posing any constraint on the authoring technique to use for developing the applications.

The work [5] presents a solution for partial Web migration, it allows users to interactively select parts of existing interfaces and have them migrate to a target device. This approach has a native application that allows the user to select the parts of the web application interface to migrate. Our approach is not dependent on a native application.

In [2], Bandelloni and Paterno have shown that a web interface can be partially or completely migrated. Here, partial migration implies the web interface is split up in two or more parts that each run on a separate device. This is accomplished by exploiting information that is available about the interactive system and by using a flexible language to describe the interface presentation. Our approach is significantly different from this work in that our approach provides a Framework to all Web Applications.

The work [13] proposes a catalog of distribution operations and a toolkit based on this catalog. The catalog of distribution operations is composed of: SET, DISPLAY, UNDISPLAY, COPY, MOVE, REPLACE, MERGE, SEPARATE, SWITCH and DISTRIBUTE. The toolkit provides a native command line interface to allow manual redistribution at runtime. Our approach is not dependent on a native application.

In summary, the Framework that will be presented is significantly different from this previous work in that it provides a unique combination of the following features:

- Proxywork supports DUIs and not only UIs.
- Proxywork transforms any Web application developed with the standard Web languages (XHTML, CSS, etc.) and Javascript to a Distributed Web Application.
- Allows migrating Web applications without posing any constraint on the authoring technique to use for developing the applications.
- The transformation is done in runtime, when users request the Web Application.
- Because the Framework is for Web Application, it is supported by all platforms.
- The granularity is not limited to tasks that are predefined before the application starts. The granularity of distribution can range from the application level to the widget level: an entire application can be distributed across platforms for instance, but also the different components of any widget.
- Proxywork is not dependent on a native application to distribute application´s elements.

3. FRAMEWORK TO TRANSFORM WEB APPLICATION

In this section, we start showing two case studies that we have chosen to demonstrate the power of our Framework named *Proxywork*. Afterwards, we present Framework software architecture, functions that supports, details of granularity for the permitted actions and runtime architecture that allows that the Framework can be applied to any Web Application. Finally we apply the Framework in the case studies.

3.1 Case studies

We could list countless case studies for the Framework which we propose, but we will only present two case studies to show the power of the Framework.

The first case study is Distributed Brower in Web Application. This scenario is very common when you are viewing a Web Application that has a menu to navigate through the different sections or categories of application. This menu can be distributed to another device (i. e. a Smartphone). Thereby, the Smartphone can remotely control the navigation of the Web Application as a remote control in question. Figure 1 shows the action of distributing the navigation menu of the University of Castilla - La Mancha Web Application. Application is initially displayed on the computer and the menu is distributed to a Smartphone. Once distributed the menu disappears from the computer. If user clicks on the menu, the effect should occur on the computer.

Figure 1 Distributable menu in University of Castilla-La Mancha Web Application

Figure 2 Distributable news on University of Castilla-La Mancha Web Application

The second case study is Distributed News Reader. Another common scenario is news Web Applications. It is quite usual to access Web Applications of our favorite newspapers from our Smartphone. However sometimes it is not so comfortable to read an extensive news from the Smartphone. In these cases, it is more convenient to read the news on a large screen such as a computer or laptop. The news could be distributed from the Smartphone to a desktop computer to read the news on a large screen. Figure 2 shows the action of distributing a news from the University of Castilla - La Mancha Web Application. Initially, the Smartphone

shows the application that displays the news and the news is distributed from the Smartphone to a computer. The news disappears from the Smartphone and it appears only on the computer.

3.2 Software Architecture

As shown in Figure 3, the *Proxywork* is composed of 5 main modules: Code Manager, Devices Manager, Links Manager, Granularity Manager and Distribution Manager.

- **Code Manager**. This module is the main module. It is responsible for inserting the necessary code into Web Applications requested by devices. The inserted code allows application components can be distributed. In other words, it is in charge of transforming a normal Web Application into a distributable Web Application. It makes use of other modules to carry out its function.
- **Devices Manager**. This module is responsible for maintaining the status of the devices connected to the distribution environment. Whenever a device makes a request, it checks if it is the first time, if so prompts the user for a device name. Thus at all times, *Proxywork* knows connected devices for distribution.
- **Links Manager**. It is the module responsible for transforming internal links of Web Applications. In this way, when users press them, links have effect on the corresponding device if they have been distributed.
- **Granularity Manager**. This module is responsible for setting the granularity of distribution. It sets that parts of the web application interface can be distributed and which cannot. By default, the distribution is carried out at the container level (div tag in HTML). The granularity manager adds for each container a menu with functions that users can be performed, these functions will see in section 3.3.
- **Distribution Manager**: It is the module responsible for showing or hiding interface elements. It maintains the distribution state of each device. The module stores information of the Web Application for each device. Thus, *Proxywork* knows the elements of the application that should be visible/not visible for each device.

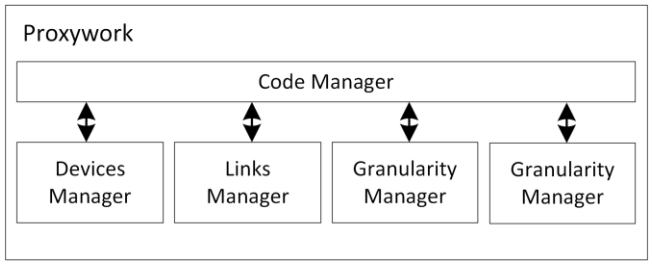

Figure 3 Proxywork software architecture

3.3 Functions

In the state of the art section, some authors propose a catalogue of distribution primitives. This section lists and describes the primitives that have been implemented within our proposal. In *Proxywork* have been implemented the most important primitives to demonstrate the potential of the Framework. In future work, we will expand the implemented primitives to cover the catalogue.

Proxywork implements the next primitives or actions:

- **Connect**. This action connects a device to a distribution environment. It associates the IP address of the device with a device name. User is asked for the device name when the first request from the browser is make. Once the device is connected, the name will be shown under each action that requires a target device (Copy and Distribute).
- **Disconnect**. It disconnects a device from a distribution environment. Once the device is disconnected, its name disappears from the list of target devices associated with actions that require a target device (Copy and Distribute).
- **Rename**. This action allows to change the registered name of the device.
- **Display**. This action allows you to display parts of the interface which have previously been marked with Hide action.
- **Hide**. This action allows you to hide parts of the interface that are being displayed.
- **Copy**. It allows to copy a part of the interface from one device to another device connected to the same distribution environment. This action will have an associated list with all devices connected to the distribution environment and to perform the action, the user must select one of devices.
- **Distribute**. This action sends part of the interface from one device to another device connected to the distribution environment. This action will have an associated list with all devices connected to the distribution environment and to perform the action, the user must select one of them.

The Connect action executes the first time user makes a request to a Web Application by using the browser. The *Proxywork* detects that the device is not connected and user is asked for a device name. The Disconnect action will be accessible by the user in any time via a button located on the top right. Disconnect actions is in all Web Applications provided by the Framework.

Display and Hide actions affect on the same device on which the action is launched.

The Copy action is performed in one device A and affects one device B and both are different devices. Furthermore, when the action is performed, if user performs any action on the copied component in the device A the results will affect the device A and if user performs any action on the copied component on the device B, results will affect the device B.

Finally, the Distribute action is performed in one device A and affects one device B and both are different devices. When a component is distributed from A to B, the component disappears from A and appears in B. In addition, any action the user performs in that visible component in device B will have impact on device A.

It should be noted that if two devices A and B perform the Distribute action of the same component in the same Web Application to a third device C, the actions on that component in the device C will affect both devices A and B.

3.4 Granularity

We have spoken to distribute parts of the web application interface between different devices, but not explained which parts of the interface may be distributed or it makes sense to distribute. This section describes how *Proxywork* resolves the issue of the granularity.

The granularity in the distribution of web applications is determined by the HTML elements that can be distributed. To get a finer granularity when distributing interface elements must go down to basic elements such as <a>, <p>, <img>, etc. On the other hand, if what is sought is a coarser granularity, we climb to more complex elements or other elements that group as the <body>, <head>, , <table>, etc.

A fine granularity can over overload the Web Application with too many distribution menus and it may confuse the user when choosing a component to distribute. In addition, there are items associated with other elements that lose sense if we separate them. An example can be a textbox and its explanatory label text. The advantage is that everything can be distributed and is more powerful.

On the other hand, a coarse granularity further simplifies distribution actions. But it loses power because there will be items that are not allowed distribute or at least separated from others.

Therefore, we have sought a compromise to implement *Proxywork*. The Framework takes into account the default tag like element that establishes the granularity. This label has been chosen because it usually used to group elements that have a common functionality or a common goal.

The Framework settings allow anytime establish other tags to make more flexible the granularity.

3.5 Global architecture

The Framework *Proxywork* is implemented as a proxy. All configured devices to operate in the distribution environment sent all web requests to our proxy.

Figure 4 shows the global architecture of the system by making use of the Proxywork.

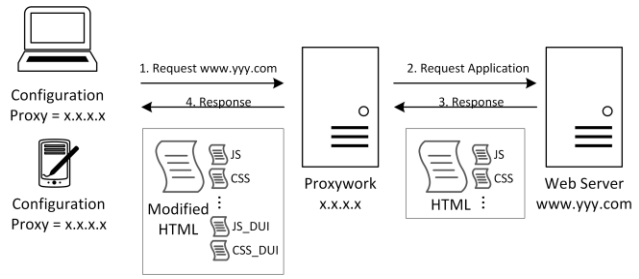

Figure 4 Proxywork global architecture

The Framework is hosted on a proxy server with an IP address x.x.x.x listening on port 80. Devices that will be connected to the distribution environment must be configured by setting the proxy with the IP address x.x.x.x and port 80.

The process from a user requests a normal Web Application (www.yyy.com), and distributable Web Application is displayed in the browser is as follows:

a. The request for the application www.yyy.com departs from device browser and arrives at *Proxywork*. See step 1 from Figure 4.

b. *Proxywork* requests for the application to the web server where the application is hosted (Web Server www.yyy.com). See step 2 from Figure 4.

c. The web server returns the application to the *Proxywork*. See step 3 from Figure 4.

d. *Proxywork* inserts extra code in the HTML page to add the distribution actions and a list of devices connected to the distribution environment. Besides, some links to CSS and JavaScript files.

e. *Proxywork* returns the application transformed into a distributable application to the device. See step 4 from Figure 4.

f. Finally, the browser displays the distributable application (www.yyy.com).

3.6 Workflow Proxywork

The diagram in Figure 5 shows in detail the workflow process that the Framework makes to transform the Web Application into a distributable Web Application.

The process begins when a web request arrives from the device browser to the proxy. The first check is if the device that has made the request is registered in the system. The proxy checks the IP address of the device. If device is not registered it is redirected to a registration page where it asks for the device name (see Figure 6). Thus, the device is registered in the system. Once registered the device browser is redirected to the page that was initially requested.

If the device is already registered, the proxy checks if another device has performed some distribution action and the device from the current request need to reload to display a new page. If this condition is true, the proxy returns to the browser the URL of the new page to visualize. Thus, the browser will request the new page and the process begins again.

If the device does not need to be recharged, the proxy checks if the request corresponds to action (actions commented in section 3.3). To detect an action, the proxy compares the URL with the URL pattern associated with actions. This pattern is explained later. If it corresponds to an action, the parameters (identifiers of action and target device) are obtained from the URL. Thereafter, the proxy updates the distribution of the affected devices and the same page from which the action was sent is returned to the device.

If the request is not an action, it checks whether the request corresponds to an internal navigation (links contained in the distributable web applications are modified to be detected by the Framework as we will see later). If it matches an internal navigation, the parameters (item ID from where was the navigation and navigation URL) are obtains from the URL. Thereafter, the proxy updates the distribution of the affected devices and returns the URL which should show the device.

If the request is not internal navigation, the proxy sends the request to the web server that hosts the requested resource. The proxy receives the resource and check if it is an HTML resource. If the resource is not HTML, corresponds to another type of file that should not be modified and the unchanged resource is returned to the browser on the device that made the request.

If the requested resource is HTML, the Framework modifies the page to turn it into distributable. Here is where the Code Manager module participates (see Figure 3). The Code Manager starts adding CSS files references to design the distribution menus. Module adds JavaScript code that is responsible for checking periodically to update the page if necessary, besides, JavaScript asynchronously updates the list of devices connected to the distribution environment. Subsequently, the Code Manager uses the Links Manager module (see Figure 3) to modify the

navigation links that contained in the page. Finally, the Code Manager uses the Granularity Manager module (see Figure 3) to select the items that can be distributed according to the granularity established. The Distribution Manager module insert the code to show the actions menu to each of those items.

Once the Code Manager has finished modifying the HTML page, the page is returned to the browser that made the request to be displayed.

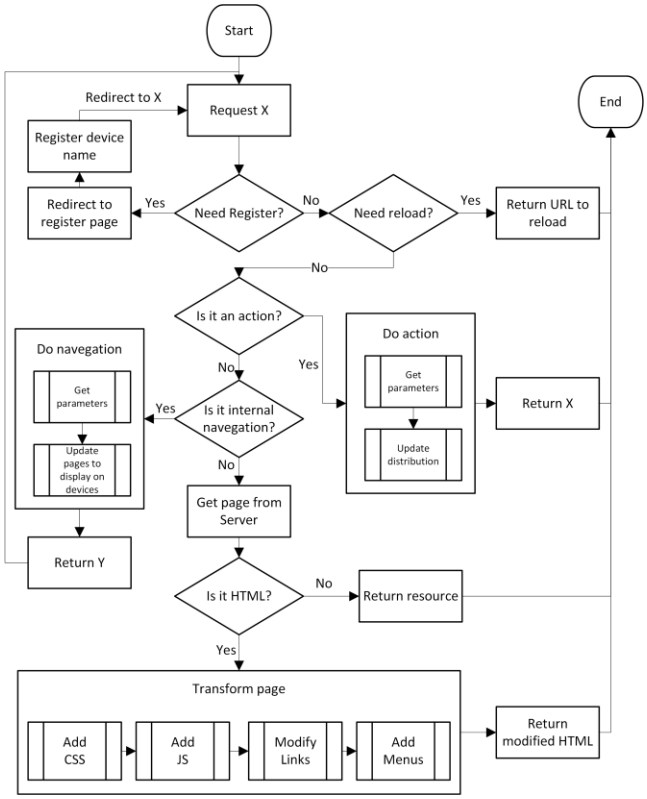

Figure 5 Diagram showing the workflow Proxywork

Figure 6 Page to register device

3.7 Distribution links and menus

This section explains more about the functionality of the Links Manager and Distribution Manager modules.

The Links Manager module finds all labels HTML <a> and modifies its *href* attribute with the following pattern:

http://nav-dui.com?Div=[***id_div***]&URL=[***previous_url***]

The URL "http://nav-dui.com" is used only for the Framework can detect this pattern and recognize it is an internal navigation of the page.

For example, a tag <a> as shown below that it is within an element identified by the Framework with id = "9":

```
<a href="www.uclm.es/english/studies.asp">
    Studies
</a>
```

It would be modified in the following way:

```
<a href="http://nav-dui.com?Div=9&URL=www.uclm.es/english/studies.asp">
    Studies
</a>
```

The Distribution Manager module inserts, for each element indicated by the Manager Granularity module, a menu with the distribution actions (Display/Hide, Copy and Distribute) and for each action, if applicable, a list of devices on which the action may impact.

The following is an example of transformation of an element . This element is shown in Figure 7 and the result is shown in Figure 8.

In this example there are two devices connected to the distribution environment (“MyiPhone” and “Nexus10”). The element "Cuerpo_Menu_Dch" contains a banner that displays an image in a Web Application. During the transformation, *Proxywork* inserts code to element . This code allows you to display a menu on the element with the permitted actions. The *onmouseover* event will make visible the distribution menu when mouse passes over the element and the *onmouseout* event will make hide the menu when the mouse leaves the element.

```
<div id="Cuerpo_Menu_Dch">
  <a href="http://www.uclm.es/estudios/online.aspx" title="UCLM Online">
        <img src="/banner_UCLM-online.jpg" title="UCLM Online"/>
  </a>
</div>
```

Figure 7 Div element before beging transformed

It also inserts a new element containing the distribution menu with the list of actions (Display/Hide, Copy and Distribute). The remaining actions (Rename, Connect and Disconnect) are not associated with the interface elements and is therefore not included here. For Copy and Distribute actions *Proxywork* adds a nested list with connected devices.

Actions are represented with tags <a> where the *href* property contents a special URL and it has the following pattern:

http://action-dui.com?Div=[***id_div***]&Action=[***id_action***]&IP=[***ip_device***]

The URL "http://action-dui.com" is used only for the Framework can detect this pattern and recognize it is an action.

The *id_div* parameter is the identifier of the menu assigned by the Framework, for our example is 42. The *id_action* parameter is the identifier of the action. And the *ip_device* parameter is the IP address of the device which the action will affect.

In Figure 8 we can also see how the banner *href* property has been transformed by the Links Manager module.

```
<div id="Cuerpo_Menu_Dch" style="visibility:visible"
    onmouseover="document.getElementById('menu_42').style.visibility='visible'"
    onmouseout="document.getElementById('menu_42').style.visibility='hidden'">
    <div id='menu_42'>
        <ul>
            <li class='duiclassundisplay'><a>Undisplay</a></li>
            <li class='duiclasscopy'><a>Copy</a>
                <ul><li>
                    <a href='http://action-dui.com?Div=42&Action=Copy&IP=192.168.1.11'>
                    MyiPhone
                    </a>
                    <a href='http://action-dui.com?Div=42&Action=Copy&IP=192.168.1.12'>
                    Nexus10
                    </a>
                </li></ul>
            </li>
            <li class='duiclassdistribute'><a>Distribute</a>
                <ul><li>
                    <a href='http://action-dui.com?Div=42&Action=Distribute&IP=192.168.1.11'>
                    MyiPhone
                    </a>
                    <a href='http://action-dui.com?Div=42&Action=Distribute&IP=192.168.1.12'>
                    Nexus10
                    </a>
                </li></ul>
            </li>
        </ul>
    </div>
    <a href="http://navega.dui.com?Div=42&URL=http://www.uclm.es/online.aspx" title="UCLM Online">
        <img src="/banner_UCLM-online.jpg" title="UCLM Online"/>
    </a>
</div>
```

Figure 8 Div element with distribution menu

3.8 Final examples

This section takes up again the case studies presented in section 3.1. We will see how it applies the proposed Framework in these case studies to solve the problems.

3.8.1 Distributed navigation menu

The first case study describes a very common scenario. Imagine a Web Application either, usually has a bar or a navigation menu that allows us to navigate between the different sections of the application. In this case study raises the possibility of distributing this navigation bar from a computer to a Smartphone. In this way, the navigation can be performed through the Web Application using your Smartphone, as if it was a remote control.

This case study has been implemented with two devices. A laptop Dell XPS M1530 with Windows 8 operating system and Google Chrome browser. And a smartphone Nokia Lumia 900 with Windows Phone 8 and Internet Explorer browser. Both devices configured with the IP address and port of the proxy where the Framework Proxywork is deployed.

From laptop we request the UCLM Web Application (URL is "www.uclm.es") and to be the first web request will display a page to register the device. It is necessary to enter a name that identify the device in all distribution actions. The name we assign is "DellXPS", then the browser will redirect to the requested page.

Similarly in the smartphone we access the UCLM web application and register the device with the name "Lumia".

When devices we use in the distribution are registered, context menus appear on each div element when the mouse cursor passes over them. These menus contain the distribution actions can be made on each element. In the laptop will appear the actions and together with them the devices list, in this case only the "Lumia" device (see Figure 9 - A).

In this example we distribute the navigation menu on the left side of the Web Application displayed on the Dell device. So we put the mouse over the navigation menu and an actions menu will appear with an action to distribute (see Figure 9 - A). Clicking on it displays a list of devices to which we can distribute and we click on "Lumia". The navigation menu of the Web Application automatically disappears and appears instantly in the Smartphone browser (see Figure 9 - C). Thus, the Chrome browser shows the UCLM Web Application completely but without the navigation menu on the left (see Figure 9 - B) and the Internet Explorer browser displays only the navigation menu which had been distributed (see Figure 9 - C).

When you click on any link in the navigation menu shown in smartphone, the navigation takes place in the laptop and not on the smartphone (see Figure 9 - D).

Note that if a third device (e.g. another laptop) is registered in the distribution environment and the same navigation menu is distributed from the new device to the smartphone, clicking on a link in the smartphone the navigation would be on both laptops simultaneously.

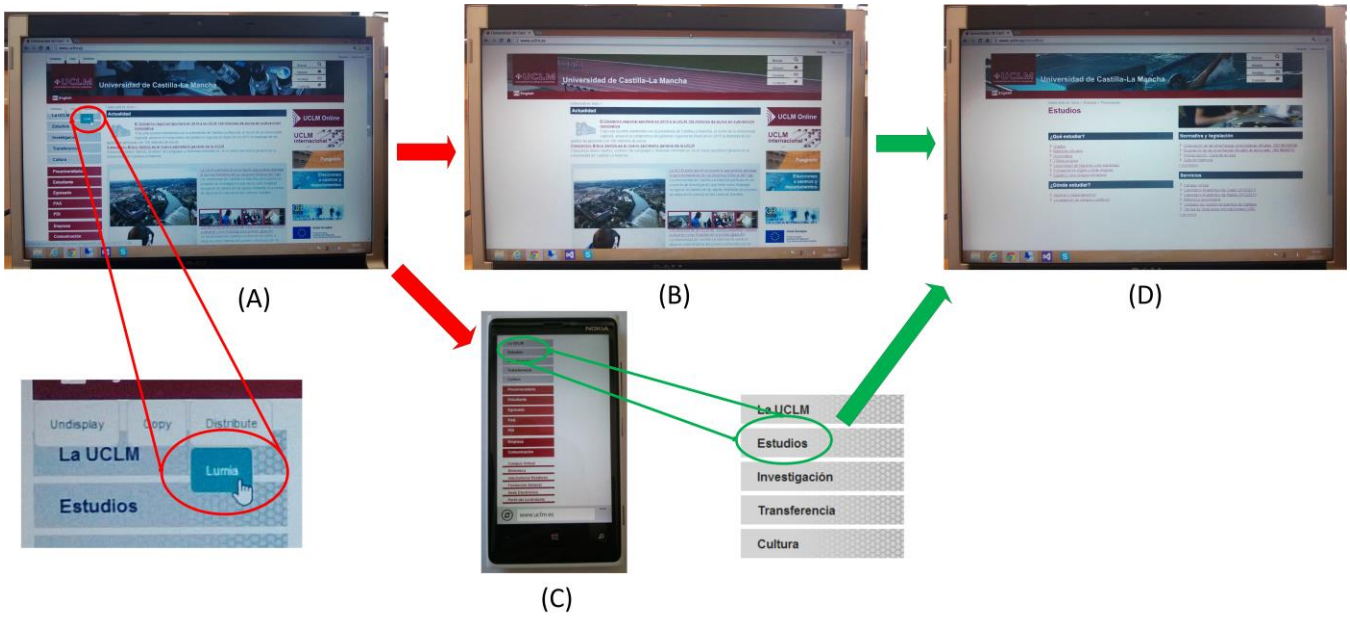

Figure 9 Distributed navigation menu case study implemented with *Proxywork*

3.8.2 Distributed news reader

The second case study is distributed news reader. As discussed in section 3.1, this case study can be found when visiting a Web Application that contains news. If we are viewing the Web Application with a small screen smartphone it can be sometimes tedious to read an extensive news. In this case study we will show how thanks to the proposed Framework we can distribute news that we are reading in a device such as a smartphone to a device with a larger screen such as a Tablet.

This case study has been implemented with two devices. A Nokia Lumia 900 smartphone with Windows Phone 8 and Internet Explorer browser. And a Nexus-10 Tablet with operating system Android 4.2 and Google Chrome browser. Both devices configured with the IP address and port of the proxy where the Framework *Proxywork* is deployed.

As in the previous case study, from the smartphone we request for news of UCLM web application (the URL is "www.uclm.es/gabinete/noticias.asp"). If it is the first web request, it will show a page to register the device. The name that we assign is "Lumia", then the browser we redirect to the requested page.

Similarly in the Tablet we access the same Web Application and register the device with the name "Nexus10".

Once registered both devices, if we access news from smartphone appears the actions menu of available distribution, and associated with each action, if required, a list of possible target devices. In our case will be shown only the "Nexus10" device (see Figure 10 - A).

Clicking on the "Nexus10" option from Copy action, automatically the news will appear in the Tablet browser (see Figure 10 - C). Besides, the news also follow showing in the smartphone browser (see Figure 10 - B).

In this way the news has appeared on another device with a much larger screen, facilitating its reading and improving the comfort of the user.

4. CONCLUSION AND FURURE WORK

This work presents a Framework called *Proxywork*. This Framework has been implemented in the form of proxy that transforms any Web Application developed with the standard Web languages (XHTML, CSS, etc.) and JavaScript into a Web Application with distributable user interface. The transformation is done in runtime, when users request the Web Application.

Because the *Proxywork* is for Web Application, it is supported by all platforms and is not dependent on a native application to distribute application´s elements.

The Framework takes into account the granularity. It takes the default tag like element that establishes the granularity because tags are usually used to group elements that have a common functionality or a common goal. Besides, the setting allow anytime establish other tags to make more flexible the granularity.

Our proposal implements seven actions related with the distribution. These actions are Connect, Disconnect, Rename, Display, Hide, Copy and Distribute.

We could list countless case studies to use *Proxywork* which we propose, but we will only present two case studies to show the power of the Framework.

On the one hand, distributed navigation menu case study presents the possibility of distributing a navigation bar from a computer to a Smartphone. On the other hand, distributed news reader case study shows how thanks to the proposed Framework we can distribute news that we are reading in a device such as a smartphone to a device with a larger screen such as a Tablet.

We are working on a new version of the environment that implements new distribution actions such as Clone, Replace, Merge, Switch and some more.

Further future work will be dedicated to allow that interface elements can be adapted to target device when they are distributed.

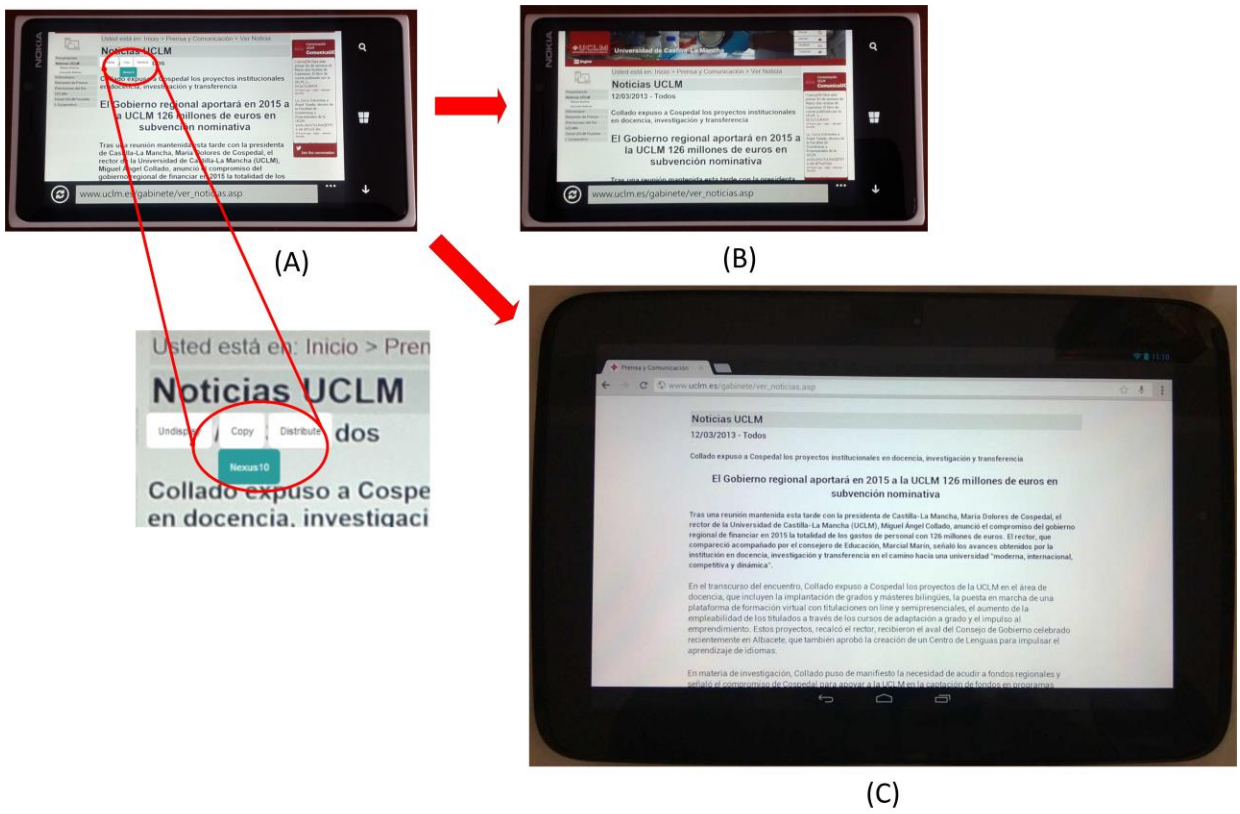

Figure 10 Distributed news reader case study implemented with *Proxywork*

5. 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).

6. REFERENCES

[1] Bandelloni, R. and Paternò, F. Migratory user interfaces able to adapt to various interaction platforms. Int. J. Human Computer Studies 60, 5-6 (2004), pp. 621-639.

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

[3] Bharat, K.A. and Cardelli, L. 1995. Migratory Applications Distributed User Interfaces. In Proc. of UIST'95 (Pittsburgh, Nov. 1995), ACM Press, New York, pp. 132-142.

[4] Fardoun, H. M., Paules, A., and Alghazzawi, D. M. CSchool: DUI for Educational System using Clouds. Proceedings of the 2nd Workshop on Distributed User Interfaces: Collaboration and Usability, DUI 2012. ISBN 978-84-695-3318-5, pp 35-38. May 5th, 2012. Austin, Texas, USA.

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

[6] González, P., Gallud, J.A., Tesoriero, R. WallShare:A Collaborative Multi-pointer System for Portable Devices. PPD10: Workshop on coupled display visual interfaces. May 25, 2010, Rome, Italy.

[7] 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.

[8] Izadi S, Brignull H, Rodden T, Rogers Y, Underwood M (2003): Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media. Proc. 16th ACM UIST '03. ACM, New York, pp 159.

[9] 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.

[10] 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.

[11] MacIntyre, B., Mynatt, E.D., Voida, S., Hansen, K.M., Tullio, J., & Corso, G.M. (2001). Support for Multitasking and Background Awareness using Interactive Peripheral Displays. Proceedings of UIST 2001, 41-50.

[12] 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.

[13] 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.

[14] Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., Fuchs, H. (1998). The Office of the Future: A Unified Approach to Image-based Modeling and Spatially Immersive Displays. Proceedings of SIGGRAPH 1998, 179-188.

[15] Rekimoto, J., Saitoh, M. (1999). Augmented Surfaces: A Spatially Continuous Workspace for Hybrid Computing Environments. Proceedings of CHI 1999, 378-385.

[16] Schank, P., & Dwyer, N. (2008). ScribbleProv: Supporting Disciplined Improvisation During Face-to-Face Discussion, First Year Project Report (NSF-IIS #0713711), Menlo Park, CA: SRI International.

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

[18] Streitz, N.A, Geibler, J., Holmer, T., Konomi, S., Muller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., & Steinmatz, R. (1999). i-LAND: An Interactive Landscape for Creativity and Innovation. Proceedings of CHI 1999, 120-127.

[19] Tan, D.S., Stefanucci, J., Proffitt, D., Pausch, R. (2001). The Infocockpit: Providing Location and Place to Aid Human Memory. Workshop on Perceptive User Interfaces 2001.

[20] Tandler, P., Prante, Th., Müller-Tomfelde, Th. Streitz, N. and Steinmetz. R. ConnecTables: Dynamic coupling of displays for the flexible creation of share workspaces, Proc. of 14th ACM Symp. on UI Software and Tech. UIST'01, ACM Press, New York, 2001, pp.11–20.

[21] Vandervelpen, Ch., Vanderhulst, G., Luyten, K., and Coninx, K. 2005. Light-Weight Distributed Web Interfaces: reparing the Web for Heterogeneous Environments. In Proc. of ICWE 2005, pp. 197-202.

[22] Wigdor D, Jiang H, Forlines C, Borkin M, Shen C. The WeSpace: The Design, Development and Deployment of a Walk-Up and Share Multi-Surface Visual Collaboration System. ACM CHI. 2009.