A Taxonomy for Distributed User Interfaces: Examples and Consequences

Pedro González Villanueva[1], **Ricardo Tesoriero**[1], **José A. Gallud Lázaro**[1]

[1]*University of Castilla-La Mancha, Escuela Superior de Ingeniería Informática de Albacete, Campus Universitario s/n, 02071, Albacete, Spain,*

Running Head: *A Taxonomy for Distributed User Interfaces*

Corresponding Author's Email address: Pedro González Villanueva (pedro.gonzalez@uclm.es)

Brief Authors' Biographies:

Pedro González Villanueva is a degreed with an interest in Distributed User Interfaces, Human-Computer Interaction and Quality in Use Models; he is a PhD Student in the Department of Computer Systems at University of Castilla-La Mancha.

Ricardo Tesoriero is a PhD in Computer Science with an interest in Human-computer interaction and Model-driven development; he is an assistant professor in the ISE Research group of the Computer System Department at the University of Castilla-La Mancha.

José A. Gallud is a PhD in Computer Science with an interest in Human-computer interaction and Distributed User Interfaces; he is an associate professor in the ISE Research group of the Computer System Department at the University of Castilla-La Mancha.

ABSTRACT

The appearance of a new generation of user interfaces (UI) which are capable of taking advantage of the diverse and growing ecosystem of interconnected screens give rise to a new distribution paradigm of UIs which are known as Distributed User Interfaces (DUI). This new paradigm is revolutionizing the way user interfaces are created, designed, and used, considering that DUIs give the user the capability to divide the user interface and distribute it among dynamically different devices, thus being capable of completing a task as if it were a traditional UI. This article studies the concept of DUI in depth, while improving previous definitions and obtaining both the theoretical and practical consequences of the new concepts of Divisible User Interface, Distributed User Interface, and Distributable User Interfaces. Based on these concepts, a taxonomy can be presented which will allow the classification of a variety of different proposals, both present and future, thereby establishing a formal base which will be the foundation of all future developments in this field.

CONTENTS

1. INTRODUCTION

Interactive screens are becoming widespread all the time, significantly replacing more traditional methods which present information to the public in a more stationary way. The use of interactive screens is constantly increasing, partly because of the flexibility which digital screens offer, given the possibility of dynamic and remote updates of the presented information, and also due partly to the significant cost reduction of the aforementioned devices. This means that nowadays we are constantly immersed in an ecosystem of screens.

The collaboration scenarios, in which various users do tasks at the same time, have an increasing influence on our everyday lives and have been present for many years. In addition, the applications of these scenarios require the content to be more dynamic and the users themselves to be the content creators. That way, applications can be created which allow the users to share information, interact, and do tasks at the same time. One clear example of this type of scenario is Web 2.0 (O´Reilly, 2005).

Due to the two previously explained tendencies, the increase of interactive screens and collaborative scenarios, we are getting more and more demanding with our user interfaces (henceforth referred to as UI) and we also increasingly need to use the concept of Distributed User Interfaces (henceforth referred to DUI).

For these reasons, the DUIs are becoming much more prominent everyday and therefore it is necessary to define the concept of DUI in detail and clearly establish the limits which allow a UI to be differentiated from a DUI. There are many current definitions of the what the concept of a DUI will be like in the future, but we lack a more detailed definition.

2. RELATED WORK

There are many definitions of DUI which can be found in the biography. Those which are considered most important are detailed next.

Melchior, Grolaux, Vanderdonckt, and Van (2009) claim that “a Distributed User Interface consists of a user interface with the capacity to distribute all or part of its components among various monitors, devices, platforms, and/or users.”

Authors such as Luyten, and Coninx (2005) and Vandervelpen, Vanderhulst, Luyten, and Coninx (2005) claim that “a Distributed User Interface can be divided in parts which migrate to different devices around the final user, and which cooperate to make the user´s tasks easier. It is essential to take into account that migration is an essential property of interface distribution.”

Berglund, and Bang (2002) also affirm that “a Distributed User Interface is a user interface which distributes its components among various interactive devices in its environment.”

Finally, López, Gallud, Lazcorreta, Peñalver, and Botella (2011) conclude that "a Distributed User Interface is the collection of interaction elements that forms a set of User Interfaces, that is to say, a set of elements with a common functionality. These elements are distributed over a set of platforms without losing the common functionality given by the user' tasks. It is also possible to consider a group of users instead of a single user. "

It should also be noted that some authors such as Berti, Paterno, and Santoro (2006) refer to DUI as Migratory User Interfaces.

As we can conclude from these definitions, the limit which differentiates a UI from a DUI is not completely clear. For this reason, the main objective of this paper is to further explore the concept of DUI.

From the point of view of the taxonomies which allow a classification of UIs, in the bibliography we can find some which we will comment on next.

Berti, Paterno, and Santoro (2006) propose a taxonomy which classifies all Migratory User Interfaces according to: activation type (on demand and automatic migration), migration type (total, partial, distributed, incorporating, and multiple), combination of migratory methods (single-method, trans-method, multiple methods), type of active interfaces (precomputed user interfaces and runtime generation of user interfaces, etc), etc. In this taxonomy proposal, Berti, Paterno, and Santoro only take into account the concept of migration, without considering concepts like the division and distribution of the design time and runtime.

3. BEYOND DISTRIBUTED USER INTERFACES

In the previous section, some definitions were presented which can be found in previous papers and which we have used as a starting point for our own research. Among all the revised literature, the studies of Melchior, Grolaux, Vanderdonckt, and Van (2009), Vandervelpen, Vanderhulst, Luyten and Coninx (2005), and López, Gallud, Lazcorreta, Peñalver, and Botella (2011) stand out because all of them present the concept of DUI and its most important properties.

The only formal definition of the concept of DUI can be found in the research of López, Gallud, Lazcorreta, Peñalver, and Botella (2011), for whom the only DUI is the collection of UIs that are run on a collection of platforms and which have the common goal of carrying out the user´s tasks. This definition means a step forward in the formalization of the concept of DUI but it is not enough, considering that it doesn´t cover all the possible distribution configurations and fails to answer the following questions:

- How can we distinguish the case of a UI that can divide, distribute, and run on a collection of platforms from an application that has a UI which was previously divided and distributed in the design time?

- What happens if all the platforms are really the same? In that case we would have an application with a UI which can break down into various subUIs although all of them operate on the same platform.

- Are the results the same or not for a UI which can divide itself in runtime?

- Is a divisible UI a DUI?

- How many distinct types of DUIs exist?

- What is understood by “platform”?

- Is the Web a platform?

It can be demonstrated that it is not possible to respond to these questions with the definitions that can be found in the cited works, and therefore previous definitions of DUIs cannot be used to cover the wide spectrum of configurations that gives way to the division and distribution of UIs.

3.1. New characteristics of the UIs within the distribution paradigm

In Figure 1, the new universe of the possible combinations of user interfaces is presented, which will give way to a new distribution paradigm. This new characterization presents the following types of combinations:

- Undivided UI

- Divided UI or DivUI

- Distributed UI or DUI

- Distributable UI or DeUI

- Divisible UI or DiveUI

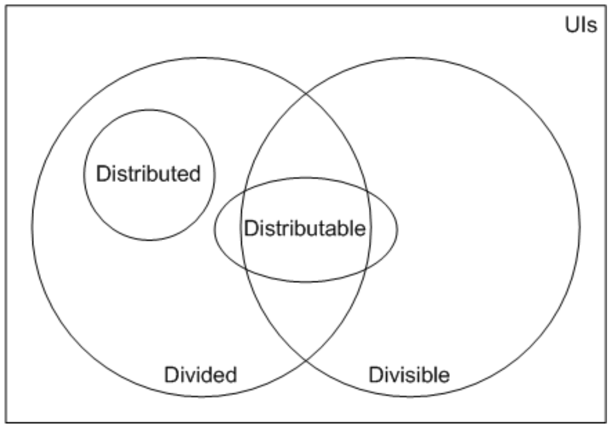

Figure 1. The new UI universe which leads to the distribution paradigm

In the following paragraphs, we are going to textually define each of the combinations which are present in Figure 1. The next section will formalize these textual descriptions through a metamodel.

The first type of application is that which has an undivided UI. This regards the most simple application case in which the UI is not divided into independent elements in the design time and, obviously, it also is not possible to divide it during the runtime. One example of this type of application is the calculator.

An application with a divided UI could be considered the most frequent case in the majority of platforms. This has to do with those applications which have a UI with floating elements or elements which are independent from the main window. We could say that this UI is divided in design time and generally it isn´t possible to separate an element from the UI during the runtime (unless the separation was previously planned in the design time). One example of this type of application is Paint or something similar.

A distributed UI or a DUI is a UI is divided and distributed in a stationary way during the design period and uses more than one different platform. Recent research in this field has referred to all classes of a UI system that is distributed as DUI without taking into account the more complex or different configurations which are explained in this article. One example of this type of application is Wallshare (Villanueva, Tesoriero, & Gallud, 2010).

A distributable UI (DeUI) is a UI that is divided that can be distributed during the runtime on a valid combination of platforms. Unlike DUIs, DeUIs have the capacity to, if the user wants to, be carried out on a combination of platforms. One example is Vandervelpen´s application.

A divisible UI is a UI that can divide during its runtime into elements that weren´t foreseen during the design period. There are no known examples of this kind although hopefully one will appear in the near future.

3.2. The matter of the Platform

It is important to make clear what we mean by Platform, a key element in the conceptualization of Distributed User Interfaces (DUIs). Their importance can be seen in recent publications about DUIs, as in Lopez´s research and Vanderdonckt´s article, to point out two relevant reports.

The articles by Lopez et al. about DUI define four fundamental properties. Curiously, the first is Portability, which the authors define as “The user interface (as a whole) or elements of the interface which can be transferred between platforms and devices (…)” As we can see, there is a reference to the platform, although they do not explain exactly what is understood by this term.

Vanderdonckt´s report also gives a great deal of importance to the concept of platform when they define the analogous term “Multi-platform use” as “ a single user uses heterogeneous computing platforms, perhaps running different operating systems." In this paper, a clear idea is provided of what is meant by the term Platform: Heterogeneous machines which run operating systems which could be different.

We must clearly define the term Platform due to the necessity of studying the implications that the platform has on the distribution of the interface elements. The distribution will be more or less complex or simple depending on what is understood by Platform. The interaction between the users and the distribution support which can be offered by the DUI will depend, in large part, on what is understood by platform.

Therefore, and along the lines proposed by Vanderdonckt et al., in this paper we will refer to the hardware as a combination of hardware and software formed by the computing system (PC, mobile device, etc) and the operating system which runs it. Accordingly, for example, two units which have equivalent hardware and run the same operating system, we would consider them to be two distinct platforms. In the same way, we would also consider two units with different operating systems to be two distinct platforms even if the hardware was identical. Along the same lines, two virtual machines running on the same unit we would also consider to be two platforms.

One concrete case is that of Web applications. Continuing with the outline that we have used so far, we would consider the navigator to be a virtual machine, which means that an application which is run on a different platform would be considered as running on a different platform from the machine that is running the navigator.

Once we have clarified the concept of platform, in this paper we will also consider the concept of a platform during the runtime (RunTimePlatform) as the entity which represents the capacity of a UI to run on a specific hardware and software.

3.3. Metamodel of the distribution paradigm

In this section we present the metamodel of the new distribution paradigm (see Figure 2) with which it is possible to represent the different combinations of UIs which were mentioned in the previous section.

Our proposal of a distribution metamodel is an extension of the metamodel of abstract UIs (AUIs) in UsiXML (Limbourg, Vanderdonckt, Michotte, Bouillon, & Lopez, 2004) which is based on the Cameleon Framework.

The entity UISystem is the central element of the proposed metamodel. The interactive applications will have a system of User Interfaces which will be represented through each entity. One UISystem can be implemented on more than one platform y is associated with a least one RunTime platform in order to represent the fact that one particular UI is running on a RunTimePlatform.

Just like DivUIs, both DUIs and DeUIs are based on the concept of UI.

The concept of UI which we are using has an Interaction Object (hereafter referred to as IO) as its central element, which is largely equivalent to the concept Abstract

Interaction Object (AIO). We have defined IO as an element with which the user can interact with the system and can be either container or component. The Container element is equivalent to the Abstract Container as defined by UsiXML, and the Component element is equivalent to the Abstract Individual Component, except that our Component element is “composed” by Facets (control of entrances and exits) whereas the metamodel of AUI in UsiXML establishes this as a hierarchy. One container could have other containers, with a repetitive pattern, and other Component elements. An IO also has a collection of ECA rules, that is, Event-Condition-Action rules that define the behavior of the UI.

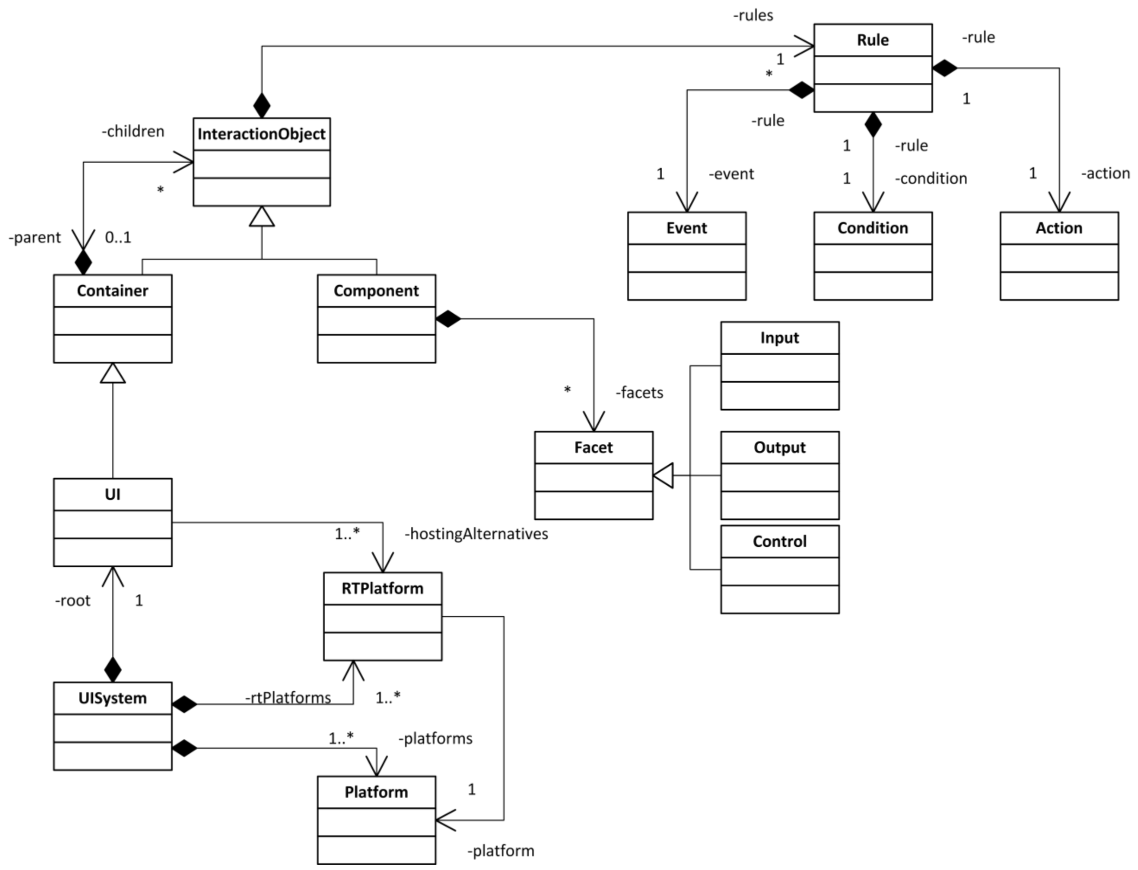

Figure 2. Metamodel of the new UI distribution paradigm

An UI system, whatever type it may be (indivisible UI, DivUI, DUI or DeUI), has an initial UI or root. This UI root, linked as previously explained, can be as complex as the hierarchy in the control and container tree will allow.

We can formally define the UI root through an OCL invariant:

```
context: UISystem
inv: self.root.parent.oclIsUndefined()
```

This invariant defines that for a UI to be a root, it must not be contained in any other UI, that is, it must not have a father.

It would be best to explain now one important aspect in the design of the UIs of the applications which affect the distribution paradigm. Some of the containers of the hierarchy tree which the UI is structured on can also be UIs (sub user interfaces or subUI) and, as we will see shortly, can be distributed. This characterization by which some containers are also considered to be UIs can take place during the design time (which produces divided UIs, DivUIs) or in runtime (which produces divisible UIs, DiveUIs).

Formally, the invariant in OCL for divided UIs would be the following:

```
context: UISystem
inv: UI.allInstances().size() > 1
```

This invariant requires the UI system to have more than one UI, in other words, if the UI is made up of containers and interactive objects, at least one of those containers must be defined as a UI. An application like Paint which has a principal window (UI root) that has floating windows for the color palette or the tools would meet this requirement.

As previously explained, the distinction between DUIs and DeUis has to do with the RunTimePlatform. In the case of the DUIs, the UI root must be made up of more than one UI (at least one of the containers must be classified as a UI), in which each UI has a runtime platform associated with it and at least two runtime platforms. These conditions written in OCL lead to the following invariant:

```
context: UISystem
UI.allInstances().size() > 1 and
UI.allInstances() -> forAll(ui:UI | ui.hostingAlternatives.size()
= 1) and
UI.allInstances() ->
collect(hostingAlternatives).flatten().asSet().size() > 1
```

At least two platforms are required because an application can have its principal UI divided into a combination of UIs, and even if each UI has a runtime platform associated with it, for us to consider it to be a DUI there must be at least two platforms, otherwise it would be considered a divided UI (DivUI).

As far as Distributable User Interfaces (DeUI), we have the same conditions which apply to DUIs except in the case of runtime platforms which are associated with each UI, at least one of them must be associated with more than one runtime platform, which means that there is a possibility that this particular UI can be distributed on more than one different platform. The OCL invariant would be expressed as following:

```
context: UISystem
UI.allInstances().size() > 1 and
UI.allInstances() -> exists (ui:UI | ui.hostingAlternatives() > 1)
and
UI.allInstances()->
collect(hostingAlternatives).flatten().asSet().size() > 1
```

In this section, the new concepts of DivUI, DUI, DiveUI and DeUI have been formalized as those which extend the distribution capacity of user interfaces.

4. THE NEW DISTRIBUTION TAXONOMY FOR UIS: EXAMPLES AND DISCUSSION

In Figure 3, various examples can be seen which represent previous publications about DUIs which were classified according to the new taxonomy. In this section, we will justify the classification of each of the components into one of the combinations presented in this paper: DivUI, DUI, DeUI, and DiveUI.

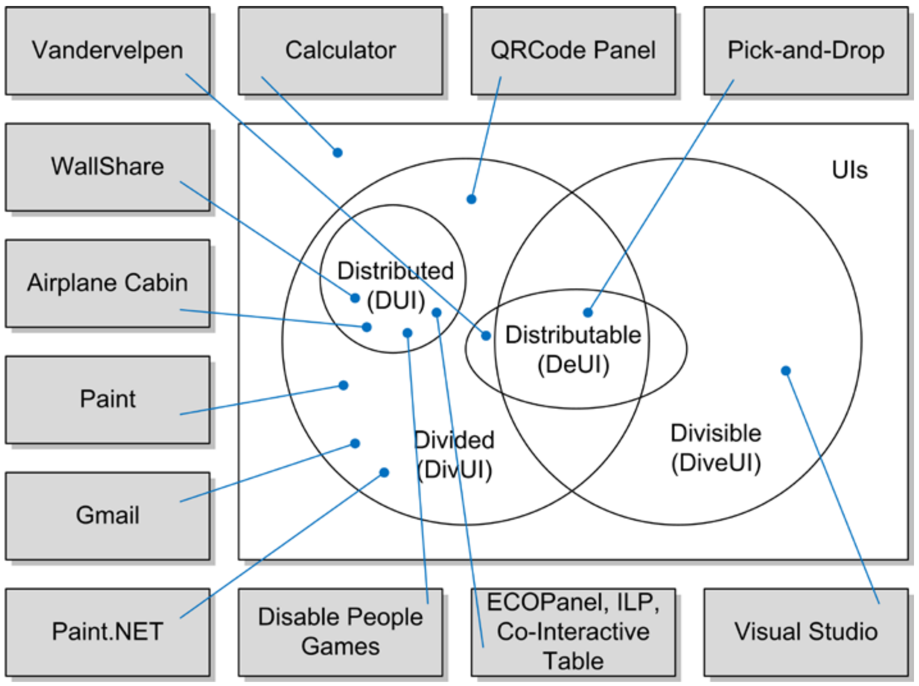

Figure 3. Examples classified according to the new taxonomy

The Calculator application, as we have already mentioned, is a paradigmatic example of an application which has a UI which is not divided, nor can it be divided. This represents the simplest applications which are present in any framework (MacOS, Windows, Linux).

As examples of applications which have a UI that divides in the design time, we have QRCode Panel, Paint, Paint.NET (http://www.getpaint.net) and Gmail (http://www.gmail.com). The application Paint is considered to be divided within the taxonomy since it cannot be divided in runtime; in addition, it´s parts can only be run on the same platform. For the same reason, the applications QRCode Panel, Paint.NET and Gmail are considered to be divided.

Some examples of a distributed UI are an Airplane Cabin, WallShare (Villanueva, Tesoriero, & Gallud, 2010), ECOPanel (Tesoriero, Tébar, Gallud, Penichet, & Lozano, 2008), ILP, Co-Interactive Table (Guía, Lozano, Gallud, Tesoriero, & Penichet, 2010) and Disable People Games. The complex control system in an airplane cabin is considered to be within distributed User Interfaces because it is made up of numerous parts which are hosted in different interaction spaces and all of them are carried out in order to realize a common goal. Each of the parts is distributed initially and not during the runtime. The WallShare system is considered to be distributed since each of the UIs that form the whole are hosted in different interactive spaces. For example, the shared zone can be run on a PC which is connected to a projector, the WallShareMovil on a mobile device and WallShareDesktop on a PC. In addition, the UI of WallShareDesktop is divisible in and of itself, which is why WallShare is classified between Distributed and Divisible.

The example of a web application which is presented by Vandervelpen is within the category of distributable and divided. It is considered distributable because parts of the interface itself can be moved to different interactive spaces. For example, the interface in charge of Zoom is distributed to a PDA and the interface in charge of Scroll is distributed to another PDA, both from a PC. Furthermore, it is considered to be divided because the parts are defined within the design time and not during the runtime.

The Visual Studio is an application which is considered divisible since we can break down the interface itself into other interfaces, all of which are hosted in the same interactive space. One example of this is division is the editing area, a component of the interface which by default forms part of the main interface but can separate itself into another new interface.

Finally, the Pick-and-Drop system is a distributable, divided and divisible UI due to properties showed by Rekimoto (1997).

5. CONCLUSIONS AND FUTURE WORK

Due to the increase of interactive screens and collaborative scenarios, users are more demanding on the functionality provided by user interfaces leading to the development of the concept of Distributed User Interfaces.

Thus, this work is focused on the definition and classification of Distributed User Interfaces establishing well-defined boundaries among different kinds of multi-display user interfaces.

Therefore, the article proposes a new classification of user interfaces that goes beyond of the traditional definition of DUIs. In this way, the article introduces the concept of Divided UI, Divisible UI and Distributable UI.

The definition of the taxonomy is based on a formal metamodel to represent the different sets of UIs. MOF enriched with OCL constraints was used to establish the formal definition of the taxonomy metamodel.

As result of the analysis of different user interfaces according to this new taxonomy, we have concluded that the proposal provides UI designers and developers with a formal basis to work with. Besides, the analyzed examples show the benefits of using this taxonomy on actual and future user interface.

As future work, we are working on a framework for building Divided, Divisible, Distributed and Distributable UI applications taking advantage of the new concepts that were defined on this paper.

NOTES

Acknowledgments. This research has been partially supported by the Spanish MEC/MICINN CICYT TIN2011-27767-C02-01 project and the regional projects with reference PPII10-0300-4174 and PII2C09-0185-1030-1.

Authors' Present Addresses. Pedro González Villanueva, University of Castilla-La Mancha, Escuela Superior de Ingeniería Informática, Campus Universitario s/n, 02071, Albacete Spain. Email: Pedro.Gonzalez@uclm.es. Ricardo Tesoriero, University of Castilla-La Mancha, Escuela Superior de Ingeniería Informática, Campus Universitario s/n, 02071, Albacete Spain. Email: ricardo.tesoriero@uclm.es. José A. Gallud Lázaro, University of Castilla-La Mancha, Escuela Superior de Ingeniería Informática, Campus Universitario s/n, 02071, Albacete Spain. Email: Jose.Gallud@uclm.es.

HCI Editorial Record. Xxx xxx xxx
Supplied by the Editor.

REFERENCES

Berglund, E., & Bang, M. (2002). Requirements for distributed user-interfaces in ubiquitous computing networks. *Proceedings of MUM2002*, Oulo, Finland, December 11-13.

Berti, S., Paternó, F., & Santoro, C. (2006). A Taxonomy for Migratory User Interfaces. In: Gilroy, S.W., Harrison, M.D. (eds.) DSV-IS 2005. LNCS, vol. 3941, pp. 149–160. Springer, Heidelberg.

Guía, E.D.L., Lozano, M.D., Gallud, J.A., Tesoriero, R., & Penichet, V.M.R. (2010). Co-Interactive Table: a New Facility to Improve Collaborative Meetings. *MobileHCI 2010*. Publisher ACM. September 7th-10th, 2010. Lisbon, Portugal.

Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., & Lopez, V. (2004). UsiXML: a Language Supporting Multi-Path Development of User Interfaces. *Proceedings of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004* (Hamburg, July 11-13). Lecture Notes in Computer Science, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 200-220.

López, J.J., Gallud, J.A., Lazcorreta, E., Peñalver, A., & Botella, F. (2011). Distributed User Interfaces: Specification of Essential Properties. *Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem*. Springer. ISBN 978-1-4471-2270-8. Chapter 2, pp 13-21.

Luyten, K. & Coninx, K. (2005). Distributed User Interface Elements to support Smart Interaction Spaces. *Proceeding of the 7th IEEE Int. Symposium on Multimedia, IEEE Comp. Society*, Washington, DC, pp. 277-286.

Melchior, J., Grolaux, D., Vanderdonckt, J., & Van, R. P. (2009). A Toolkit for Peer-to-Peer Distributed User Interfaces: Concepts, Implementation, and Applications. *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*. Pittsburgh, PA, USA, pp 69-78.

O'Reilly, T. (2005) What Is Web 2.0? Design Patterns and Business Models for the Next Generation of Software.

Rekimoto, J. (1997). Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. *Proceeding Of UIST'97*. ACM Press, New York, pp. 31–39.

Tesoriero, R., Tébar, R., Gallud, J. A., Penichet, V. M. R., & Lozano, M. (2008). Interactive EcoPanels: Paneles Ecológicos Interactivos basados en RFID.

Proceedings of the IX Congreso Internacional de Interacción Persona-Ordenador Interacción 2008. ISBN: 978-84-691-3871-7; pp 155-165.

Vandervelpen, Ch., Vanderhulst, K., Luyten, K., & Coninx, K. (2005). Light-weight Distributed Web Interfaces: Preparing the Web for Heterogeneous Environments. *Proceeding of 5th Int. Conf. on Web Engineering. ICWE'2005*. Springer-Verlag, Berlin.

Villanueava, P. G., Tesoriero, R., & Gallud, J. A. (2010). Multipointer and collaborative system for mobile devices. *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, ACM Press, pp 435-438.

FOOTNOTES

Make a copy of all footnotes on a separate page here. This only has to be done for the final submission for production. During the review process, it is okay to just have footnotes at the bottom of pages.

1. xxx
2. xxx
3. xxx

FIGURE CAPTIONS

Make a copy of all figure captions on a separate page here. This only has to be done for the final submission for production. During the review process, it is okay to skip this page.

Figure 1. xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx

Figure 2. xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx

Figure 3. xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx

FIGURES

Include all figures here with one figure per page. This is only needed for the final submission. During the review process, it is better to embed the figures in the body of the paper for ease of reviewing.

Figure 1. **xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx**

<figure or table here>

Note that in HCI all tables are considered as Figures. That is, there is only one numbered sequence of all “out-of-text” items.

Figure 2. xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx xxx

<figure or table here>