4th Workshop on Distributed User Interfaces and Multimodal Interaction

ICWE 2014 Workshop
1st July 2014
Toulouse, France

The Association for Computing Machinery
2 Penn Plaza, Suite 701
New York New York 10121-0701

ACM ISBN: 978-1-60558-724-0

4th Workshop on Distributed User Interfaces and Multimodal Interaction
ICWE 2014 Workshop
1st July 2014
Toulouse, France

This Workshop is the fourth in a series on Distributed User Interfaces. On this occasion, the workshop is focused on DUIs and Multimodal interaction. The main goal is to join together people working in extending the Web and other user interfaces to allow multiple modes of interaction such as GUI, TUI, Speech, Vision, Pen, Gestures, Haptic interfaces, etc. Multimodal interaction poses the challenge of transforming the way we interact with applications, creating new paradigms for developers and end-users. Through active group discussion, participants will have the chance to share their knowledge and experience to advance in this field.

These articles were submitted to the 4th Workshop on Distributed User Interfaces and Multimodal Interaction at ICWE 2014 that was held on July 1st, 2014 in Toulouse, France.

Editors

Prof. Dr. María Dolores Lozano
University of Castilla-La Mancha
Computing Systems Department
Campus Universitario, s/n
02071, Albacete, Spain
maria.lozano@uclm.es

Prof. Dr. Jean Vanderdonckt
Université catholique de Louvain
Louvain School of Management
Place des Doyens, 1
B-1348, Louvain-la-Neuve, Belgium
jean.vanderdonckt@uclouvain.be

Prof. Dr. Habib M. Fardoun
King Abdulaziz University
Information Systems Department
Faculty of Computing and Information Technology
Jeddah, Saudi Arabia
hfardoun@kau.edu.sa

Prof. Dr. Ricardo Tesoriero
University of Castilla-La Mancha
Computing Systems Department
Campus Universitario, s/n
02071, Albacete, Spain
ricardo.tesoriero@uclm.es

Prof. Dr. José A. Gallud
University of Castilla-La Mancha
Computing Systems Department
Campus Universitario, s/n
02071, Albacete, Spain
jose.gallud@uclm.es

Prof. Dr. Víctor M. R. Penichet
University of Castilla-La Mancha
Computing Systems Department
Campus Universitario, s/n
02071, Albacete, Spain
victor.penichet@uclm.es

WORKSHOP PROGRAMME COMMITTEE

Jose Luis Garrido (Universidad de Granada, Spain)
Fabio Paterno (CNR-ISTI, Italy)
Carmen Santoro (CNR-ISTI, Italy)
Silvia Abrahao (Universidad Politécnica de Valencia, Spain)
Christopher Kolski (University of Valenciannes, France)
Gaelle Calvary (University of Grenoble, France)
Kris Luyten (Hasselt University, Belgium)
Toni Granollers (Universitat de Lleida, Spain)
Peter Forbrig (University of Rostock Germany)
Aaron Quingley (University of St. Andrews, Scotland)
Miguel Nacenta (University of St. Andrews, Scotland)

Cover designed by Gabriel Sebastián (Photo: Shutterstock)

Organizers

King Abdulaziz University

Belgian Lab. of
Computer-Human
Interaction

Université catholique
de Louvain UCL

Sponsors

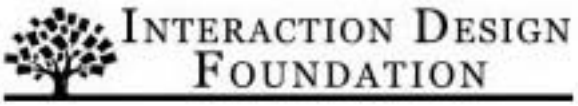

Table of Contents

Distributing User Interfaces

Ricardo Tesoriero
University of Castilla-La Mancha
Campus Universitario de Albacete
(02071) Albacete, Spain
ricardo.tesoriero@uclm.es

ABSTARCT
The distribution of user interfaces is a reality. To represent this reality this paper presents a metamodel to characterize user interface distribution capabilities and states. This metamodel allows analyzers/designers to manipulate user interface distribution models by the means of two model editors in order to calculate their capabilities and states. Based on these characteristics, five cases of study are analyzed and as result of this analysis, we redefine the distributed user interface concept as a user interface state, and define the distributable user interface concept as a user interface capability. Finally, we present the Proxywork system to illustrate the distributable user interface concept.

Author Keywords
Human-Computer Interaction; Distributed User Interfaces; Distributable User Interfaces; Web-based User interfaces.

ACM Classification Keywords
H.5.2 User Interfaces. *Theory and methods*; D.2.2 Design tool and techniques. *User Interfaces*;

INTRODUCTION

The popularity and diversity of devices that are available to users is rising. They can be classified as multi-purpose devices and specific devices. Among multi-purpose devices we have mobile devices and stationary devices.

On the one hand, Mobile devices such as laptops, tablets, smartphones and even Smartwatches are affordable and easy to acquire. They can be controlled in different ways according to the peripheral devices that are available on them. While, laptops employ keyboards, mouse and trackpads; Smartphones, tablets, Smartwatches, and so on, employ touchscreens, accelerometers, gyroscopes, GPSs, RFID, etc.

4th Workshop on Distributed User Interfaces and Multimodal Interaction
DUI'14, July 01 2014, Toulouse, France
Copyright 2014 ACM 978-1-60558-724-0/14/07$15.00
http://dx.doi.org/10.1145/2677356.2677669

On the other hand, stationary devices such as SmartTVs, projectors connected to desktop computers, game consoles and so on are controlled with novel interaction devices such as the Microsoft Kinect, the PlayStation Move jointly with the PlayStation Eye or the Wii-mote.

Specific purpose devices such as RFID-based panels, plane cockpits, advertisement panels, etc., are also interesting applications that are part of the interaction environment that is available to users.

In the beginning, all these devices were used separately. And they were unaware of the existence of other devices even if they were in the same interaction environment.

However, this tendency has changed in recent years and the concept of user interface ecosystem is being adopted gradually.

The concept of ecosystem of coupled displays was defined by Terrenghi in [1]. Under this configuration, a set of displays is connected/synchronized to each other to enrich users' experience.

In this paper, we present a conceptual model to characterize the distribution of user interfaces that are part of an ecosystem. Besides, we present a Web application that is capable of distributing components that are part of Web applications.

This work is organized as follows. Next section presents five user interface ecosystem scenarios to analyze the characteristics of user interfaces ecosystem characteristics. Afterwards, we propose a metamodel as well as a model editor to derive the ecosystem properties and possible states it may reach. Later, we present five models to illustrate user interface ecosystem characteristics. Then, we expose the Proxywork system as an example of distributable user interface where we present the implementation of a set of distribution primitives applied to Web application environments. Finally, we present conclusions and future work.

THE USER INTERFACE DISTRIBUTION DICHOTOMY

This section presents a set of five scenarios of user interface ecosystems. These scenarios show different distribution characteristics of the user interface.

The first example is about the use of a Smartwatch jointly with a Smartphone where the Smartwatch notifies the user

about an event (i.e. an email has arrived, an incoming phone call is in progress, a Skype call, etc.) Then, the user is able to retrieve relevant information regarding the event. For instance, if it is an email, s/he is able to retrieve the sender address, the subject, and first lines of the mail contents. As the capabilities if the Smartwatch display are limited, the user is not able to answer the email from the Smartwatch; therefore s/he employs the "See on device" action to synchronize the application user interface associated to the event on the Smartwatch (i.e. email client) with the version of the application running in the Smartphone automatically (i.e. activating the mail client user interface and opening the mail that is being observed on the Smartwatch).

The coupled user interface concept is not only applicable on mobile devices. This concept is also valid on stationary displays and even on mixed scenarios. For instance, the Samsung AllShare[1] service allows users to distribute information among different devices that are compatible with the service. Through this service users are able to distribute photos, videos, music etc. among PCs, TVs, mobile phones, tablets, digital cameras, etc. by the means of a wireless network.

Previous scenarios describe two User Interface Ecosystems that support Distributed User Interfaces where users "transfer" information from one device to another one. However, users interact with two independent user interfaces that share information.

The question is "Are we really sharing the user interface?" Or we are just sharing the information between two different user interfaces.

Let analyze the following hypothesis. Suppose that you are browsing information on the Internet using your Smartphone. You go to your favorite Web Site and you start reading an interesting article while you are on the bus, train or metro on your way home. When you arrive home you find your flatmates watching the Smart TV. As soon as you tell them about the article you have read, they suddenly got interested in it. To quickly share this information with them, you "transfer" the article from the Smartphone to the SmartTV as can be observed in Figure 1.

Note that we are not "transferring" or "synchronizing" the information on both displays; instead we are only "distributing/moving" the **article** HTML tag from the Smartphone to the Smart TV and not the whole page.

In this case, we have overcome the Smartphone display size limitations by employing the display of a device in the same ecosystem that overcomes the size limitations of the first device.

[1] Samsung AllShare service. URL =

http://www.samsung.com/es/experience/allshare/pcsw/

Figure 1: Distributing the ARTICLE HTML tag of a Web page.

As consequence of the synergy that emerges from the combination of the characteristics of both devices (mobility vs. display size), users take advantage of the intrinsic advantages of both devices.

Although the article HTML tag was distributed from one user interface to another one; the control (i.e. scrolling the article contents) is also transferred to the TV. It is coherent since the scrolling capabilities depend on the display size. However, it is not always the case.

Let analyze the following scenario. Suppose that you are projecting a Web page on the Wall to show participants information about a Web site. To address the site, you use the laptop keyboard and trackpad. Once the page has been loaded, you have a navigation menu on the left to navigate through the site; therefore, the keyboard and the trackpad are not required any more to show the contents of the Web site. To make the presentation more dynamic, you decide to distribute the **nav** HTML tag from the Web page projected on the wall to the Smartphone. Thus, you are able to navigate through Web site pages using the Smartphone. This scenario is depicted on Figure 2.

Note that unlike the first scenario, actions on the Smartphone affect the projection (i.e. when you click on a link in the Smartphone menu, the contents on the projection change accordingly). It is worth to highlight that the information is not the only entity to be distributed because the behavior of the Web component is also distributed to the target device.

Figure 2: Distributing the NAV HTML tag of a Web page.

The last two scenarios show how the application information as well as the behavior of the Web components are distributed across different devices.

Finally, we expose a new scenario that adds a new aspect to be taken into account, the user interface state continuity. Suppose that you are filling a Web form using your Laptop, but it is getting late to catch the train. As you have filled many fields of the form, you decide to "transfer/distribute" the Web form of the Web page to the Smartphone to continue the task. In this particular case, the distribution action also involves the transference of Web component states. Thus, the information that was introduced using the laptop is not lost during the transition (see Figure 3).

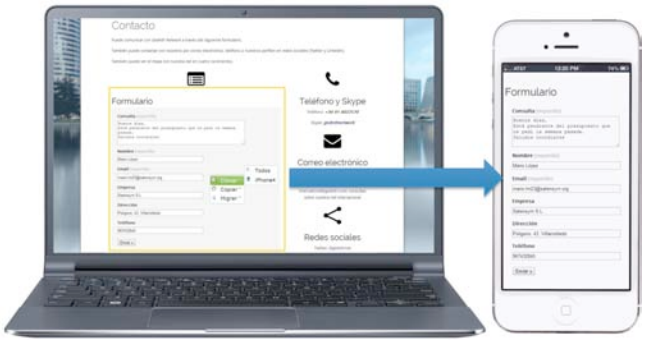

Figure 3: Information continuity

As result of the analysis of these scenarios, we can state that:

1. The first two scenarios (Smartwatch - Smartphone and Smartphone - SmartTV) define two user interface ecosystems. Each ecosystem is composed by two user interfaces, which are coupled to show shared information.
2. The last three scenarios (**article** component distribution, **nav** component distribution and continuity) define three user interface ecosystems. As in the first two scenarios, each ecosystem is also defined by two user interfaces; however, these scenarios besides sharing information, they share components (i.e. actions performed on the **nav** Web component in the fourth scenario, which was distributed from the laptop to the Smartphone, affects laptop user interface)

While the first two scenarios define distributed user interfaces, the last three scenarios define user interfaces that allow users to distribute user interface components among devices.

The need for the characterization of user interface ecosystems leaded us to analyze different user interface models, such as Cameleon Reference Framework (CRF) [2] based models (i.e. UsiXML), or the Interaction Flow Modelling Language [3]. Most of these models employ tree-based structures to describe the user interface structure (i.e. all interaction objects, except for the root, have a single parent).

Therefore, these models are not suitable to characterize user interface ecosystems where components can be hosted by mode that one container.

CHARACTERIZATION OF THE USER INTERFACE DISTRIBUTION

This section describes the user interface distribution metamodel to characterize user interface ecosystems where components can be hosted in more than one container.

The metamodel is defined using the ECORE EMOF [4] dialect enriched with Object Constraint Language (OCL) [5]. It is depicted on Figure 4 and allows users to define user interface ecosystems as graphs (represented by the UISystem metaclass). This graph is defined by two types of nodes: interaction objects (represented by the InteractionObject metaclass) and platforms (represented by the Platform metaclass). Besides, it defines three types of edges: interaction dependencies (represented by the InteractionDependency metaclass), hostings (represented by the Hosting metaclass) and implementations (represented by the implementation metaclass).

The *Platform* metaclass defines the combination of Hardware and operating system that supports the user interface. For instance, tablets or Smartphones running the Android or iOS operating systems, or laptops running the Microsoft Windows or Linux Operating systems.

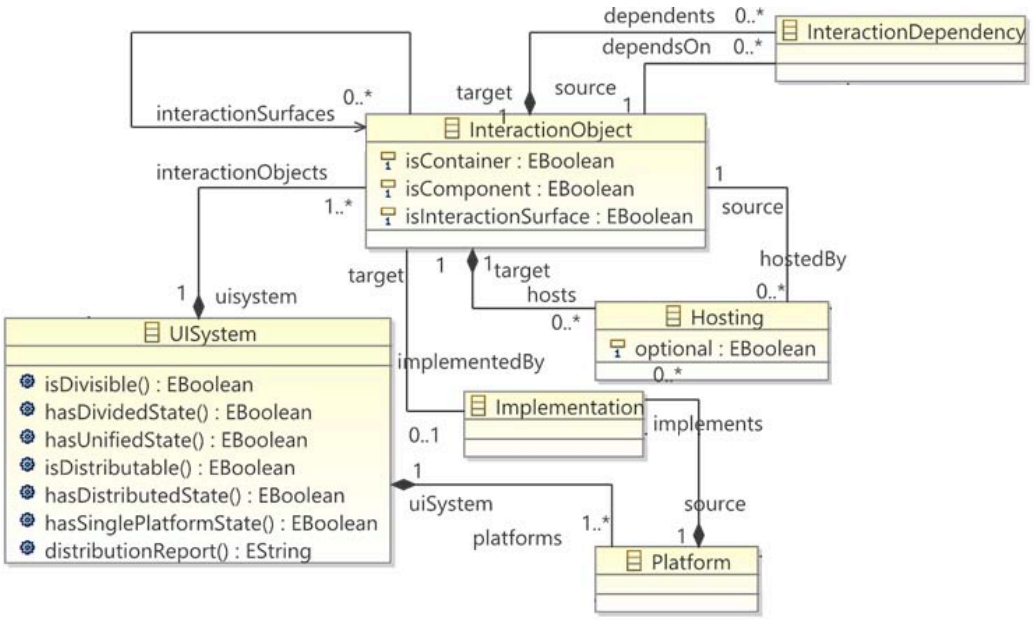

Figure 4: The user interface distribution metamodel

The Web as a platform generates several dilemmas. After analyzing the pros and contras, we consider the Web browser as a platform itself. However, although two Web browsers running in the same machine define two different platforms; two Web browser tabs (even in separate windows/frames) define a single platform. Another controversial issue regarding the platform is related to the use of multiple monitors connected to the same device. From our perspective, monitors are considered peripheral input/output devices (no operating system runs on a monitor). Therefore, N monitors connected to the same computer belong to a single platform.

The *InteractionObject* metaclass plays a similar role to the Abstract Interaction Object [8] defined in the CRF [2]. According to how an interaction object is related to the rest of the user interface, it plays one of the following roles: Interaction Component, Interaction Container or Interaction Surface. Some concrete examples of Interaction Objects are: frames, windows, dialogs, panels, text fields, buttons, labels, etc.

The *Hosting* metaclass defines a relationship between two Interaction Objects, the host and the guest. It represents that the guest Interaction Object can be hosted in the host Interaction Object during the execution of the user interface. Therefore, a guest Interaction Object can be hosted in more than one host Interaction Object. Besides, all guest Interaction Objects must be hosted in at least one host Interaction Object during the user interface lifetime (note that the host of a guest Interaction Object may change during the user interface lifetime).

If an Interaction Object does not host any Interaction Object then it becomes an *Interaction Component* (i.e. a button, a text field, a menu item, a NFC tag, etc.). However, if an Interaction Object hosts another Interaction Object then it becomes an *Interaction Container* (i.e. a panel, a layout, a menu, a submenu, a table, etc.). Note that all interaction containers must be contained in another Interaction Container.

The *Implementation* metaclass defines a relationship between an Interaction Container and a Platform. It represents that an Interaction Container is supported by a Platform. An Interaction Container is implemented by at most one Platform.

An Interaction Container, which is implemented by at most one Platform, turns into an *Interaction Surface*. The main difference between an Interaction Surface and an Interaction Container lays on the capability of the Interaction Surface of not being hosted on any other Interaction Object. Some examples of Interaction Surfaces are: Windows, Activities, NFC Panels, Pages, Views, etc.

Finally, the *InteractionDependency* defines a relationship between two Interaction Surfaces (master and slave). It represents that the lifetime of the slave Interaction Surface depends on the lifetime of the master Interaction Surface. When the master Interaction Surface is destroyed, all slave Interaction Surfaces are destroyed too. For instance, floating toolbars depend on the window/frame they are docked.

USER INTERFACE DISTRIBUTION CHARACTERISTICS

Once we have defined all concepts and relationships in terms of the metamodel depicted in Figure 4, the distribution characteristics of the user interface are defined in terms of states and capabilities.

User interface distribution states

We define the user interface distribution state as the organization/configuration of all the Interaction Objects that are part of a UI System (representing a user interface ecosystem) at a given instant in time.

Unified State: A UISystem reaches the Unified State iff all Interaction Objects are hosted on the same Interaction Surface at a given time

Divided State: A UISystem reaches the Divided State iff it has at least two Interaction Surfaces which host at least one Interaction Object each at a given time.

Distributed State: A UI System reaches the Distributed State iff it defines at least two Interaction Surfaces that are hosted on different Platforms. As Interaction Surfaces are Interaction Containers, they host at least one Interaction Component each at a given time.

Single Platform State: A UI System reaches the Single Platform State iff all Interaction Objects that are part of the UI System are hosted on a set of Interaction Surfaces that share the same Platform.

User interface distribution capabilities

We define the user interface capability as the set of user interface configurations (states) that a user interface ecosystem is able to reach.

Divisible: A user interface ecosystem is divisible iff exists an Interaction Object that can be hosted in more than one Interaction Surface. It means that any application that supports floating toolbars defines a user interface that is divisible.

Distributable: A user interface ecosystem is distributable iff exists at least one Interaction Object that can be hosted on at least two Interaction Surfaces implemented on different Platforms. It means that any application that allows users to "transfer" components from one platform to another one defines a user interface that is distributable.

USER INTERFACE DISTRIBUTION MODEL EDITORS

The metamodel is supported by two types of graphical editors implemented as Eclipse plugins to manipulate user interface distribution models. These editors were developed using the Eclipse Modeling Framework (EMF) [6] and the Graphical Modeling Framework (GMF) [7].

The Figure 5 shows the GMF model editor. It allows developers to easily manipulate and validate models as graphs.

The validation constrains where defined in OCL on the metamodel as well as on the graphical parts of the diagrams.

The user interface allows analyzers/developers to easily locate validation errors using the Problems view as well as the icons on the graphical components that violate model constraints. Besides, it supports both, on demand and live validations to improve model editor performance on big models.

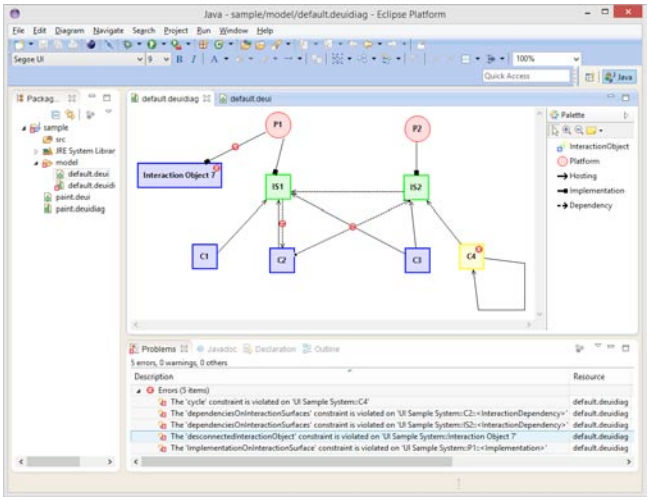

Figure 5: GMF-based user interface distribution model editor

The Figure 6 depicts the domain specific language supported by the GMF model editor. On the left, we show the Paint .NET user interface; on the right, we show the distribution model of the Paint .NET user interface built with the GMF model editor.

Rectangles represent Interaction Objects. The color changes according to the role they play on the model. Interaction Surfaces are green, Interaction Containers are yellow and Interaction Components are blue.

Red circles represent Platforms.

The hosting relationship is represented by a solid black line with an arrow pointing to the host Interaction Objects.

The implementation relationship is also represented by a solid black line which ends with a square pointing to the Interaction Object.

Finally, the dependency relationship is represented by a dashed line with an arrow pointing to the Interaction master object.

Although the GMF editor is a good model manipulation tool, the EMF-based editor (aka reflexive editor) allows analyzers/developers to manipulate models as trees where all model elements are represented as tree nodes.

The main advantage of this representation is the possibility to select any element of the model to set the context of OCL expressions that can be executed on the model using the OCL console tool as can be seen in Figure 7.

The UISystem metaclass defines a set of queries to derive user interface ecosystem characteristics (states and capabilities) in order to answer if the user interface ecosystem: is divisible or distributable, and if it reaches the unified, divided, single platform or distributed states

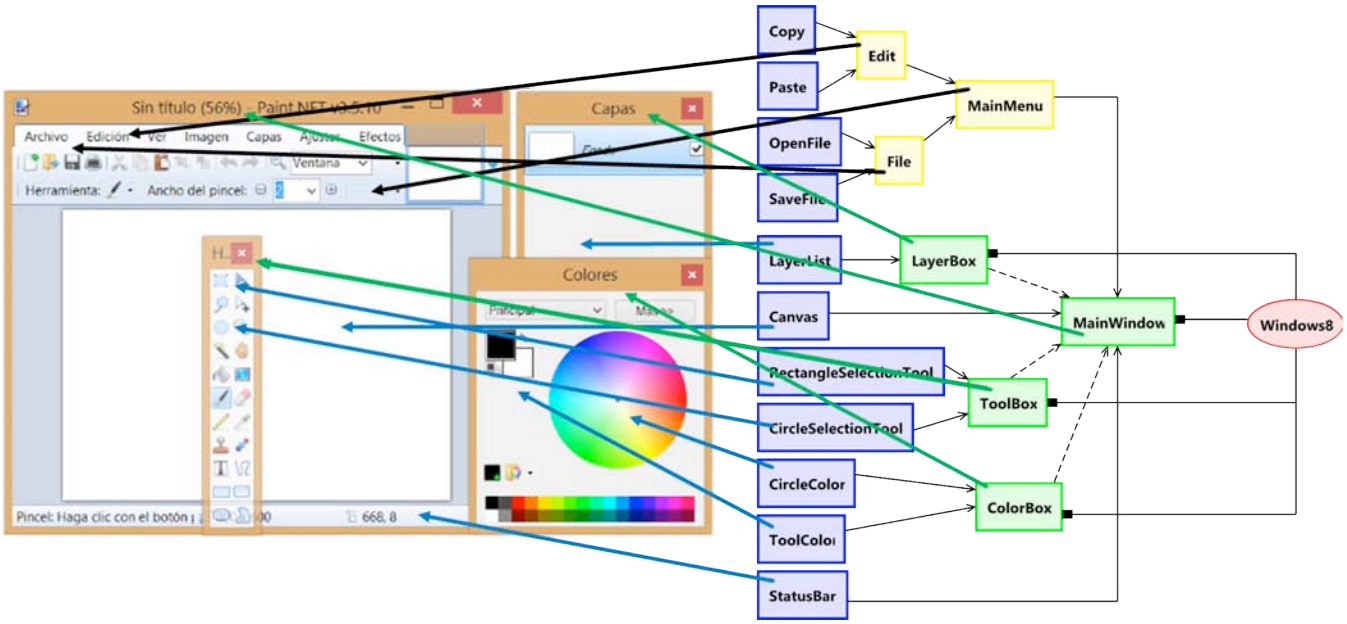

Figure 6: Domain specific language for the user interface distribution metamodel. The Paint .NET sample

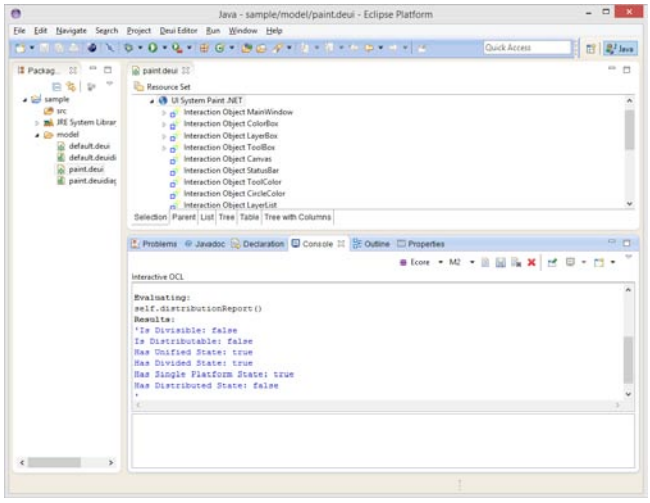

Figure 7: EMF-based user interface distribution model editor

The UISystem also defines a *distributionReport* query that returns the characteristics of the user interface ecosystem.

CASES OF STUDIES

This section analyzes the user interface distribution properties (capabilities and states) of five cases of study.

The Quiz user interface

The first case of study to analyze is the Quiz application [9]. The Figure 8 shows the Quiz user interface distribution model that shows two platforms (desktop and mobile). Each platform defines two Interaction Surfaces.

While the first one defines the application user interface, the other one defines the meta-user interface. Using the meta-user interface, the application is able to distribute components among different platforms (i.e. *distributable*).

However, as the meta-user interface is attached to each platform, the *single platform* state cannot be reached.

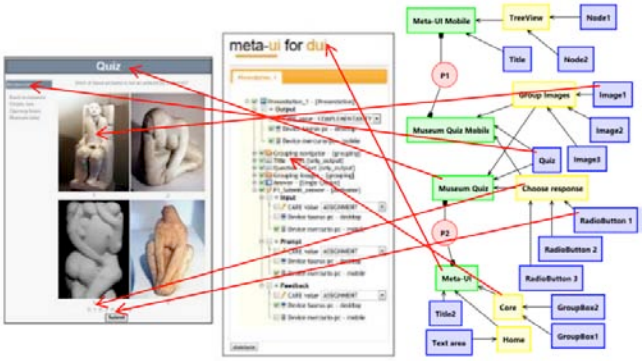

Figure 8: The Quiz user interface distribution model

The Paint .NET user interface

The second case of study to analyze is the Paint .NET user interface model, which is depicted in Figure 6. This application does not have the *divisible* neither the *distributable* capability.

Even though it supports floating toolbars, which is an indication that it might be considered a *divisible* user interface, it is not the case because although they depend on the Main Window Interaction Surface, they are not hosted in it (i.e. they cannot be attached). Regarding the distribution, as it runs on a single platform, it cannot reach the *distributed* state.

The GIMP 2.7 user interface

The third example analyzes the user interface model of the GIMP 2.7 application which is depicted on Figure 9.

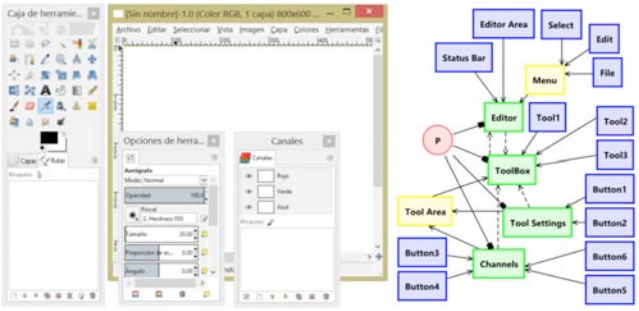

Figure 9: GIMP 2.7 user interface distribution model

As the model defines one platform only, there is no distribution possible. Therefore, it is not *distributable* and it is a *single platform* user interface.

However, from the divisibility perspective, this example shows how an Interaction Surface can be docked on an Interaction Container (i.e. the Channels Interaction Surface on the Tool Area Interaction Container).

Thus, the Channels Interaction Object plays two roles according to the user interface state. It plays the Interaction Surface role when it is a floating toolbar, or it plays the Interaction Container role when it is docked on the Tool Area Interaction Container. Therefore, this user interface is *divided*.

Besides, it is worth to highlight that this user interface cannot does not reach the *unified* state because the editor and Tool Box Interaction Surfaces depend on each other and there is no Interaction Object that is able to host both of them.

The WallShare user interface

The fourth user interface to be analyzed is the WallShare [10, 11]. The WallShare runs on 3 different platforms and it is a peculiar example of user interface distribution.

It is a *distributed* user interface because it reaches the *distributed* state. However, it is not a *distributable* user interface because it cannot "transfer" user interface components from one platform to another one. Consequently, it cannot reach the *single platform* state. However, it able to reach the *unified* state because users can close all WallShare clients while the WallShare Server is still running.

Besides, even if it reached the *divided* state it is not *divisible*. The Figure 10 shows the distribution model of the WallShare user interface.

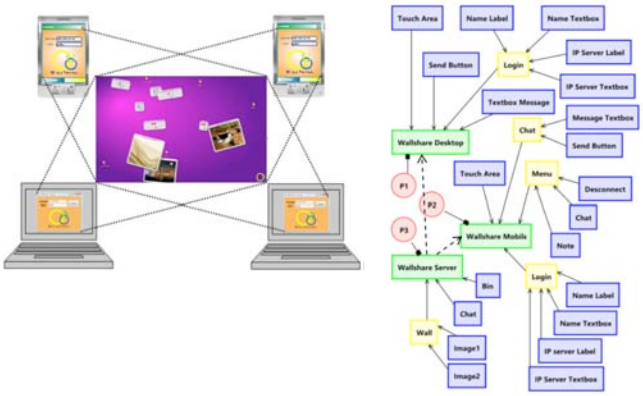

Figure 10: The WallShare user interface distribution model

The RFID ECOPanels user interface

The last user interface distribution model to analyze is the RFID ECO Panels [12, 13] application. This application presents a heterogeneous user interface that couples the user interface represented by RFID tag icons on a panel and a mobile device display. The Figure 11 shows the user interface distribution model.

As in the previous case, this user interface is not *divisible* neither *distributable*. Therefore, it cannot reach the *single platform* state. However, it can reach the *unified* state because the RFID platform is still able to interact when no client is working.

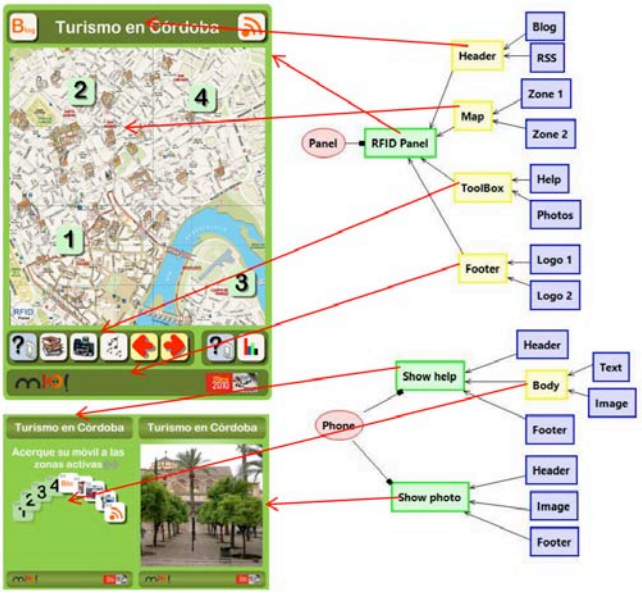

Figure 11: The RFID ECOPanels user interface distribution model

Besides, it is a *divided* and *distributed* user interface due to the capability of running in more than two platforms.

DISTRIBUTED USER INTERFACES VS. DISTRIBUTABLE USER INTERFACES

This section presents a summary of the user interface distribution capabilities and states of the cases of study we have presented.

The Table 1 shows a summary of the user interface distribution capabilities of each case of study.

Case of Study	Capabilities	
	Divisible	Distributable
Quiz	✓	✓
Paint .NET	✗	✗
GIMP 2.7	✓	✗
WallShare	✗	✗
RFID EcoPanels	✗	✗

Table 1. Cases of Study user interface capabilities

The Table 2 shows possible states reached by cases of study user interfaces.

Case of Study	States			
	Unified	Divided	Single	Distrib.
Quiz	✓	✓	✗	✓
Paint .NET	✓	✓	✓	✗
GIMP 2.7	✗	✓	✓	✗
WallShare	✓	✓	✗	✓
RFID EcoPanels	✓	✓	✗	✓

Table 2: Cases of study user interface distribution states

As conclusion, we have defined two user interface distribution characteristics: the user interface distribution capabilities and the user interface distribution states that can be reached.

The Distributed User Interface (DUI) concept has been redefined to become a state of a user interfaces instead of a capability. Besides, the Distributable User Interface (DeUI) was coined to represent the capability of a user interface to be distributed among different platforms.

Thus, a user interface that reaches the *distributed* state may not be *distributable*. In addition, a user interface that reaches the *divided* state may not be *divisible*.

PROXYWORK

To illustrate the concept of *distributable* user interfaces, we expose the Proxywork system [14, 15], which allows users to distribute the user interface components of Web applications among a set of displays.

The distribution is controlled by the user through a set of primitives (i.e. show, hide, copy, move, etc.) attached to Web page components.

The Proxywork Web proxy automatically attaches these primitives to Web page components on runtime.

Therefore, Web pages do not require any extra information to be distributed among different displays.

The Figure 12 shows how Web pages are processed in order to allow users to distribute their contents.

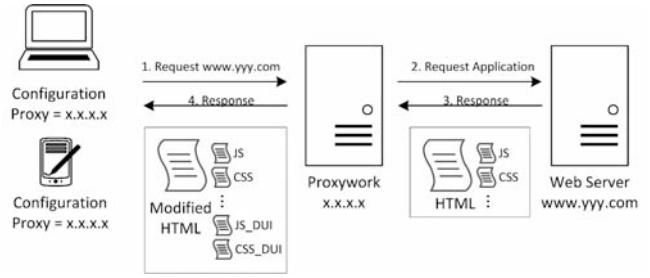

Figure 12: Proxywork Web page processing

All devices that are part of the user interface ecosystem are connected to a Web proxy.

As soon as a device requests a Web page the Web proxy the Web proxy checks if the device is registered on the system. If it is not, then the user has to register the device with a name to identify it. Once the device is registered, the Web proxy accesses the page on the Web server and introduces distribution primitives' functionality into the page. Then, the modified version of the page is sent to the device that requested it.

Proxywork primitives

Proxywork defines the following set of primitives:

- **Connect/Disconnect**: They allow users to add/remove a device to/from the user interface ecosystem.
- **Rename**: It allows users to change the device name/id on the user interface ecosystem it is part of.
- **Copy**: It allows users to copy a Web component from one interaction surface to another one. Actions performed on the copy affect the source interaction surface (see Figure 13).
- **Clone**: It allows users to copy a Web component from one interaction surface to another one. Actions performed on the copy affect the target interaction surface (see Figure 13).
- **Migrate**: It allows users to move a Web component from one interaction surface to another one. Actions performed on the copy affect the source interaction surface (see Figure 13).

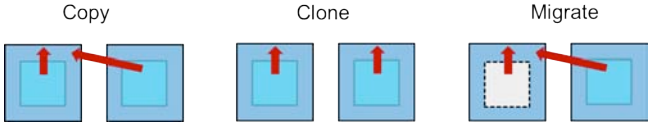

Figure 13: Proxywork distribution primitives

Proxywork limitations

The Proxywork system the following limitations regarding the Web pages to be distributed.

First, Web pages should be "well-formed". It means that the Web page structure should be defined in terms of **divs**, **articles**, **section**, **forms**, **navs**, etc, HTML tag elements. It is due to the fact that although Proxywork is able to distribute any Web component; the distribution is limited to structural HTML tags.

Another important limitation to highlight regarding Proxywork is the lack of the support of The HTTPS protocol.

Currently, Proxywork does not support any user interface device adaptation, though it is not difficult to introduce adaptation rules on the proxy.

Finally, the system does not support cross-domain communication due to HTML 5 restrictions.

CONCLUSIONS AND FUTURE WORK

This paper presents the difference between user interface distribution capabilities and states. To characterize these properties the user interface distribution metamodel is presented. This metamodel allows analyzers/designers to build user interface models to find out user interface capabilities and calculate user interface states.

This article also presents a graphical model editor, which was implemented as an Eclipse plugin to create and manipulate user interface distribution models, and a reflexive model editor to calculate user interface capabilities and states.

Using these editors, five user interface cases of study were analyzed. As result of this analysis, the distributed user interface (DUI) concept is redefined as a state instead of a capability. Besides, the distributable user interface (DeUI) concept is presented to define the user interface capability of distributing their components among the set of platforms that are part of the user interface ecosystem.

Finally, to illustrate a distributable user interface (DeUI), this paper present the Proxywork system which allows users to distribute user interface Web components among different devices.

As future work, we are considering the following lines. First, we are dealing with the HTTPS Proxywork limitation, which really restricts Proxywork capabilities and application scenarios, such as Facebook, Twitter, etc.

Besides, we are studying the way to link task models to user interface distribution models in order to validate and generate distributable user interfaces.

Finally, we are working on the control of distributed components.

ACKNOWLEDGMENTS

We thank the CICYT-TIN 2011-27767-C02-01 Spanish project.

REFERENCES

1. L. Terrenghi, A. Quigley and A. Dix. A taxonomy for and analysis of multi-person-display ecosystems. Personal Ubiquitous Comput., vol. 13, nº 8, pp. 583-598, 2009.
2. G. Calvary, J. Coutaz, L. Bouillon, M. Florins, Q. Limbourg, L. Marucci, F. Paternò, C. Santoro, N. Souchon, D. Thevenin and J. Vanderdonckt. The CAMELEON reference framework, Deliverable 1.1, CAMELEON Project, 03 September 2002. URL: http://www.w3.org/2005/Incubator/model-based-ui/wiki/Cameleon_reference_framework. [last access: 06/12/2013].
3. Object Management Group. IFML: The Interaction Flow Modeling Language. URL= http://www.ifml.org/ [last access: 06/20/2014].
4. Object Management Group. Meta-Object Facility (MOF). URL=http://www.omg.org/mof/ [last access: 06/20/2014].
5. Object Management Group. Object Constraint Language (OCL). URL=http://www.omg.org/spec/OCL/ [last access: 06/20/2014]
6. Eclipse Foundation. Eclipse Modeling Framework Project. URL= http://www.eclipse.org/modeling/emf/ [last access: 06/20/2014]
7. Eclipse Foundation. Graphical Modeling Project. URL= http://www.eclipse.org/modeling/gmp/ [last access: 06/20/2014]
8. J. Vanderdonckt and F. Bodart. Encapsulating knowledge for intelligent automatic interaction objects selection. Proc. of INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems, Amsterdam, 1993.
9. M. Manca y F. Paternò, Distributed User Interfaces with MARIA. Proc. of Distributed User Interfaces 2011 (DUI 2011) CHI 2011 Workshop, Vancouver, BC, Canada, May 2011.
10. P. G. Villanueva, J. A. Gallud and R. Tesoriero. WallShare: A collaborative multi-pointer system for portable devices. Proc. of Workshop on coupled displays visual interfaces in conjunction with AVI 2010, pp. 31-34. May 25th, 2010. Rome, Italy.
11. P. G. Villanueva, J. A. Gallud and R. Tesoriero. Multi-pointer and Collaborative System for Mobile Devices. Proc. of the MobileHCI 2010. ACM. September 7th-10th, 2010. Lisbon, Portugal.
12. P. G. Villanueva, R. Tesoriero, J. A. Gallud, A. H. Altalhi. A Framework to Develop Web Applications Based on RFID Panels. International Journal of Universal Computer Science. ISSN: 0948-695x, vol. 19(12), pp 1792-1807. 2013.

13.R. Tesoriero, P. G. Villanueva, H. M. Fardoun, G. Sebastián. Distributed User Interfaces in Public Spaces using RFID-based Panels. International Journal of Human-Computer Studies. ISSN: 1071-5819, vol. 72(1), pp 111–125. 2014.

14.P. G. Villanueva, R. Tesoriero and J. A. Gallud. Distributing web components in a display ecosystem using Proxywork. BCS-HCI '13. Proc. of the 27th International BCS Human Computer Interaction Conference, Brunel University, London, UK, 9-13 September 2013, art. 28. URL=http://dl.acm.org/citation.cfm?id=2578048

15.P. G. Villanueva. Distributable User Interfaces. PhD Thesis. University of Castilla-La Mancha. 2014.

Towards User-Centered Distributed Mashups

Oliver Mroß
Faculty of Computer Science
Technische Universität Dresden
Dresden, Germany
oliver.mross@tu-dresden.de

Klaus Meißner
Faculty of Computer Science
Technische Universität Dresden
Dresden, Germany
klaus.meissner@tu-dresden.de

ABSTRACT

Today's availability of web-enabled and mobile devices has led to a paradigm shift in the development of web applications. They are no longer restricted to a single device that is used by a single user. Future web applications are distributed across the borders of heterogeneous devices as a set of interconnected components using a message brokering system. With this approach new challenges arise, e. g., the inclusion of dynamically available devices during the application's load time or the discovery and integration of their capabilities (sensors, communication interfaces or installed apps etc.) at run time. In this paper, we present our ongoing work towards a distributed *client-server runtime environment* (CSR) that should support the dynamic distribution and user-centered adaptation of composite multi-device web applications – denoted as *distributed mashups*.

Author Keywords

Distributed Mashups; Multi-Device Web Application; Dynamic Device Composition.

ACM Classification Keywords

D.2.2 Software Engineering: Design Tools and Techniques; D.2.11 Software Architectures: Domain-specific architectures

INTRODUCTION

The increasing availability of web-enabled and interactive devices leads to the need for the combined use of their capabilities, for instance, to use the smart phone's motion sensor as input source for a remote controller scenario or using a tablet PC as a collaborative notepad in a multi-user application. Regarding the combination of devices, we are following the approach of [2] and use the term *multi-device application* for denoting such scenarios. They are subject to dynamic variations of their physical execution environment, due to joining and leaving (mobile) devices. Furthermore, integrated application fragments (presentation, application logic, data access or device-specific I/O components) are simultaneously distributed across heterogeneous platforms and are communicating with each other based on a distributed message brokering system. Considering these characteristics, current development methods (for desktop and web applications or apps) are resulting in an unreasonable development effort, due to the lack of concepts for supporting the context-sensitive and platform-independent distribution of application fragments as well as its adaptation at run time.

Model-based distributed UI development approaches [7, 8] are lacking concepts for adapting the application's composition and communication relations as consequence on changes in the physical environment. Günalp et al. propose a rule-based development approach for multi-device scenarios, which empowers the application's runtime environment to react autonomously on context changes [3]. However, resulting applications are not flexible enough to address varying user requirements, for example, to integrate the personal smart phone situationally as remote presentation controller in an informal team meeting. We argue that the application developer neither can anticipate all relevant situations nor the expectations of the user.

Considering the dynamic nature of multi-device applications, the *mashup paradigm* becomes an interesting development approach, because mashup components are independently executable and loose-coupled web-based building blocks with a clearly defined interface facilitating their use in dynamic application creation and adaptation scenarios. Moreover, combining the mashup paradigm with new extension mechanisms of modern web-browsers, e. g., by using the *Cordova framework*[1], the mashup development approach is no longer restricted to web resources. However, there are a few multi-device mashup approaches [4, 5], which are only focusing on the distribution of UI elements without providing concepts for the dynamic discovery and integration of device capabilities and resources encapsulated as mashup components with additional context properties, such as the location, constrained resources or access privileges.

In this paper, we introduce our model-based approach for developing multi-device mashups considering the dynamic nature of their physical execution environment at load and run time. In Section 2, we describe a sample scenario for illustrating our vision of an adaptable multi-device mashup. Afterwards, Section 3 comprises fundamental concepts and a meta-model for describing the mashup's initial distribution in

[1] https://cordova.apache.org/

DOI http://dx.doi.org/10.1145/2677356.2677658

a multi-device scenario. In Section 4, we discuss adaptation use cases and immanent challenges for our ongoing work towards user-centered distributed mashups.

SCENARIO

John would like to present some slides to a group of people. To this end, he connects his tablet pc with a smart board virtually and starts loading a distributed slide presentation mashup. The application's bootstrap process includes several steps (cf. Figure 1). At first, John logs on to a *multi-device mashup web-service* (MDMS) using the browser-based client runtime that registers his current tablet pc (step 1).

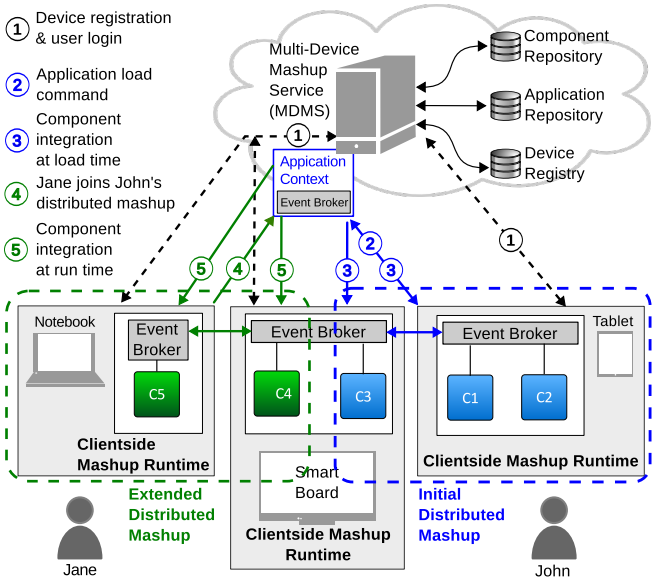

Figure 1. Client-Server-Runtime Overview

We assume, the smart board was already registered at the MDMS. After John has logged on, his tablet pc receives a list of available mashups from the MDMS. John selects a slide presentation mashup, whose components (C1 - slide controller, C2 - slide loader, C3 - slide presenter) will be distributed to his tablet pc and the smart board simultaneously (step 3). While John is presenting his slides, suddenly Jane has some regarding questions. In order to enhance the communication, she prefers to use a visual sketching UI component. For this purpose, she logs on to the MDMS and joins John's application (step 4). Afterwards she modifies the application's composition by integrating (step 5) a visual sketching component to her notebook (C5) and the smart board (C4). While Jane is interacting with the sketch component presented on her notebook, every modification is synchronized with the instance presented on the smart board.

In the next Section, we describe concepts concerning the mashup's composition and component model, communication relations and its initial distribution with respect to the potentially changing physical execution environment.

DISTRIBUTED MASHUPS

With the term *distributed mashup* we follow the idea of a virtual and distributed *application space* [1] in that each device could be regarded as *runtime container* to execute loose coupled *black-box mashup components*, which are communicating across device/platform borders using a server-side message brokering system based on the event publish-subscribe paradigm. The physical environment of a distributed mashup is subject to dynamic variations, due to joining and leaving devices. In our approach, we denote the abstraction of the application's physical environment on runtime layer as *environment model*. It aggregates functional and non-functional properties of registered containers at run time. Moreover, we distinguish between the concept of a *global* and an *application specific environment model*. The former includes all devices registered on the runtime layer. The latter is bound to a specific application space and contains only those containers, which were added into the application space explicitly.

Our approach is based on the component and composition meta-model developed as results of the CRUISe-project [9]. In the next Section, we give a brief introduction to relevant concepts. For describing the application's distribution state as well as validating its modifications at run time, we added further concepts on the meta-model layer, which we characterize afterwards.

Composition and Component Model

The application's structure, communication behavior and distribution state is defined respectively in the *conceptual-*, *communication-* and *distribution model* as entities of the *composition model*, such as presented in Figure 2. The conceptual model contains functional requirements modeled as component entities by reusing concepts of the following component meta model. Uni- and bidirectional communication channels between mashup components are defined in the communication model that supports following paradigms: *fire-and-forget*, *request-response* and *bidirectional property synchronization*. Assigning components to different devices at load- and run-time is realized using concepts of the distribution model, which is described in the next Section.

Each mashup component – encapsulating application or web service logic as well as UI elements – adheres to the same platform independent meta-model describing the component interface using three abstractions: *operations, events* and *properties* (cf. Figure 2). Furthermore, the component model includes domain ontology concepts to specify data and functional semantics of the interface elements [9]. In addition, it includes concepts for describing the component's dependencies (e. g., external source code, documents etc.) and their required platform or device features, e. g., sensor APIs, functionalities of native apps or embedded in- and output devices. The latter aspect is required to compute possible distribution changes or device capability integration options.

Distribution of Mashup Components

Considering the dynamic device availability, we provide the *distribution* and *context condition* as elementary concepts of the *distribution model*, such as presented in Figure 2. A *distribution* represents the assignment of a group of components to one or more runtime containers. The amount of potential runtime containers can be greater than one, because we model the distribution of at least one component as *context*

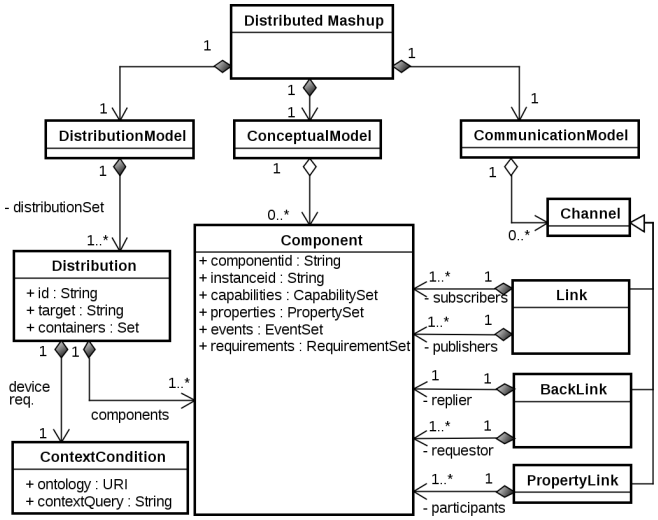

Figure 2. Overview of distributed mashup meta model

condition request. Concerning the latter concept, we follow the definition of the term *delivery context* [6] and use device, platform and modality characteristics to describe the required container and its "carrier" device. A fitting example is the following query statement:

```
SELECT ?c WHERE {
 ?c a s:CSRClient; s:runOn ?d.
 ?d a d:Smartphone; d:hasOutputDevice ?td.
 ?td a d:Display; m:supportsModality m:Tactile.
}
```

It selects a container that is executed on a smart phone, which is providing a display with tactile input support. The query statement is formatted using the *SPARQL Protocol and RDF Query Language*[2] (SPARQL). As statement vocabulary we use the *DoCUMA platform ontology*[3] that is based on the concepts defined in the W3C delivery context ontology [6]. Moreover, to determine applicable runtime containers, each device has to provide a self descriptive profile while joining into the application space. It comprises concepts of the aforementioned ontology and is aggregated into the environment model managed by the MDMS.

Before a distributed mashup is generated and initialized, at least a valid runtime container should be registered in the application space. From a component's perspective a runtime container is valid when it provides all requested features, e. g., min/max resolution or a specific rendering engine. If no matching container is available, the user can select between the cancellation, the interruption of the loading process (until a new device joins the application space) or the modification of the application's composition. The latter end-user development aspect is out of scope of this paper. If the loading process is interrupted, the MDMS will listen for joining containers. After such an event occurs, the application's distribution requirements as well as the device requests of each component's interface are matched against the provided features described in the associated device profile. For each distribution entity the set of valid distribution targets results from the intersection between the set of *potential distribution targets* and the intersection of those containers, which are able to execute the component candidates derived from each associated component template. In other words, the resulting target set includes containers, which are able to execute the component candidates and fulfill the distribution requirements defined in the application model. They are presented to the user and s/he can specify the assignment of one or more components with respect to one or more devices of the application's environment model. Besides the mashup's initialization, we use the meta-model concepts to serialize and persist the mashup to reload it in another session, e. g., to continue an interrupted multi-device presentation.

After we have proposed our initial model-driven development approach of a distributed mashup, in the next Section we discuss adaptation use cases – derived from the previous scenario – and their immanent challenges and requirements on a supporting *client-server runtime* (CSR) environment.

Adaptation Use Cases

In our vision, we see the user as an integral element of the application's environment, who is able to modify the mashup's composition, communication and distribution.

Modifying a predefined application - Due to the device mobility and its constrained resources, loading a predefined distributed mashup is not always possible. Hence, following user-centered adaptation options should be supported by the CSR.

Updating the mashup's distribution becomes possible by the provision of a service that recommends valid runtime containers with regards to a fixed set of components of an associated distribution item. This option is activated, when every container feature request of each component is fulfilled considering the application-specific environment model, but not each requirement of the distribution model. If no runtime container was registered in the application space, the recommendation service may analyse the global environment model. In this case, not all valid containers could be integrated automatically, because of restricted access rights. This comes with the challenge of proposing a valid access right model concerning dynamic distribution state modifications (push/pull migration/replication) or the integration of protected device capabilities and resources in public or private scenarios.

Replacing components by alternative variants should be used in case of fulfilled distribution requirements, but unmet component feature requests. Instead of searching for an alternative container, the aforementioned service recommends functional equivalent and applicable alternatives with respect to the original components and the associated target container specified by the user. As mentioned before, loading a predefined mashup can be performed as resumption of a suspended application. This comes with the challenge of exchanging state information between original and alternative components, with different interface configurations. Thus, the CSR should apply mediation techniques for transferring component-specific states into a representation applicable by an alternative component.

[2] http://www.w3.org/TR/sparql11-overview/
[3] http://goo.gl/4LeKBX

Joining a running application - After the user registers her/his personal device at the MDMS, s/he can choose from a set of active applications, determined from the user's access rights. The join into an existing application space results in the recommendation of following adaptation options.

Distributing new components - For this purpose, either the user defines target devices at the beginning and afterwards selects applicable mashup components or at first s/he defines required components and subsequently choose one or more target containers currently available. In this use case, the MDMS recommends components by validating their feature requirements using reasoning algorithms with respect to the current state of the application-specific environment model. To this end, real-time data of each container (current energy level, processor and memory load etc.) has to be included in the validation process.

Remote device coupling - After joining into an application, the user can build up a mental model using a visualization of associated mashup components and communication channels for each runtime container registered in the application space. Selecting an arbitrary component results in several coupling recommendations. A recommended entity includes the selected component as sender/receiver and another running or not integrated component. In this use case, the challenge is the similar to the one described in the first use case. Another challenge is the dynamic encapsulation of device capabilities and resources as composable mashup components using the device profile. Possible solutions are the derivation of matching components or the ad-hoc creation of generic components using model-to-code transformations executable on each runtime container.

Dynamic distribution modification - In this use case, we distinguish between the *migration* and the *replication* of mashup components. Considering the latter approach, two different options are valid - the *coupled* and *uncoupled* replication. The first copies the component's state into a corresponding functional equivalent alternative, that is connected with the original component through at least a single channel. Migrating components between different runtime containers modifies the application's distribution state, but not its composition and communication. As consequence of migrating UI components between different devices, replacement strategies should be performed by the MDMS. The immanent challenge of both distribution update operations is the conception of a state synchronization and mediation mechanism in scenarios of heterogeneous but functional equivalent mashup components.

Leaving a running application - Removing a runtime container from an active mashup could lead to the application's breakdown, due to missing information providing components. Such failure situations should be mitigated by finding alternatives or migrating mashup components to other devices or to the MDMS for persisting the application state for later reuse.

CONCLUSION AND FUTURE WORK

Considering the dynamic nature of multi-device scenarios, we proposed the use of *delivery context* condition statements for describing the application's initial distribution as well as container feature requirements defined in the component model of a *distributed mashup*. In combination with semantic device profiles, our approach can be the basis to perform several adaptation options at run time in ad-hoc integration scenarios, which we discussed at the end of this paper.

Our future work comprises the extension of our client-server-runtime for adapting migrating/replicating components in heterogeneous multi-device scenarios using a replacement strategy. Moreover, we will integrate existing mediation techniques to realize the component state transfer described in the previous Section.

REFERENCES

1. Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J., and Zukowski, D. Challenges: An Application Model for Pervasive Computing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, ACM (New York, NY, USA, 2000), 266–274.
2. Chmielewski, J., and Walczak, K. Application architectures for smart multi-device applications. In *Proceedings of the Workshop on Multi-device App Middleware*, ACM (2012), 5.
3. Günalp, O., Gürgen, L., Lestideau, V., and Lalanda, P. Autonomic Pervasive Applications Driven by Abstract Specifications. In *Proceedings of the 2012 International Workshop on Self-aware Internet of Things*, Self-IoT '12, ACM (New York, NY, USA, 2012), 19–24.
4. Husmann, M., Nebeling, M., and Norrie, M. C. MultiMasher: A Visual Tool for Multi-device Mashups. In *ICWE Workshops*, Q. Z. Sheng and J. Kjeldskov, Eds., vol. 8295 of *Lecture Notes in Computer Science*, Springer (2013), 27–38.
5. Kovachev, D., Renzel, D., Nicolaescu, P., and Klamma, R. DireWolf - Distributing and Migrating User Interfaces for Widget-Based Web Applications. In *ICWE*, F. Daniel, P. Dolog, and Q. Li, Eds., vol. 7977 of *Lecture Notes in Computer Science*, Springer (2013), 99–113.
6. Lewis, R., and Fonseca, J. M. C. Delivery Context Ontology. `http://www.w3.org/TR/dcontology/` (02.04.2014), June 2010.
7. Manca, M., and Paternó, F. Distributing user interfaces with MARIA. In *Distributed User Interfaces 2011 (DUI 2011), CHI 2011 Workshop* (2011), 93–96.
8. Manca, M., and Paternó, F. Flexible Support for Distributing User Interfaces Across Multiple Devices. In *Proceedings of the 9th ACM SIGCHI Italian Chapter International Conference on Computer-Human Interaction: Facing Complexity*, CHItaly, ACM (New York, NY, USA, 2011), 191–195.
9. Pietschmann, S., Radeck, C., and Meißner, K. Semantics-based Discovery, Selection and Mediation for Presentation-oriented Mashups. In *Proceedings of the 5th International Workshop on Web APIs and Service Mashups*, Mashups '11, ACM (New York, NY, USA, 2011), 7:1–7:8.

Interacting with Tangible Objects in Distributed Settings

Elena de la Guía
Computer Science Research
University of Castilla-La Mancha, Albacete, Spain
Mariaelena.guia@uclm.es

María D. Lozano
Computer Systems Department
University of Castilla-La Mancha, Albacete, Spain
Victor.Penichet@uclm.es

Victor M. R. Penichet
Computer Systems Department
University of Castilla-La Mancha, Albacete, Spain
Victor.Penichet@uclm.es

ABSTRACT
The rapid evolution of technology has changed the way in which we can interact with interactive systems. New physical workspaces have appeared such as Multi-Device Environments (MDE). These scenarios implicitly support Distributed User Interfaces allow us to distribute user interfaces on different devices. In this way, we take advantages of distributed human innate cognition. However, interactions with pixels on these GUI screens are inconsistent with our interactions with the rest of the physical environment in which we live. In this paper, we propose two different interaction techniques based on Tangible User Interfaces (designed with NFC technology). Then, we discuss the strengths and weaknesses of interaction techniques in distributed systems settings.

Keywords
Tangible interaction, Distributed User Interfaces

ACM Classification Keywords
H.5.2. [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces, Interaction styles.

INTRODUCTION

The rapid evolution of technology has changed the way in which we can interact with interactive systems. New scenarios have appeared such as Multi-Device Environments (MDE). These scenarios implicitly support Distributed User Interfaces. According to González [3], Distributed User Interfaces (DUI) can be classified depending on the feature of the interface in a MDE. The interfaces can be divided into or distributed through among the ecosystem according to state, platform and distribution properties. The main goal of a DUI is to facilitate users tasks in the software system by means of providing them with an optimal configuration of user interface which are available in the user working environment. According to Vandervelpen and Coninx [17] we can find two types of interactive components that make up the system. *Interaction Resources (IRs)* are atomic input or output channels that are available and that can be used to carry out several tasks. This includes I/O facilities like keyboards, mice, screens, speakers, speech-recognizers. In this context the resource is limited to a single input or output modality. *Interaction Devices (IDs)* are computing systems that handle the input or send output to individual IRs that are connected to it. This includes devices such as mobile devices, desktop computers, and so on. However, the spatial distribution of interfaces is complex; the main challenge is how to configure and distribute the IUs among IR, ID to achieve a usable system. There are studies about it [16], however they do not emphasize what interaction technique would be more appropriate depending on the system, user, task, and so on.

We propose two interaction techniques based on Tangible User Interfaces (TUI). It refers to user interfaces which give physical form to digital information [8]. These are settle on smart objects, and provide a natural and easy style of interaction that proves intuitive and motivating for non-experts in technology. They have been developed with NFC (Near Field Communication) technology.

In the next section, we describe the interaction techniques used in Multi-Devices Environment. Then, we explain the novel interaction mechanism found on Tangible User Interfaces. Afterwards, we evaluated two prototypes that used Tangible User Interfaces to interact with the system and discuss advantages, disadvantages and some conclusions.

INTERACTION TECHNIQUES

An interaction technique is a combination of hardware and software elements that provides a way for computer users to accomplish a single task. Its aim is to facilitate user interaction with the system. That is, it should be intuitive, simple, easier-to-learn, etc. Several techniques have proposed to interact with multi-devices environments. In the next study [14] the authors explored interactions with connected devices by *moving a stylus* along paths on a printed map of the infrastructure that is annotated with barcodes. Rukzio and Holleis [13] discuss the design space of interactions and applications enabled by *pico projector units* integrated into mobile devices; and projection showing information related to the object on which the projector currently focuses on (augmented reality). *Stitching* [6] is an interaction technique that allows users to

July 01 2014, Toulouse, France

http://dx.doi.org/10.1145/2677356.2677659

combine pen-operated mobile devices utilizing wireless networking, by using pen gestures that span multiple displays. Beardsley et al. [1] They interact with the system through of a handheld projection system that lets users create opportunistic displays on any suitable nearby surface. *Touching* is a technique that involves touching an object, either with a finger or with a mobile device, to perform a task. Some examples of projects using this technique can be found in [2]. *Select-and-point* [10] consists in a touch-sensitive tabletop display and surrounding wall screens which is set up for efficient collaborative works. *Scanning,* in this case, the mobile device or any other device is able to scanne information and interact with the system to provide a service to the user [18]. *Pick & Drog* [11] is a pen-based direct manipulation technique. It allows users to pick up an object on a display and drop it on another display. *PointRight* [9] is a technique founded on a peer-to-peer pointer and keyboard redirection system that operates in multi-device environments. On the one hand, *Stitching, Pick&Drog* are techniques used to interact with the system pen and mobile devices. In this particular case, the combination (IR:pen and ID:mobile devices) requires to interact with the system. This atomic interaction can be diversified through the combination with input from keyboards, mouse, joysticks or sensors. However, almost all keyboards and pointing devices are tethered to a single computer; we cannot share a mouse between two users. Moreover, it can be confusing to distinguish the input device from its real/corresponding device. On the other hand, *touching, select-and-point, scanning* are techniques based on physical mobile interaction (used like ID). Thus, the collaborative tasks are more complex, because each user needs a capable device for it. Furthermore, pointing requires some cognitive effort to point at the smart device and it needs line of sight. There are IDs such as Projector, Kinect, Wii and so on. These permit an interaction technique based on gestures, being more intuitive by user. However, they are costly and have complex infrastructures. In addition, users need considerable concentration and physical skills, especially from inexperienced users. For instance, according to Igual [7] the interaction founded on mobile devices (touching) is difficult to people with limitations. It was therefore necessary to adapt it to the user. In order to provide tangible interaction we have based on Approach&Remove technique. It is a style of interaction that allows the user to interact with distributed user interfaces by approaching a mobile device to digitized objects. Our proposal is supported by this interaction technique; however we use input and output variants [12][15].

TANGIBLE USER INTERFACES

In order to interact with the multiple-devices we have digitalized everyday objects (now smart objects). The technology used has been NFC. It allows direct manipulation of wireless network connections by means of proximal interactions. For that reason, a tag (or more) is integrated inside the object or card depending on the size of the object; describing each tag a unique identifier. When the tangible object is brought closer to the NFC reader in the mobile device, the NFC tag inside the object is excited by electromagnetic waves sent by the NFC reader, and then the controller component sends the identifier to the server. The server checks this information and returns the appropriate user interfaces to the output device. We describe two different types of interaction depending on the interactive resource used as input in the system. The internal operation of the system is the same.

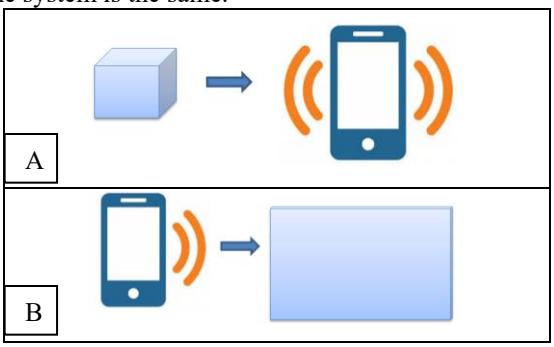

Figure 1. Inputs in Distributed Settings a) Interaction Resource is the smart object; b) Interaction Device is the mobile device

Input: Smart object

The interaction technique is called *Approach&Remove object.* In order to interact with the system, the user needs tangible user interfaces (smart objects). These interactive resources used the mobile phone (with an NFC reader) to connect with the system and send the required task. The user can interact with the system through everyday objects such as cards, toys, coins, etc. They only have to bring the object or tangible interface closer to the mobile device (see Figure 1a).

Input: Mobile Device

The interaction technique is denominated *Approach&Remove mobile device*. The input to use the system is the mobile phone (interaction devices). The style of interaction consists in bringing the mobile device close to a tangible interface. NFC-enabled mobile phones serve as pointing devices for the interaction with the diverse content of dynamic NFC-displays, including text, pictures, links, maps or custom widgets (see Figure 1b).

Output: Mobile Devices and Screen

In order to display the interface results, there are multiple output devices and multi-modal communication which permit output from different ways (auditory, visual, textual, and graphical). In this way, user experience is more pleasant, satisfactory and they can feel immersed in the task.

PROTOTYPES

In order to be able to evaluate the interaction techniques based on tangible user interfaces, we have developed two prototypes. This section presents their features and explains how interaction techniques are used in the systems.

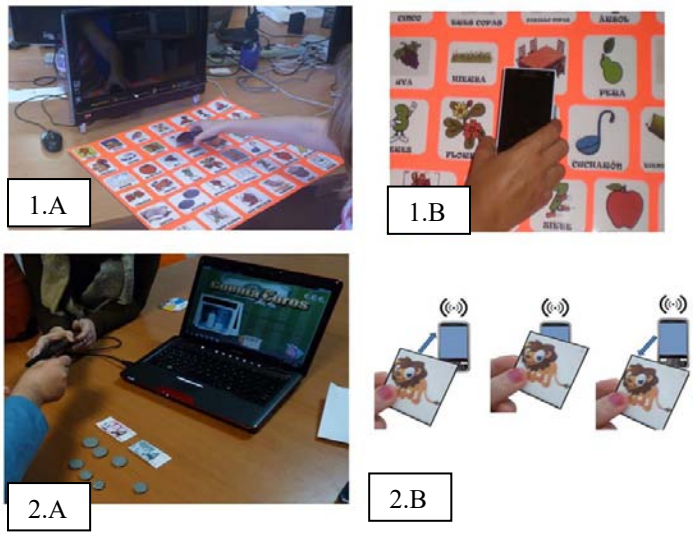

Figure 2. 1.a)Co-Brain Training 1.b) mobile device as input combined with the smart panel. 2.a) TraInAb Sytem 2.b) smart object to interact with the system.

Co-Brain Training [4] is a collaborative and interactive game based on a Distributed and Tangible User Interface in order to support cognitive training. The interaction techniques consist in bringing the mobile device close to a tangible interface (see Figure 2.1a).

TraInAb (Training Intellectual Abilities) [5] is an interactive and collaborative game designed to stimulate people with intellectual disabilities. The user can interact with the system through everyday objects such as cards, toys, coins, etc. Users only have to bring the object or tangible interface to the mobile device (with an NFC reader) (see Figure 2.2a).

EVALUATION AND DISCUSSION

In order to test the strengths and weaknesses of the interaction style we compared two kinds of interactions in the prototypes Co-Brain Training and TraInAb. The main goal was to test the effect of the new user interaction based on tangible objects. 12 users were recruited, and there were carried out 3 tasks by means of three different games (memory, calculation and linguistics). Direct observation was the method used in this research. The results are as follows: quantitative data concerning errors in the technique *Approach&Remove mobile device* was 17% −it is with regard to the 6% of errors when using the technique *Approach&Remove objects*. These studies have shown that people are able to learn fast and make very few errors after using it. Common errors are exemplified by touching NFC tag too briefly, complicating its reading in the mobile device; or selecting a wrong tag or smart object. **Infrastructure** is similar to the previous one but its cost is higher in the system which uses *Approach&Remove mobile device* technique. The higher cost stems from the fact that each user needs his/her own mobile device with regard to the other technique which allows them to interact with all users.

Rating: High, Medium,Low	**Approach&Remove object**	**Approach&Remove mobile device**
Error rate	Low	Medium
Infrastructure and Cost	Low	Medium(each)
Tangibility	High	Low
Affordance	High	Medium
Grouping	Medium	High

Table 1. Comparison of properties of the tangible interaction techniques

Tangibility is the attribute of being easily detectable with the senses. The objects are more common and familiar. The technique *Approach&Remove objects* offers a high tangibility. **Affordance** is often taken as a relation between an object that affords the user to perform an action. **Grouping**, *Approach&Remove mobile devices* technique offers the opportunity to group or organize functions or objects. Moreover, when they need to interact with more complex interfaces this technique is more appropriate. For instance, the spelling game had 27 options (related to the alphabet) and they preferred to use the digitized panels because if they are looking for any option in the panel these ones are more organized. That is, if we must distribute information or need to have it more organized, it is preferable panels as tangible interface. For this way, the panels can distribute menus and shortcut to use the system (see Table 1).

CONCLUSION

In this paper we describe two interaction techniques based on tangible user interfaces. The main objective is to provide simple and novel interaction mechanisms for environments that support distributed user interfaces. We developed and evaluated two prototypes that implement this concept. Tangible user interfaces are very intuitive and simple for users. Using tangible interaction provides benefits as explained below. Interaction with the system is simple and intuitive. Common items are familiar and can be easily assimilated by users, making it more predictable to use. They do not need prior knowledge of the system or device to use it. Direct interaction with objects allows a better understanding of the task. Furthermore, the tangible interaction is integrated into a real space and therefore it is always located in a specific place (not just on a screen). In addition, it allows us to extend a part of user interface in physical objects, simulating how user usually works in its surroundings, that is, focusing on a particular task (main UI) and interacting with everyday objects scattered around. For future work we want to perform a more detailed assessment and define patterns or guidelines to help designers to choose the method of interaction depending on the task and the user profile.

ACKNOWLEDGMENTS

This research has been partially supported by the Spanish CICYT research project TIN2011-27767-C02-01, from the Ministerio de Economía y Competitividad, and the regional projects with reference PPII10-0300-4174 and PII2C09-0185-1030, funded by the Junta de Comunidades de Castilla-La Mancha and the Diputación Provincial de Albacete.We would also like to especially thank Francisco Vizcaino García, Erika González ,Yolanda Cotillas and Nuria Belmonte their collaboration in the project; without your support all this work would not have been possible.

REFERENCES

1.Beardsley, P., van Baar, J., Raskar, R., Forlines, C. (2005) Interaction using a handheld projector. IEEE Comput Graph Appl 25(1):39–43.

2. Broll, G., Graebsch, R., Holleis, P., Wagner, M. Touch to play: mobile gaming with dynamic, NFC-based physical user interfaces, Proceedings of the 12th international conference on Human computer interaction with mobile devices and services, September 07-10, 2010, Lisbon, Portugal

3.González, P., Tesoriero, R. and Gallud, J. A. Revisiting the Concept of Distributed User Interfaces. In Distributed User Interfaces: Usability and Collaboration. Springer, Human-Coomputer Interaction Series. Eds. M. D. Lozano, J. A. Gallud, R. Tesoriero, and V. M. R. Penichet. ISBN: 978-1-4471-5498-3, pp. 1-15. 2013. url= http://dx.doi.org/10.1007/978-1-4471-5499-0_1.

4.Guía, E. de la, Lozano, M.D., Penichet, V.M.R: Tangible and Distributed User Interfaces to Improve Cognitive Abilities within People Affected by Alzheimer's Disease. DUI 2013: 3rd Workshop on Distributed User Interfaces: Models, Methods and Tools. In conjunction with ACM EICS 2013 June 24th, 2013. London, UK. ISBN-10: 84-616-4792-0 ISBN-13: 978-84-616-4792-7

5.Guía, E. de la, Lozano,M.D Penichet, V.M.R.: TrainAb: a solution based on tangible and distributed user interfaces to improve cognitive disabilities. In Proc. CHI EA 2013. ACM; 2013, p. 3039-3042. ISBN: 978-1-4503-1952-2 doi>10.1145/2468356.2479605

6.Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P., Smith, M. (2004) Stitching: pen gestures that span multiple displays. In Proceedings of the working conference on advanced visual interfaces. ACM, pp 23–31.

7.Igual, R., Plaza, I., Martín, L., Corbalan, M., Medrano, C. Guidelines to Design Smartphone Applications for People with Intellectual Disability: A Practical Experience. In Ambient Intelligence-Software and Applications; Springer: Berlin, Germany, 2013; pp. 65–69.

8.Ishii, H. and Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 234–241, New York, NY, USA. ACM Press. 19, 20, 27, 36, 73, 96, 187

9.Johanson, B., Hutchins, G., Winograd, T. and Stone, M. PointRight: experience with flexible input redirection in interactive workspaces. In Proceedings of the ACM Symposium on User Interface Software and Technology, 227–234, 2002.

10.Lee, H., Jeong, H., Lee, J., Yeom, K., Shin, H., Park, J. (2008) Select-and-point: a novel interface for multi-device connection and control based on simple hand gestures. In: CHI'08: extended abstracts on Human factors in computing systems. ACM, Florence, Italy, pp 3357–3362. doi:10.1145/1358628.1358857

11.Rekimoto, J. (1997) Pick-and-drop: a direct manipulation technique for multiple computer environments. In: Proceedings of the 10th annual ACM symposium on user interface software and technology. ACM, Banff, AB, Canada, pp 31–39. doi:10.1145/263407.263505

12.Romero, S., Tesoriero, R., González, P., Gallud, J. A., Penichet, V. M. R.: Interactive System based on NFC to manage Georeferenced Documents. Interacción 2009, X Congreso Internacional de Interacción Persona-Ordenador. Barcelona. Septiembre 2009. ISBN-13:978-84-692-5005-1

13.Rukzio, E., Holleis, P. (2010) Projector phone interactions: design space and survey. In: Workshop on coupled display visual interfaces at AVI 2010. Rome, Italy

14.Siio, I., Masui, T., Fukuchi, K. (1999) Real-world interaction using the FieldMouse. In: Proceedings of the 12th annual ACM symposium on user interface software and technology—UIST'99, vol1. ACM Press, pp 113–119. doi:10.1145/320719.322592.

15.Tesoriero, R., Tébar, R., Gallud, J. A., Penichet, V. M. R., Lozano, M.: Interactive EcoPanels: Ecological Interactive Panels based on NFC . Proceedings of the IX Congreso Internacional de Interacción Persona-Ordenador Interacción 2008. ISBN: 978-84-691-3871-7; pp 155-165.

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

17.Vandervelpen, C., Coninx, K. Towards model based design support for distributed user interfaces, Proceedings of the third Nordic conference on Human-computer interaction, p.61-70, October 23-27, 2004, Tampere, Finland [doi>10.1145/1028014.1028023]

18.Want, R., Fishkin, K.P., Gujar, A., Harrison, B.L.: Bridging physical and virtual worlds with electronic tags. In: Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, ACM Press, Pittsburgh, Pennsylvania, United States, 1999.

User-aware Distributed User Interface for Tiled-display Environments

Vít Rusňák
Masaryk University
Brno, Czech Republic
xrusnak@fi.muni.cz

Lukáš Ručka
Masaryk University
Brno, Czech Republic
xrucka@fi.muni.cz

ABSTRACT

Large-sized display walls and tabletops stand for state of the art visualization platforms providing a great opportunity for group collaborative tasks. The integration of multi-touch overlays enables multiple users to interact concurrently with the system. However, continuous user tracking and association of input events with users, which could considerably improve user experience, is still a largely unexplored topic. In this paper, we present a concept of the distributed user-aware interface and a prototype of the modular framework that implements the concept using commodity sensor devices. Although our target platforms are display walls and tabletops, it can be utilized for other collaborative systems.

Author Keywords

tiled-display systems;group collaborative environment;user tracking;distributed user interface

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): Input devices and strategies, Prototyping

INTRODUCTION

Display-wall systems are said to be state of the art platforms for visualization analysis tasks providing high resolution and the performance of cluster computing. The concept of OptIPortals [3], envisions the interactive visual multi-user interface to the OptIPuter cyber-infrastructure which extends these systems with natural user interaction interface [1].

Opposed to single-user platforms such as personal computers, tablets or smartphones, where users interact with the device from a close proximity, these advanced group collaborative systems there can be more *interaction zones* [7] based on the distance from the interactive surface (or displays), positioning accuracy or user's comfort—up-close, mid-air (or distance) and remote. Each of them is beneficial for different types of tasks (precise operations in up-close zone vs. global changes in mid-air zone).

Current prototypes of tiled-display systems allow for multi-user interaction using IR overlay frames but are unable to associate input events with individual users [7]. This behavior results in conflicts during collaborative tasks. For instance, when two users want to reposition an application window in opposite trajectories at the same time, the window is magnified. Correct behavior in such case is the reposition of the window towards the user who touched the surface earlier.

We focus on building come-up-and-use unobtrusive user-aware interface which enables association of users to input events they performed using commodity multi-touch and depth sensors. To achieve this goal, we combine an unobtrusive user tracking (i.e., marker-less and without wearables) in combination with location of a touch point registered on a multi-touch sensor. This could significantly increase a way of multi-user interaction [14] and makes it more real-world behaving. Our approach utilizes commodity multi-touch and depth sensors and enables continuous user tracking and association of them with input events.

The reminder of the paper is organized as follows: first, we overview related work covering user tracking and distinguishing topics; then, we describe the concept of distributed user-aware interface followed by description of the framework and its informal evaluation; finally, we conclude by reviewing of ongoing work prototype limitations and presentations of ongoing and future work.

RELATED WORK

Until recently, large-sized tiled-display systems were controlled remotely from command line or graphical control interfaces by a single person. Integration of commodity multi-touch IR overlay frames integrated in these systems open new possibilities and enabled multi-user group collaborative interaction [10, 20]. However, multi-touch overlays can only detect location of a touch event without specification of the user who did it.

User tracking and distinguishing techniques are heavily explored in small-sized (single-display) tabletops. DiamondTouch [4] uses capacitive coupling through the users touching the sensor but requires them to stay at one position. Use of cheap proximity sensors integrated in tabletop borders enabled user tracking at a close proximity with the surface [1, 16, 19]. The sensors, on the other hand, struggle with retro-reflective materials and the issues with problematic detection of overlapping hands were also reported.

[1] OptIPuter is a global-scale computing grid that enables world-wide collaboration platform that enables effectively share and collaborate with content-rich data over photonic networks. [15]

4th Workshop on Distributed User Interfaces and Multimodal Interaction
DUI'14, July 01 2014, Toulouse, France

http://dx.doi.org/10.1145/2677356.2677660

Motion capture technology is great for prototyping and evaluation of interaction concepts. Optical motion capture systems provide robust method for continuous distinguishing of tangible objects (used e.g., in LambdaTable [9]) or even users in room-sized environments [2]. On the other hand, they might be obtrusive and unnatural for regular use due to mandatory installation of markers on pointing devices and clothing of users.

The advent of MS Kinect depth sensor opened new possibilities for top-view finger and hand tracking and enabled interaction above the tabletop surfaces. Various approaches for finger tracking [13, 12] or arm tracking [6] were published earlier. LightSpace [21] project utilize multiple depth sensors for user tracking in small-room environment with multiple interactive surfaces. Existing approaches either track users as a whole [11] or track only body parts such as arms [6].

In general, individual setups are unique and differ from each other. Although there are toolkits for developing interactive interfaces (e.g., [5, 17, 8, 18]), various tactile and vision-based modalities were used autonomously. On the contrary, our approach is based on composition of different modalities to achieve seamless user-aware interface.

DISTRIBUTED USER-AWARE INTERFACE

We define a *distributed user-aware interface* as an interaction subsystem of a group collaborative environment that combines multiple input modalities (e.g., different types of sensor or pointing devices) and enables unobtrusive user tracking and association of input events with individual users. Such an interface ensures correct response of the system on conflict situations. At the same time, all of these tasks require complex description of multi-touch surface as well as the space around the tabletop or in front of the display wall where users operate. Further, we present three essential features of the distributed user-aware interface: *i)* complex interaction space description, *ii)* user tracking, *iii)* the association of input events with users.

Complex interaction space description

Users can move freely between the up-close and mid-air interaction zones and possibly, there could be more physical sensors or interaction devices (including smartphones or tablets). These assumptions put our focus on complex description of an interaction area where users operate. We consider setups, where a multi-touch surface is the central part of an group collaborative system and a depth sensor captures a space above the interactive surface as seen in Figure 1 and Figure 2. The interaction space can be defined by mutual positions and relations between physical components—e.g., displays, multi-touch sensor(s) or depth sensors. Description of the interaction area also implies the existence of a global coordinate system where each sensor is aware of position and arrangement of other devices integrated within the interactive system.

User tracking

While multi-touch sensors provide accurate positioning of touch events on the surface of up-close interaction zone, depth sensors provide user tracking in the whole interaction space. We suppose users interacting with their hands and fingers only. Due to this premise, our concept combines a user tracking and a palm tracking both realized by depth sensors. The user tracking ensures distinguishing of individuals within the interaction area and the palm tracking allows precise delimitation of the area in which fingers can occur. Consequently, the palm tracking can be used in the mid-air interaction zone for hand gestures. Depth data help avoiding occlusion mistakes when users overlap their hands.

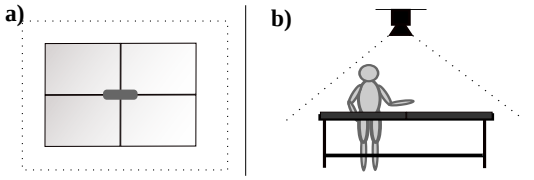

Figure 1. The tabletop use case—a depth sensor is placed above the center of the tabletop; a) top view, b) side view.

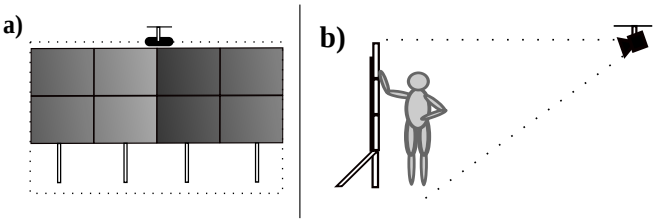

Figure 2. The display wall use case – a depth sensor captures the space in front of the wall; a) top view, b) side view.

The association of input events with users

To provide seamless association of input events with the corresponding user aforementioned features are combined. In principle, it requires to find the intersection of the touch event described by its coordinates within the touch surface with the palm of the corresponding user. Thanks to the global coordinate system, we are able to locate precise positions of the touch surface, interacting users and their palms. Simple comparison of the touch event coordinate position with the palm areas we find a pairing between them. When the touch event is paired, the identification of the corresponding user is appended to its description and could be utilized in further processing in application.

UNIVERSAL INTERACTION FRAMEWORK

We designed and implemented a framework for building seamless interaction interfaces for group collaborative systems based on large tabletops and display walls. Its modular architecture enables utilization in various systems and an easy way for implementing new extensions in future. Extensive context description related to user tracking and distinguishing makes the framework to be used as a basis for further research in group collaborative systems. The framework implements features of distributed user-aware interface. It can be also utilized as an natural multi-user interaction subsystem to existing rendering middleware for tiled-display systems [2].

[2]E.g., SAGE (`http://www.sagecommons.org`) or DisplayCluster(`https://www.tacc.utexas.edu/tacc-projects/displaycluster`).

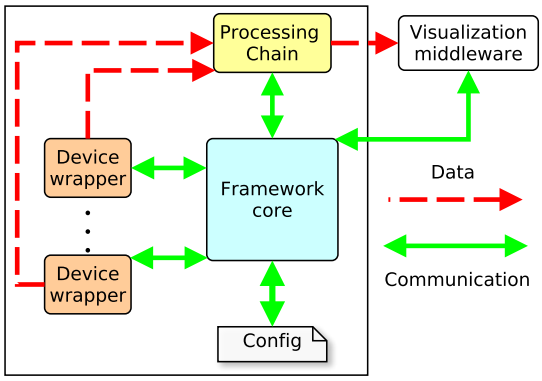

Figure 3. Framework internal structure

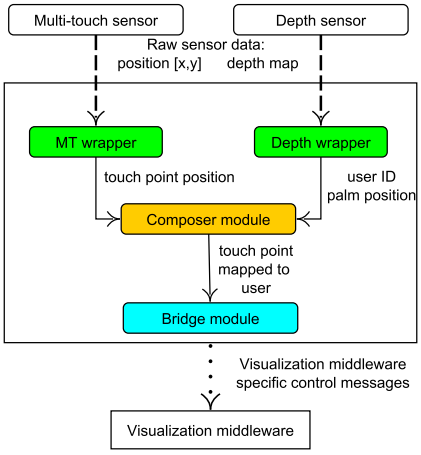

Figure 4. Workflow of a touch event association with a user

Figure 3 shows the internal structure of the framework with communication and data channels. *Framework Core* works as a dispatcher that mediates internal communication between framework modules. It also provides building the DAG based on configuration file and its dynamic update if necessary. *Configuration file* contains description of physical components and their configuration, framework modules and workflow meta-data created during prior calibration of the interactive system. The framework modules are of three categories—wrappers, composers and bridges. The input event processing and the user tracking are realized in *processing chain*, which is represented by a direct acyclic graph (DAG). Vertices represent framework modules and oriented edges represent data flow between them as shown in Figure 4.

Wrappers provide an interface between physical sensors and the unified framework environment. They convert incoming raw data from devices to a framework communication protocol. Each wrapper module represents one physical device. Currently, we distinguish wrappers of two types—multi-touch and tracking wrappers. Multi-touch wrappers perform direct transformation of raw sensor data—i.e., touch point coordinates—to pointer message of the communication protocol. Tracking wrappers perform complex processing of input data related to tracking of users and individual body parts. The depth sensor tracking wrapper performs object detection and skeleton tracking to find and track image blobs describing users or their palms and returns messages describing palms with corresponding user IDs for each frame.

The *composer module* receives messages from the wrappers and provides the association of touch inputs with corresponding image blobs. In principle, the algorithm pairs a touch point with an image blob representing user's palm as described in previous section. The outgoing messages contain positions of input events with auxiliary information about associated user IDs.

Since an internal communication usually varies at various visualization platforms, it is necessary to generate platform-specific control messages on the framework output. The *bridge module* receives messages from the composer and translates framework communication protocol messages to the target visualization middleware.

The communication protocol of the framework is based on Open Sound Control format [3], which provides an efficient binary encoding method for the transmission of arbitrary controller data. Communication protocol messages are of two types describing either static representation of the system (sensor properties, mutual position of devices) or its dynamic behavior (location of users, their palms and touch events). Further description of the protocol is beyond the scope of this paper.

The framework is implemented in C++. We implemented the multi-touch wrapper and two tracking wrappers for depth sensors (for top-view and rear-view tracking), composer module and bridge module to selected distributed visualization middleware (SAGE) so far. The informal evaluation was focused on the accuracy of event-to-user association. We tested the framework in both tabletop and display-wall settings with a single depth sensor tracking. The algorithm associated more than 90 % of registered touch points with users correctly. False associations emerged when users stand close to each other since user tracking algorithms were unable to distinguish them correctly and when one user completely overlapped hands by their heads or torsos. Further improvements in occlusion handling might be realized with involving additional depth sensors in different placements and will be explored in future. Beside the accuracy evaluation, we measured end-to-end latency of the processing chain from the moment when user touched the surface till the response of the system. Measured latency was 311 ± 12 ms (24 repetitions of the task). From the total time, the interaction framework occupied 78 ± 2 ms, SAGE middleware took the rest (including complete background color change). The large portion of latency is caused by SAGE middleware itself and its sources will be investigated in future.

[3] `http://opensoundcontrol.org/specification`

CONCLUSION AND FUTURE WORK

Extending multi-touch sensor with an association of input events with users is the next step towards the real-world interaction experiences in group collaborative systems. In our work, we presented the concept of distributed user-aware interface based on combination of multiple input modalities provided by multi-touch and depth sensors.

We further presented the framework for building unobtrusive DUIs for large-sized visualization systems and shortly summarized the results of its informal evaluation. Although we demonstrated the framework on display-wall and tabletop use cases, it can be easily adaptable for other types of distributed user interface with multi-surface and/or multi-display environments where it could serve as complex multi-modal interaction input layer.

Our initial observations are quite encouraging. The proof-of-concept evaluation showed the processing speed of the framework is high enough to ensure real-time event processing. We will continue with integration of multiple depth sensors to handle occlusion problems. The framework is also ready for an extension towards mid-air interaction zone where users can interact with hand gestures in space or using hand-held devices.

ACKNOWLEDGMENTS

This work has been kindly supported by CESNET Large Infrastructure Project (LM2010005) and internal grants of Masaryk University (MUNI33/022012, MUNI33/192013).

REFERENCES

1. Annett, M., et al. Medusa: A Proximity-Aware Multi-Touch Tabletop. In *Proc. UIST '11* (2011), 337–346.
2. Ballendat, T., Marquardt, N., and Greenberg, S. Proxemic Interaction: Designing for a Proximity and Orientation-Aware Environment. In *Proc. ITS '10* (2010), 121–130.
3. DeFanti, T. A., et al. The OptIPortal, a scalable visualization, storage, and computing interface device for the OptiPuter. *Future Generation Computer Systems 25*, 2 (2009), 114 – 123.
4. Dietz, P., et al. DiamondTouch: A Multi-User Touch Technology. In *Proc. UIST '11* (2011), 219–226.
5. Dragicevic, P., and Fekete, J.-D. Support for Input Adaptability in the ICon Toolkit. In *Proc. ICMI '04* (2004), 212–219.
6. Haubner, N., et al. Detecting interaction above digital tabletops using a single depth camera. *Machine Vision and Applications 24*, 8 (Aug. 2013), 1575–1587.
7. Jagodic, R. *Collaborative Interaction And Display Space Organization In Large High-Resolution Environments*. Ph.D. thesis, University of Illinois, Chicago, 2011.
8. Kim, M., Cho, Y., and Park, K. S. Design and Development of a Distributed Tabletop System Using EBITA Framework. In *Proc. of the 4th International Conference on Ubiquitous Information Technologies Applications*, ICUT '09 (Dec. 2009), 1–6.
9. Krumbholz, C., Kooima, R., and Rao, A. Tangible User Interface Testing on the LambdaTable: A High Resolution Tiled LCD Tabletop. *Interactions of the ACM 14* (2006), 15–22.
10. Leigh, J., and Brown, M. Cyber-Commons: merging real and virtual worlds. *Comm. of the ACM 51*, 1 (2008), 82–85.
11. Migniot, C., and Fakhreddine, A. 3D Human Tracking from Depth Cue in a Buying Behavior Analysis Context. In *Proc. 15th CAIP*, CAIP 2013 (2013).
12. Murugappan, S., et al. Extended Multitouch: Recovering Touch Posture and Differentiating Users Using a Depth Camera. In *Proc. of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, ACM (New York, NY, USA, 2012), 487–496.
13. Ramakers, R., et al. Carpus: A Non-intrusive User Identification Technique for Interactive Surfaces. In *Proc. of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, ACM (New York, NY, USA, 2012), 35–44.
14. Schmidt, G., et al. A Survey of Large High-Resolution Display Technologies, Techniques, and Applications. In *Proc. of the IEEE Virtual Reality Conference 2006*, IEEE (2006), 223–236.
15. Smarr, L. L., et al. The OptIPuter. *Comm. of the ACM 46*, 11 (Nov. 2003), 58–67.
16. Tanase, C. A., et al. Detecting and Tracking Multiple Users in the Proximity of Interactive Tabletops. *Advances in Electrical and Computer Engineering 8*, 2 (2008), 61–63.
17. Tuddenham, P., and Robinson, P. T3: A Toolkit for High-Resolution Tabletop Interfaces. In *Proc. of the 2006 ACM Conference on Computer Supported Cooperative Work*, CSCW '06 (2006), 3–4.
18. van de Camp, F., and Stiefelhagen, R. GlueTK: A Framework for Multi-modal, Multi-display Human-machine-interaction. In *Proc. of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, ACM (New York, NY, USA, 2013), 329–338.
19. Walther-Franks, B., et al. User Detection for a Multi-touch Table via Proximity Sensors. In *Proc. of the ACM International Conference on Interactive Tabletops and Surfaces 2008*, ITS '08, ACM (2008), 2 pp.
20. Westing, B., et al. Integrating Multi-touch in High-resolution Display Environments. In *State of the Practice Reports*, SC '11 (2011), 8:1–8:9.
21. Wilson, A. D., and Benko, H. Combining Multiple Depth Cameras and Projectors for Interactions on, Above and Between Surfaces. In *Proc. of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, ACM (New York, NY, USA, 2010), 273–282.

Context-sensitive and Collaborative application for Distributed User Interfaces on tabletops

Amira Bouabid
LAMIH-UMR CNRS 8201,
Univ. of Valenciennes, France
UR SETIT, University of Sfax-Tunisia PB 1175 -3038
amira.bouabid@gmail.com

Sophie Lepreux, Christophe Kolski
LAMIH-UMR CNRS 8201,
Univ. of Valenciennes, France
{firstname.lastname}@univ-valenciennes.fr

Clémentine Havrez
LAMIH-UMR CNRS 8201,
Univ. of Valenciennes, France
Play Research Lab – CCI Grand-Hainaut, France
clementine.havrez@univ-valenciennes.fr

ABSTRACT
This paper focuses on collaborative work on interactive tabletops. To optimize the travel time of team members (remote workplace, telecommuting, and so on), collaborative work is now often remotely done. This brings many user interfaces issues between distributed platforms of each member. In the domain of context-sensitive user interfaces, which aims at an adaptation to the users, the platforms and the environment, context models have been proposed in the literature. We propose, in this paper, a context model for distributed applications centered on collaboration and interactive tabletops. The proposed model is validated by a distributed application, which is developed on two interactive tabletops with tangible interaction; these tabletops are equipped with RFID technology. This application, which has educational purposes, highlights the collaborative aspect and exchanges between remote users. The paper ends with a conclusion and several perspectives.

Author Keywords :
Context; Model; Distribution; Collaboration; Interactive tabletop; Tangible object; RFID.

ACM Classification Keywords :
H.5.2; H.5.3. Information Interfaces and Presentation (e.g.HCI): User Interfaces; Group and Organization Interfaces

INTRODUCTION
With the development of remote work on various platforms, user interfaces have evolved. In the 2000s, an application was intended for a single end user and worked on a single platform in a single environment. Nomadic applications engendered researches about the adaptation of applications to the various types of platform (e.g. PDA, Smartphone). Thus, the consideration of the interaction context became essential for their adaptation. The introduction of interaction surfaces with a more important size has led to applications available to multiple users interacting in a single environment and on the same support [13]. The improvement of the network capacities has brought new remote collaborative applications based on a variety of platforms [9]. This led to treat the distribution of user interfaces. Collaboration became possible between users in the same interaction context or in different contexts.

This paper focuses on collaborative interactions on interactive tabletops. Section 2 presents the state of the art on collaboration. Some types of collaboration are presented with an example for each type. Section 3 is dedicated to present our proposition. It is a context model which supports collaboration between remote or collocated users. In section 4, a case study on a pedagogical application adapted from [10] is shown on interactive tabletops with a distributed tangible interaction. The paper ends with a conclusion and research perspectives.

COLLABORATION AND INTERACTIVE TABLETOPS
The high quality networks coupled with the arrival of efficient systems increases the remote collaborative work. Many researches are carried out in the field of CSCW since [7]. Some researches dealing with collaboration are focused on many domains, such as education [11], information retrieval on the web [6, 5], and so on. In this paper, we particularly focus on collaboration via interactive tabletops according to two types of collaboration: (1) collaboration where the team is in the same workspace (co-located) and (2) remote collaboration (distributed).

The technology of interactive tabletops allows sharing information and tasks between team members located face to face or side by side thanks to their large surface. It is therefore particularly suitable for co-located collaboration. Interactive tabletops provide the ability to handle multiple real and / or virtual objects depending on the capture technology used.

The DTLens system [4] uses an interactive tabletop in a multi-user environment. It is based on a zoom-in-context that enables group exploration of spatial data with multiple

July 01 2014, Toulouse, France

http://dx.doi.org/10.1145/2677356.2677661

individual lenses on the same direct-touch interactive tabletop.

Most of the works, which aim to realize distributed collaborative systems, seek to find a good remotely data presentation to perform a task or a common work between the different members of a geographically dispersed team [2]. Balakrishnan and colleagues are looking for solutions to solve complex problems by sharing remote data visualizations [1]. They conclude that the more members have access to a complete visualization of their data, the more they can be fruitful and provide solutions to problems. Another research encourages task sharing and data personalization in a collaborative framework: Brennan and colleagues adopt an approach that requires the explicit sharing and merging views of data during distributed analysis [2]. In this way team members can first work separately on sub-tasks, and then gather all the ideas in a common view that takes charge of the construction of the common sense. The Keel system [8], which is dedicated to analysts of distributed data, offers a solution for group-work. Each member works in his/her personal workspace and offers the data considered as appropriated in the common workspace. This method allows the implicit sharing of information.

Collaboration takes place in a collaborative environment. This environment can support two types of interaction: synchronous or asynchronous [14]. The types depend on the goal of the application and the manner in which several collaborators interact. [15] mentions a type of interaction called semi-synchronization. We face this situation when it is possible for some users to see and interact with the work of other absent users. This is possible thanks to traceability of past activities.

In the case of the tangible interaction, the concept of Tangiget was proposed in [13]. Tangigets are tangible objects which support the interaction on one or several tabletops. They are divided into six categories: (1) Control objects of the application, (2) Context objects of the application, (3) Control of the interface objects, (4) Communication objects, (5) Coordination objects (6) Creation objects. Among these categories, we note that some Tangigets support communication and coordination, two activities required during collaboration. They are exploited in our proposition.

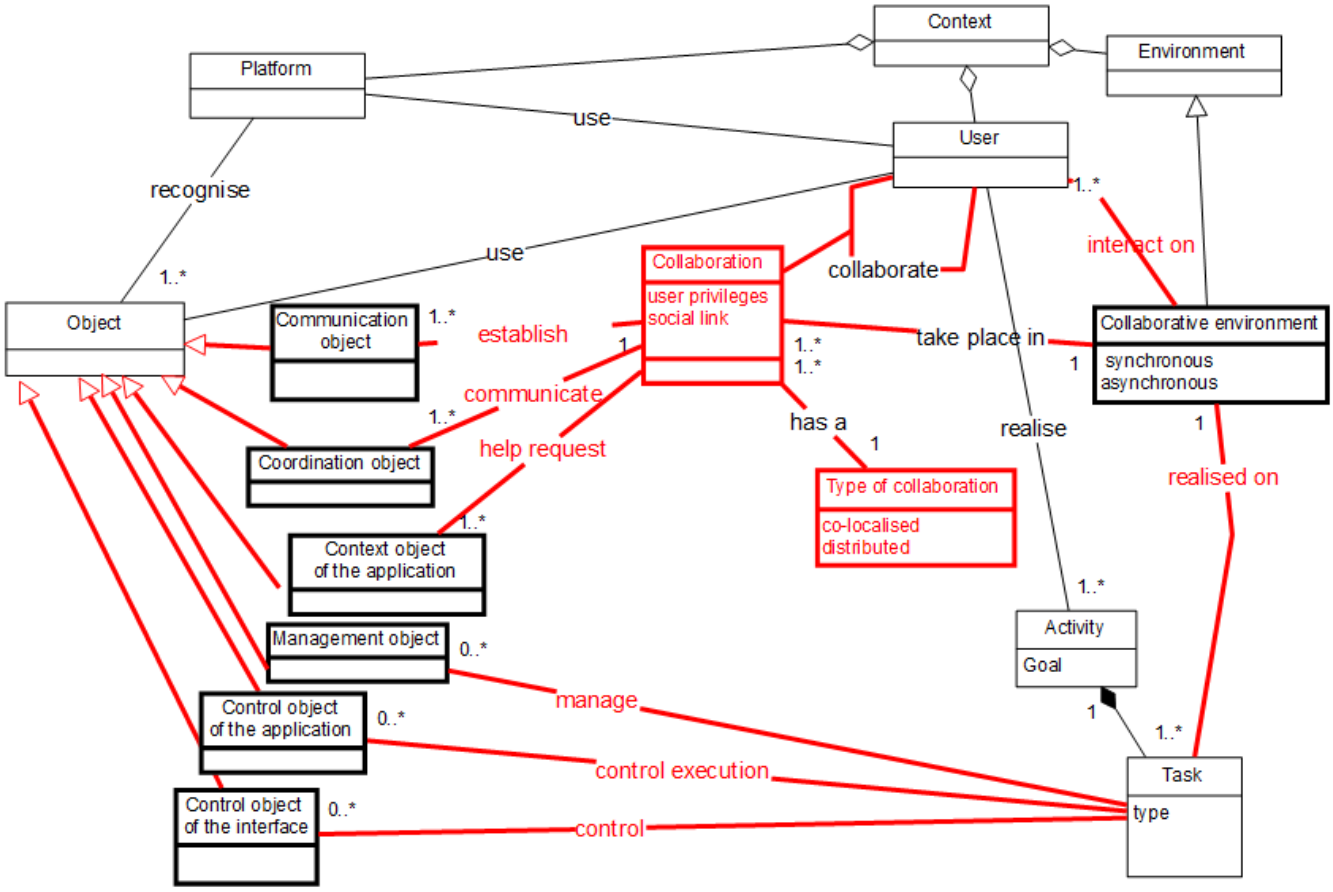

Figure 1. Context model focused on collaboration properties

PROPOSITION

Our proposition is based on the work of Calvary and colleagues [3]: they define the context as a triplet <user, platform, environment>. This model was adapted by [9] to focus on interactions on interactive tabletops in a distributed context. Our goal is to incorporate in this previous context model [9] the notions linked with collaboration presented above and the different types of Tangigets used for collaboration between remote tables.

Main evolutions are shown in bold in the Figure 1. Concerning the **user** part, the most significant changes are the adding of a reflexive relationship between users and of a *Collaboration* class. This class contains the properties of the collaboration between users. It sets the *privileges* granted to each user according to his or her role in the application and the social link between the users who collaborate. Another class named *Type of collaboration* is added. This class specifies the type of collaboration between users. An *Interact on* relationship is added between user and collaborative environment in which he/she interacts. Concerning the **environment** part, a relationship named *realise on* is created between tasks and collaborative environment. Another relationship named *takes place on* specifies the environment in which the collaboration is performed. Concerning the **platform** part, the several categories of Tangigets were introduced in the model through six new classes. They are connected either to the *Collaboration* class or to the local tasks.

CASE STUDY: COLLABORATIVE APPLICATION

The "learning color" application (previously proposed and described in [10]) is developed with a collaborative goal, following a scenario inspired from [9] in which a child and parents interact in a distributed manner. This distribution is achieved through a multi-agent system developed with JADE [12]. The child is doing his or her color learning exercise on one tabletop. Parents are in another room where they also have a tabletop. The two tabletops are connected. There could be several users on each table. The scenario which aims to generate collaboration is: "*The child has a difficulty; he or she wants to seek assistance from parents who are distant but also possess an interactive tabletop. Parents wish to let the child work independently but also want to control the work done by the child without disturbing him or her with their presence.*"

A remote collaboration between users is possible. The child can ask parents for help. The Parents can have a visualization of child labor. The picture shown in Figure 2 shows the distributed application on both *TangiSense* tabletops equipped with RFID technology (developed by the RFidées Company, http://www.rfidees.fr/). On one side (child), we see tangible objects placed by the child and their virtual feedback. On the right side, the parents have a view of objects put by the child. The Parents in this way can monitor the progress of the game on the tabletop without having to move.

In Figure 3, we present the object diagram of the *Request for assistance* collaborative use case. We see in this figure the Tangigets used for the realization of a distributed task between a child and his or her father.

Figure 2. Picture of the distributed application showing the child's view on the left side (with tangible objects) and the parents' view on the right side (with virtual objects as feedback).

CONCLUSION AND PERSPECTIVES

In this paper, we focused on the concept of collaboration between users on interactive tabletops involving distributed user interfaces. Two types of collaboration were considered: co-located collaboration and distributed collaboration. Based on this notion of collaboration exploiting a distribution of user interfaces, a context model has been proposed, built on the distributed interfaces and integrating collaboration features on tangible inter-connected tabletops. A case study involving two RFID interactive tabletops with tangible objects implements this model. As research perspectives we aim to diversify the types of distributed platforms and to implement variants of the tangigets proposed in [13].

ACKNOWLEDGMENTS

This research was partially financed by the French National Research Agency (ANR IMAGIT project ANR-10-CORD-017, partners LIG & RFIdees). We would like to thank the Region Nord-Pas de Calais, France for its support, as well as E. Adam, S. Kubicki, Y. Lebrun & R. Mandiau.

REFERENCES

1. Balakrishnan, A. D., Fussell, S. R., Kiesler, S., and Kittur, A. Pitfalls of information access with visualizations in remote collaborative analysis. In *Proc. of CSCW* (2010), 411–420.
2. Brennan, S. E., Mueller, K., Zelinsky, G., Amakrishnan I., Warren, D. S., and Kaufman, A. Towards a Multi-Analyst, Collaborative Framework for Visual Analytics. In *Proc. of VAST*, (2006), 129–136.
3. Calvary, G., Demeure, A., Coutaz, J., and Dâassi, O. Adaptation des Interfaces Homme-Machine à leur contexte d'usage. *Revue d'intelligence artificielle 18*, 4 (2004), 577–606.

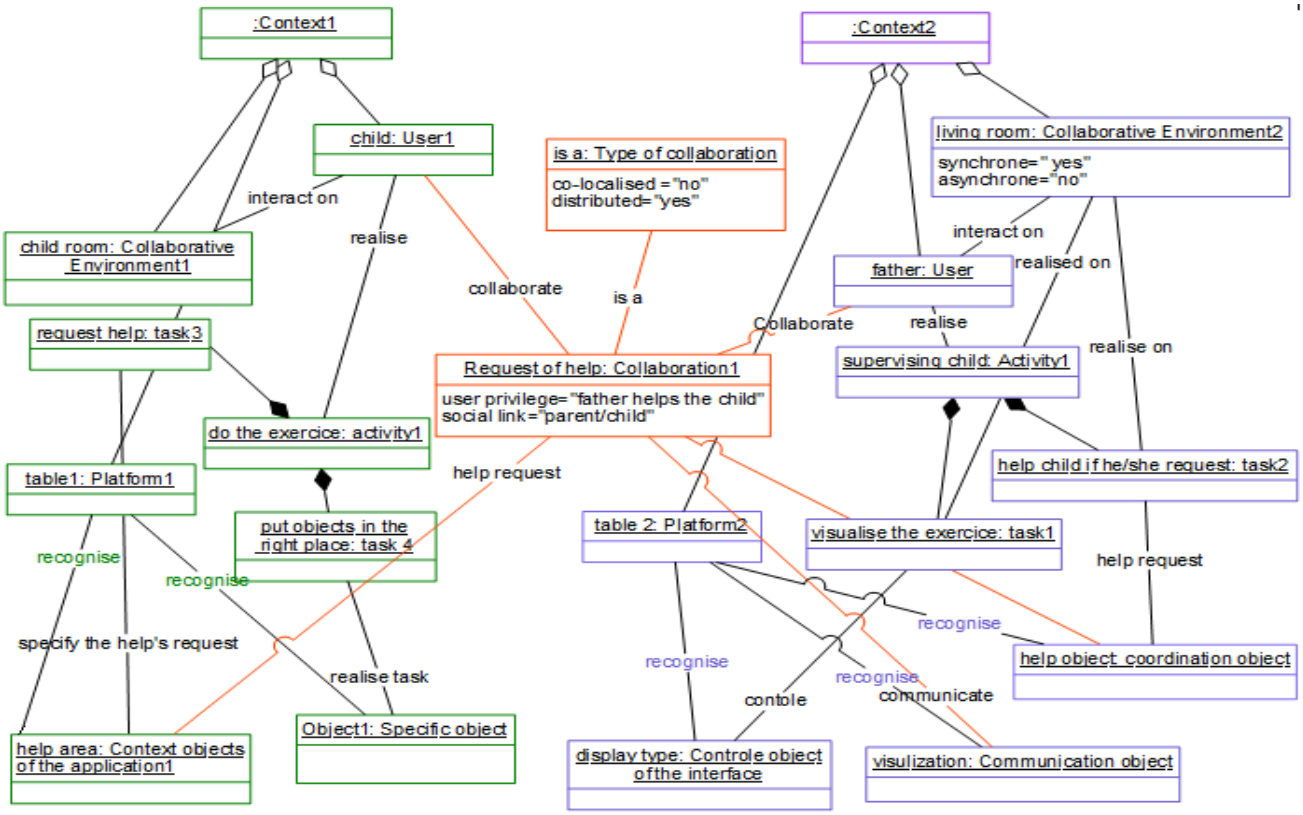

Figure 3. Object diagram for the use case: *request for assistance*

4. Forlines, C. and Shen, C. DTLens: Multi-user Tabletop Spatial Data Exploration. In *Proc. of UIST* (2005), 119–122.
5. Hansen, P., and Jarvelin, K. Collaborative Information Retrieval in an Information-Intensive Domain. *Information Processing and Management* 41, 5 (2005), 1101-1119.
6. Isenberg, P., and Fisher, D. Collaborative Brushing and Linking for Co-located Visual Analytics of Document. *Collections IEEE-VGTC Symposium on visualization*, Volume 28 (2009).
7. Johansen, R. *Groupware-Computer Support for Business Teams*. The Free Press (1988).
8. Keel, P. E. Collaborative Visual Analytics: Inferring from the Spatia Organization and Collaborative Use of Information. In *Proc. of VAST*, (2006), 137–144.
9. Kubicki, S., Lepreux, S., and Kolski, C. Distributed UI on Interactive tabletops: issues and context model. In M.D. Lozano, J.A. Gallud, R. Tesoriero, V.M.R. Penichet (Eds.), Distributed User Interfaces: Collaboration and Usability, Springer, 27-38.
10. Kubicki, S., Lepreux, S., Kolski, C. Evaluation of an interactive table with tangible objects: Application with children in a classroom. *Proceedings 2nd Workshop on Child Computer Interaction*, CHI'2011, 70-73 (2011).
11. Large, A., and Beheshti, J. The web as a classroom resource: Reactions from the users. *Journal of the American Society for Information Science*, 51(12), 1069–1080. (2000).
12. Lebrun, Y., Lepreux, S., Kolski, C., and Mandiau, R. Combination between Multi-Agent System and Tangigets for DUI Design on several Tabletops. *3rd Workshop on Distributed User Interfaces*, pp. 54-57, (2013).
13. Lepreux, S., Kubicki, S., Kolski, C., and Caelen, J. From Centralized interactive tabletops to Distributed Surfaces: the Tangiget concept: *International Journal of Human-Computer Interaction*, 28 (11), 709-721 (2012).
14. Yang, S., Lu, Y., Gupta, S., and Cao, Y. Does context matter? The impact of use context on mobile internet adoption. *International Journal of Human-Computer Interaction, 28*(8), 530–541. (2012).
15. Yarosh, S., Cuzzort, S., Müller, H., and Abowd, G. D. Developing a Media Space for Remote synchronous Parent–Child Interaction. In *Proc. IDC* (2009)

Fault-Tolerant User Interfaces for Critical Systems: Duplication, Redundancy and Diversity as New Dimensions of Distributed User Interfaces

Camille Fayollas, Célia Martinie, David Navarre, Philippe Palanque, Racim Fahssi
Institute of Research in Informatics of Toulouse, University of Toulouse
Interactive Critical Systems (ICS)
118 route de Narbonne, 31042 Toulouse Cedex 9, France
lastname@irit.fr

ABSTRACT
Assuring that operators will be able to perform their activities even thought the interactive system exhibits failures is one of the main issues to address when designing and implementing interactive systems in safety critical contexts. The zero-defect approaches (usually based on formal approaches such as [5]) try to guarantee that the interactive system will be defect free and thus will be fully functional during operations. While this has been proved a good mean for removing faults and bugs at development time, natural faults (such as bit-flips due to radiations) are beyond their reach. To address this kind of faults three main approaches are available: include fault tolerant mechanisms such as the ones offered by self-checking user interfaces [7], reconfigure the user interface and the interaction techniques so that part of the operations can still take place [4] or duplicate interactive systems and their user interfaces so that if one system fails, operation can still take place using a redundant one. This position paper investigates this last option connecting this redundancy approach to the concept of Distributed User Interfaces that provide a generic framework for understanding both their advantages and their limitations.

Author Keywords
Model-Based approaches, formal description techniques, fault-tolerance, interactive software engineering, distributed user interfaces.

ACM Classification Keywords
D.2.2 [Software] Design Tools and Techniques - Computer-aided software engineering (CASE), H.5.2 [Information Interfaces and Presentation]: User Interfaces - Interaction styles.

INTRODUCTION
Systems which support the management of complex tasks and of a huge amount of information usually require Distributed User Interfaces (DUIs) where information display may appear on various output devices and input of information can be performed by the operators using several input devices.

DUI '14, July 01 2014, Toulouse, France
Copyright 2014 ACM 978-1-60558-724-0/14/07…$15.00
http://dx.doi.org/10.1145/2677356.2677662

Several definitions of distributed user interfaces co-exist and present complementary viewpoints. For Vanderdonckt [8], a UI distribution "concerns the repartition of one or many elements from one or many user interfaces in order to support one or many users to carry out one or many tasks on one or many domains in one or many contexts of use, each context of use consisting of users, platforms, and environments". Another definition proposed by Elmqvist [2] identifies several dimensions for the distribution of UI components: input, output, platform, space and time. Demeure et al. [1] also propose a reference framework (called 4C) to analyse DUIs, which is composed of four concepts: computation, coordination, communication and configuration. Villanueva et al. [9] also proposes a metamodel to classify UIs as Divisible/Undivisible and Distributable/Undistributable. In this position paper we advocate for a task and context based approach for the design of DUI. For this reason the contribution fits better with the first definition as it explicitly and directly binds the tasks and context of use to the DUI.

Assuring that operators will be able to perform their activities even thought the interactive system exhibits failures is one of the main issues to address when designing and implementing interactive systems in safety critical contexts. The zero-defect approaches (usually based on formal approaches such as [5]) try to guarantee that the interactive system will be defect free and thus will be fully functional during operations. While this has been proved a good mean for removing faults and bugs at development time, natural faults (such as bit-flips due to radiations) are beyond their reach. To address this kind of faults three main approaches are available: include fault tolerant mechanisms such as the ones offered by self-checking user interfaces [7], reconfigure the user interface and the interaction techniques so that part of the operations can still take place [4] or duplicate interactive systems and their user interfaces so that if one system fails, operation can still take place using a redundant one.

This position paper investigates this last option connecting this redundancy approach to the concept of Distributed User Interfaces that provide a generic framework for understanding both their advantages and their limitations.

Next section presents briefly the concepts behind user interface redundancy and its relationship with distributed

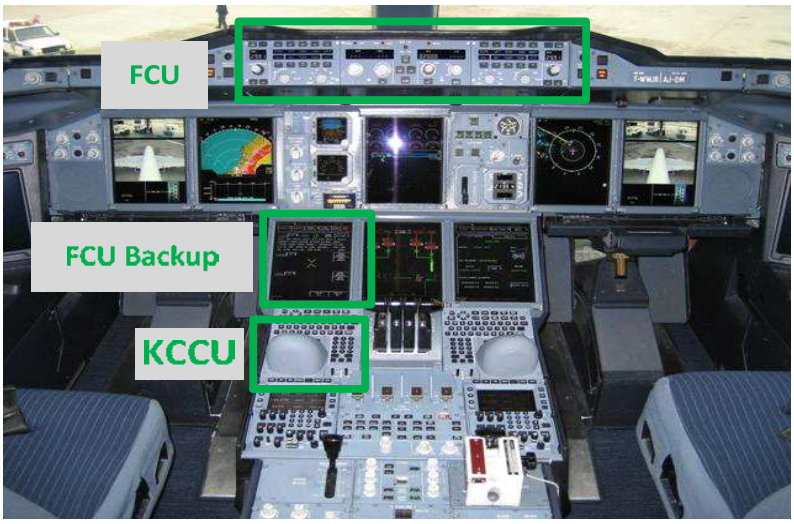

Figure 1. The two possible means to control flight heading within the A380 interactive cockpit, one using the FCU and the other using the FCU Backup application and the KCCU

user interfaces. The following section presents on a case in the avionics domain how such redundancy has been introduced in the past and how the requirement of diversity brings interesting challenges both in terms of design and in terms of use of these interfaces. Last section briefly summarizes the paper and highlights some research questions that could be discussed at the wokshop.

REDUNDANT USER INTERFACES AND DUI

In order to be efficient, fault-tolerance (i.e. guaranteeing the continuity of service), provides **duplicated user interfaces** for the command and control of a single system. This ends up with **redundant user interfaces**. If those interfaces are built using the same processes and offer the same interaction techniques, it is possible that a single fault could trigger failures in both user interfaces. This could be the case for instance when using the idea of cloning the UI as proposed by [10]. In order to avoid such common points of failure the redundant user interfaces must ensure **diversity**. Diversity can be guaranteed if the user interfaces have been developed using diverse means such as different programming languages, different notations for their specifications, executed on top of different operating systems, exploiting different output and input devices, … Such diversity is only efficient if the command and control system offers confinement mechanisms avoiding cascading faults i.e. the failure of one user interface triggering a failure in the duplicated one.

Such fault tolerant basic principles raise conflicting design issues when applied to user interfaces. Indeed, diversity

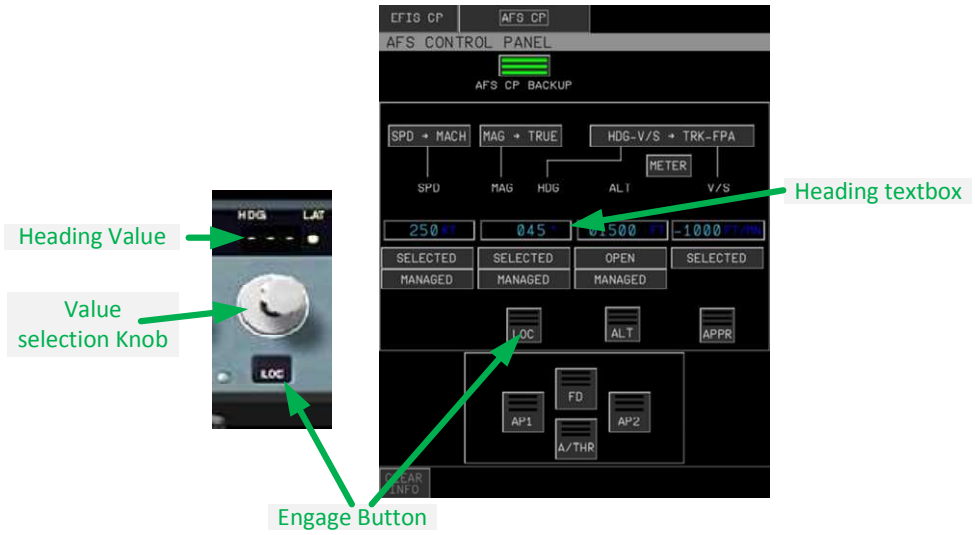

Figure 2. Heading selection.

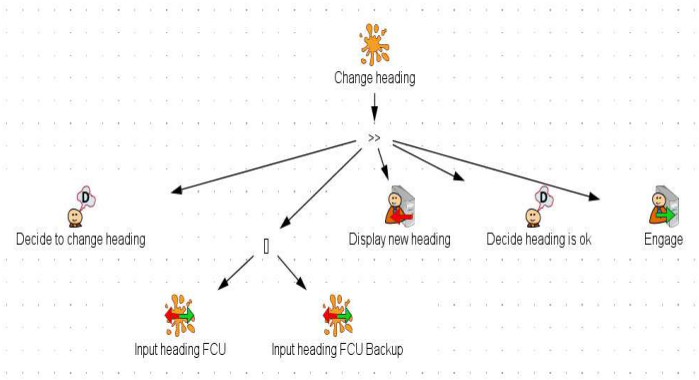

Figure 4: High level tasks involved in changing the heading.

requires the user interfaces to be very different in terms of structure, content and in terms of interaction techniques they offer even though they must guarantee that they support the same tasks and the same goals of the operators. Another aspect is that they must be located in different places in the system i.e. distributed as this is one of the most efficient way of ensuring confinement of faults.

In that context, distribution of user interface does not concern the presentation of complementary information in different contexts (as presented in [3]) but the presentation of redundant information in those contexts.

In terms of design it is important to be able to assess that the various user interfaces make it possible to the operators to reach their goals. Beyond that, it is also important to be able to assess the relative complexity and diversity of these interfaces in order to be sure that operations will not be drastically degraded when a redundant user interface has to be used following the failure of another one. In order to answer these design questions we propose the use of tasks models to describe the operators activites when interacting with those redundant and diverse user interfaces. Those tasks models can, in turn, be analysed to assess their relative complexity as we have done to assess relative complexity of tasks depending on fault-tolerance mechanisms [6].

CASE STUDY

The case study presents (in the area of aircraft cockpits) examples of redundant user interfaces. More precisely we present in the context of the cockpit of the A380 aircraft two redundant ways of using the autopilot to change the heading of the aircraft. One is performed using the electronic user interface of the Flight Control Unit while the other one exploits the graphical user interface of the Flight Control Unit Backup interface.

Figure 1 presents a picture of the A380 interactive cockpit, the heading control means are highlighted.

Figure 2 presents a zoomed view on the two ways to control the heading of the flight. On the left side of the figure, the editing of the heading is performed using a physical knob which may be turned to set a value, engaged by pressing the physical LOC push button. On the right side, the heading is set up using the keyboard of the KCCU and engaged using the dedicated software LOC push button.

Figure 4 presents an overview on the pilot's activity while setting a new heading for the flight.

The task model representing how a pilot modifies the heading value using the FCU Backup application is presented on Figure 3. The set of actions to be performed in order to modify a parameter (goal "Input heading FCU Backup") has to be performed in sequence (operator >>). The pilot edits the heading (input task "input value" after reaching the input device "Reach KCCU"). This action on the input device leads to an update in the system state (system task "update value"). Lastly, as any aircraft parameter is important (whatever its level of criticality is), the pilot at least verifies the value entered (task "perceive value" followed by "analyze the value is OK"), even though

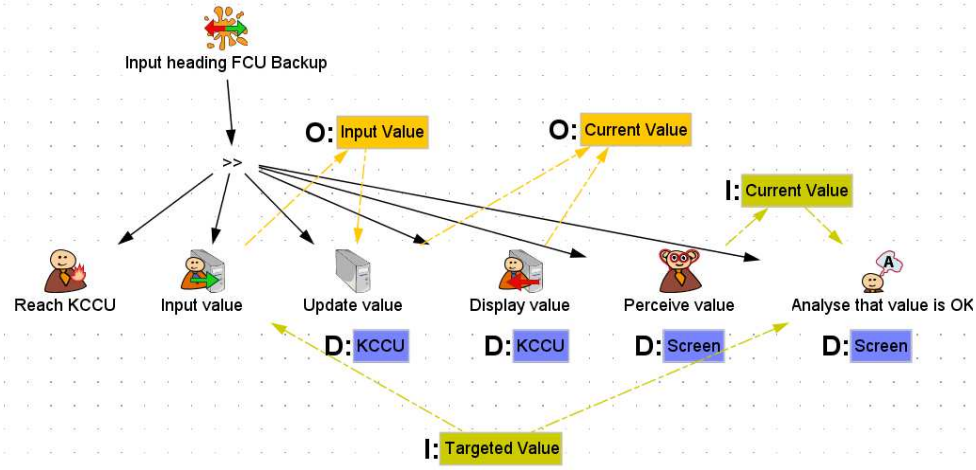

Figure 3: Tasks involved in the editing of the heading value using the FCU Backup application

no formal monitoring activity is required for non-critical interaction. It is important to note that objects represent data in the system while information represents data in the head of the user. Modifying a parameter consists precisely in moving information from the head of the user to the system.

The task model representing how a pilot modifies the heading value using the FCU is presented on Figure 5. These two task models correspond to the same operations that have to be performed by the flying crew to change the heading of the aircraft using the autopilot. It is important to note that there are other additional means to perform the same task (for instance controling directly the aircraft using the sidestick) that are not presented here.

CONCLUSION

The position paper has presented the issues raised by the duplication of user interfaces in order to improve the dependability of command and control systems in safety critical contexts. We have shown that task models can provide useful means to ensure that the user interfaces are diverse enough (via the representation of input and ouptut devices information on the task models) and that they allow operators to reach the same goals. Beyond that the tasks can be used to assess the relative complexity of the redundant interfaces thus providing ways of planning corresponding training of operators.

We believe that this way of assessing diversity, duplication and redundancy of user interfaces could be used more broadly in the larger context of distributed user interfaces.

REFERENCES

1. Demeure, A., Sottet, J.S., Calvary, G., Coutaz, J., Ganneau, V., Vanderdonckt, J. The 4C reference model for distributed user interfaces. In Proceedings of the International Conference on Autonomic and Autonomous Systems, pages 61-69, IEEE Explore, Piscataway, 2008.
2. Elmqvist, N. Distributed User Interfaces: State of the Art. In J.A. Gallud et al. (eds), Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem, Human-Computer Interaction Series, pages 1-12, 2011, Springer-Verlag, 2011
3. Martinie C., Navarre D., Palanque P. A multi-formalism approach for model-based dynamic distribution of user interfaces of critical interactive systems. Int. J. Hum.-Comput. Stud. 72(1): 77-99 (2014)
4. Navarre, D., Palanque, P., Basnyat, S., (2008) Usability Service Continuation through Reconfiguration of Input and Output Devices in Safety Critical Interactive Systems. The 27th International Conference on Computer Safety, Reliability and Security (SAFECOMP 2008) LNCS 5219, pp. 373–386.
5. Navarre, D., Palanque, P., Ladry, J., and Barboni, E. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability, ACM ToCHI, 2009, V. 16, 4, pp. 1-56
6. Palanque P., Martinie C., Fabre J-C., Déléris Y., Navarre D. and Fayollas C. An approach for assessing the impact of dependability on usability: application to interactive cockpits. European Dependable Computing Conference, 2014, Springer Verlag LNCS, pp.45-55.
7. Tankeu-Choitat, A., Navarre, D., Palanque, P., Deleris, Y., Fabre, J.-C., Fayollas, C. Self-checking components for dependable interactive cockpits using formal description techniques. In Proc of 17th IEEE Pacific Rim Int. Symp. on Dependable Computing (PRDC 2011), 10p
8. Vanderdonckt, J. Distributed User Interfaces: How to Distribute User Interface Elements across Users, Platforms, and Environments. In Proceedings of XIth Congreso Internacional de Interacción Persona-Ordenador Interacción'2010 (Valencia, 7-10 September 2010), J.L. Garrido, F. Paterno, J. Panach, K. Benghazi, N. Aquino (Eds.), AIPO, Valencia, 2010, pp. 3-14, Keynote address.
9. Villanueva,P. G., Tesoriero, R., Gallud, J. A. Revisiting the Concept of Distributed User Interfaces. In Distributed User Interfaces: Usability and Collaboration, Human–Computer Interaction Series, 2013, Springer-Verlag London, pp. 1-15.
10. Villanueva,P. G., Tesoriero, R., Gallud, J. A. 2013. Distributing web components in a display ecosystem using Proxywork. In Proceedings of the 27th International BCS Human Computer Interaction Conference (BCS-HCI '13), 2013, Steve Love, Kate Hone, and Tom McEwan (Eds.). British Computer Society, Swinton, UK, UK, Article No. 2

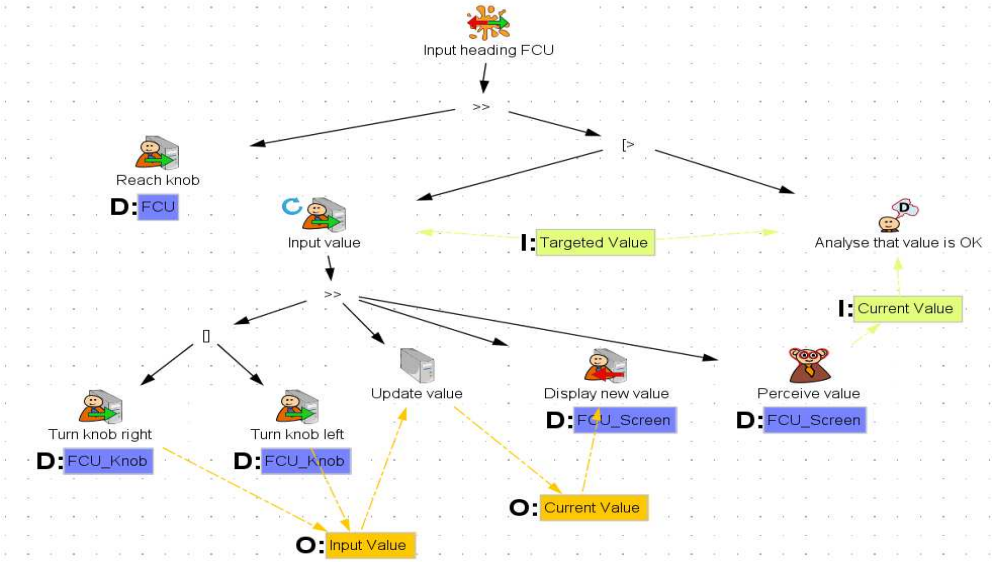

Figure 5. Tasks involved in the editing of the heading value using the FCU

Improving Surgery Operations by means of Cloud Systems and Distributed User Interfaces

Habib M. Fardoun, Abdullah AL-Malaise AL-Ghamdi
King Abdulaziz University
Jeddah, Saudi Arabia
{hfardoun, aalmalaise}@kau.edu.sa

Antonio Paules Cipres
European University of Madrid
Madrid, Spain
apcipres@gmail.com

ABSTRACT
Surgical interventions are usually performed in an operation room; however, access to the information by the medical team members during the intervention is limited. While in conversations with the medical staff, we observed that they attach significant importance to the improvement of the information and communication direct access by queries during the process in real time. It is due to the fact that the procedure is rather slow and there is lack of interaction with the systems in the operation room. These systems can be integrated on the Cloud adding new functionalities to the existing systems the medical expedients are processed. Therefore, such a communication system needs to be built upon the information and interaction access specifically designed and developed to aid the medical specialists.

Author Keywords
Surgical interventions; Model View Controller; View Cloud Controller; Distributed User Interfaces; Cloud Computing; Google Glass

ACM Classification Keywords
B.4.1. Data Communications Devices. *Receivers, Transmitters*

B.4.2. Input/Output Devices. *Image Display, Voice*

C.2.0. General. *Data communications.*

INTRODUCTION
This paper proposes a solution for the medical professionals. During a surgery, a surgeon needs to consult or acquire visual information about different medical tests and results before the operation. Nowadays in Spain such activities and information acquisition are facilitated and supported by hard copies or engaging a medical application available to the sanitary centres and related communities.

It is important for the hospital officers to manage the information correctly, facilitating medical expedients queries directed to experts and providing them with an index for monitoring medical treatments and surgical interventions. Due to the fact that this specific problem is of great magnitude, the focus in this paper is isolated on the pre-operatory tests and associated necessary procedures conducted before the surgical intervention.

This proposition suggests the use of a platform on the Cloud for medical tests processing supporting surgeons' queries. This platform facilitates faster and more precise information access; therefore, accessibility is aided via different interaction types, such as:

- Voice commands
- DUIs, where the assistant checks the medical tests and results, and passes them to the surgeon into the operation room.
- A mixed system of Voice commands and DUIs. The surgeon says the commands so to obtain the query from the Cloud on the specific interface the surgeon has access to.

A platform with such characteristics is located on a system to support on-line queries, so the specialists can help each other in real time when they conduct an operation. Thus, if an operation is planned, the surgeon is able to indicate that the operation is, for example, becoming complex and there is a need for more specialists in the operation room immediately. Such need requires the design and development of a parallel system for the messages retransmission of the operation activities as such as well as the machines' sensors in the operation room.

Here, there is a challenge for the hardware used for the new interaction and tangible systems, like tablets and DUIs, so to access information on the Cloud by both voice

http://dx.doi.org/10.1145/2677356.2677663.

commands and the assistants' help during the surgical process.

GOAL AND NECESSITIES

This paper proposes a technical solution aiming at accessing specialised information into the operations room in real time. Such information must be accessed fast and the medical tests results performed must be stored in associated medical communities' information systems. These necessities suggest a mixed system combining tangible interfaces, DUIs and new devices, such as Google Glass [1].

STATE OF ART

In Spain, the results from medical tests are stored in systems belonging to autonomous Medical Communities. Country's legislation does not impose results digitalisation and storage in a centralized system for every Spanish patient; the medical expedients are stored in both hard copies and digital format.

The specialists, in this case, the surgeons, prepare the surgeries conducting their own expedients managed by an administrative person who works for them; this person leaves this surgery request on a waiting list. This request is stored in the medical expedient space through a referral note produced by the specialist.

The specialist prepares all necessary data for the surgery in a non-established way; usually data is stored in a DVD or a piece of paper. With this information s/he reviews the surgery elements for decision-making such as actions required before entering the operations room. Thus, the surgeon cannot have access to such data results during the operation because it is unhygienic and indicates high risk for the patient.

Access to such data using mobile devices facilitates information access in a hygienic and sterilized way, in addition to accessing wide online resources banks during the surgical intervention.

There are different technological solutions in the current market, ranging from a voice recognizer for the redaction of medical expedients [2] to patients' monitors aiding in decision-making [3].

System interoperability following the suggested characteristics can be hosted on the Cloud taking advantage of the different communication methodologies. Our previous research on distributed graphical user interfaces [4], e-health organization for hospitals [5] and interaction methods in educational experiences, point to concrete results suggesting that collaboration and interaction are the main axis for methodologies and tools [6][7].

Nowadays, hardware devices evolve quickly allowing new type of interactions; thus, there are the 12" tablets and clocks with Internet connection that allow the notifications delivery [9] in the market already. It is necessary to adapt these technologies towards the suggested medical scope so to facilitate the information flow inside the operations room during the surgery. Another solution can be the augmented reality devices like the Google Glass [1], so to provide the information needed during the surgical process. Usually, medical information consisted of reports and images in high resolution so the screen size can be a significant impediment for immediate use.

As for the ways the users access information, in the Tangible Graphical User Interfaces, the users interact with screen objects by gestures. They interact with the device to complete tasks through finger movements exchanging information with other objects on the screen. Another important factor is that tangible interfaces usually are DUI based as the user interprets exchanging information as a moving object in an area on the personalised tangible interface. [10]

SCENARIO AND USABILITY

Such interactions requirements are built upon a clear and focused scenario. This is because such scenario defines, on one hand, the users' necessities, and on the other hand, the collaborating environment where the platform is going to be used [11]. In addition, all possible interaction methods need to be considered because such scenario indicates flexibility and interoperability for the inclusion of future new devices and sensors.

Here, usability is related to certain access speed and flexibility completion and fulfilment. Moreover, the information flow among the different team members also needs to happen fast and the data must be clear of misinterpretations. Consequently, the suggested tool is utilised successfully if:

- Users obtain the information fast.
- User-system interaction is fast and easy to use. The user makes the least needed number of steps to reach the target.
- Users' interaction can be carried out in different ways associated to the real time necessities.
- Data visualization simplicity as well as good and appropriate visibility is also an important factor.

WORK SPACE

Before defining the architecture, the working environment definition is necessary; this is the place where the surgeon and the team members' work, as well as the special disposition of the elements found in an operations room. At the next figure (Figure 1) we can see the elements disposition and the actors inside the operations room.

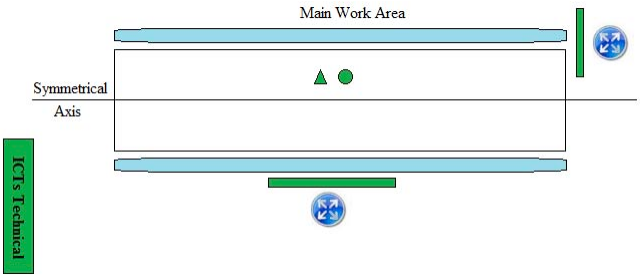

Figure 1. Elements disposition within the operations room

Blue elements indicate the working areas along a symmetrical axis where the medical staff is located. At the central axis, a triangle suggests the audio and video devices used to record the surgery, and by a circle the area to receive and send voice commands to the system is indicated. The two green rectangles correspond with the output devices depicted by images, which can be moved in all directions to facilitate the medical staff positioning. At the operations room, the ICT Technical System can be searched for information in relation to the parameters indicated by the members of the medical team, performing the search and exchanging the information using available interaction patterns. Usually the nurses located near the screens do have access to the searches and can make suggestions to the surgeon about the necessary actions using a touch screen. The surgeon via voice commands can also have access to related information search or any other necessary data.

ARCHITECTURE

A system with such characteristics is proposed to be located on the Cloud so to guarantee data security and acquisition speed. For that reason, the design and development of a system located on the Cloud system guarantees the information access and ensures that previous information is stored before the intervention at the hospital local servers. This is a mixed system where users can access the Cloud for certain operations and consult medical data previously prepared. In addition, they will be able to check the use of the devices located into the operations room.

This architecture depicts the interaction layers so to simplify the interaction process with the different interfaces and hardware devices that can be found.

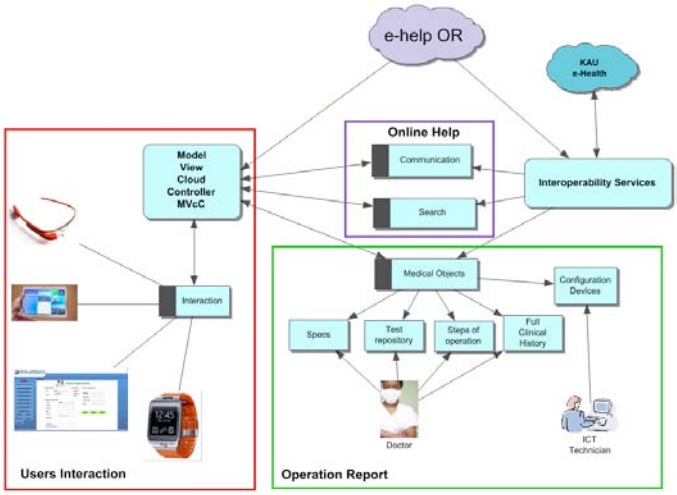

Figure 2. System's architecture

The system is divided into three parts: in red, the part of the system in charge of the interaction with users' devices is indicated; in green, the necessary services are depicted so to create the operation report; and in purple, the online help required during the surgical intervention can be viewed.

- Medical Objects: The grouped medical objects indicate everything that is necessary for the user (the medical team) to access the data, allowing interventions preparation and aiding in their completion. These objects are not found on an Internet server but in the hospital's servers to guarantee quick and secure accessibility. There is also the possibility to access the preparation process of these objects from home; however, access during the surgical intervention is imperative via the local connection inside the hospital.
    - Operation report: In this Cloud part, doctors perform the preparation process and this is related to the necessary documentation. Thus the system can bring forward data from diverse medical platforms under different administrations and hospitals.
    - Specs: The surgeon or the medical team complete the necessary specifications of the intervention within the surgery. These specifications are part of the medical object created to store the technical necessities of the operation, as for example, materials, tests, clinical analysis, staff and specialist used during the intervention.
    - Test Repository: A medical tests bank is created specifically for each patient. These medical tests results are necessary for the intervention surgery and can be acquired before or during the pre-operatory. These tests are documents to specify the types of surgical intervention so to acknowledge any consequences and the ways the surgical intervention can be directed and focused.

- Steps of operation: The surgeon establishes the intervention process and needed steps taking into consideration the previous medical tests results. The doctor can include the post-operatory recuperation process and the required treatment once the intervention has ended, thus the specialist can monitor the patient's recuperation process till s/he has the medical discharge. Therefore, the doctor can create a rich repository of data interventions for a posterior study and also, for sharing with the medical community members.
- Configuration Devices: The ICT technician located in the operations room configures the medical devices in order to adjust them to the team medical necessities, or, in other words, the technician configures the platform depending on the required interaction devices.

- On-line help: This system of online help allows the medical staff to establish searches in real time on the available sanitary platform during the intervention and also establish direct communication with the specialists.

We place an interoperability layer in the system in order to interact with the different systems mediated by Web Services [12].

The interaction process is a MVC (Model View Controller) model modification adapted to the Cloud systems. It facilitates the incorporation process simplification towards the specified devices. Different devices provide diverse ways for interaction, which can be an impediment for system development allowing the inclusion of different interaction types.

The MVC model [13]:

- Model: It refers to the information representation treated by the system; therefore it manages all access to that information, consultation as updates, also implementing the access privileges described at the application's specification (business logic).
- Controller: It responds to events (usually user's actions) and invokes requests to the Model when any of the above is performed over the data.
- View: It presents the Model in an adequate format to interact with the system.

The following figure (Figure 3) depicts the MVC model where the user sends and receives information to and from the system. The red line represents user's interaction area with the computer application. In that area a system is located to allow the user to interact with any preferable interaction method or device. To obtain that result, an abstraction layer is necessary allowing the developers to determine an output. Thus great flexibility is provided to the system without modifying the font code of the data manipulation or the view. A decision was made to provide these model layers because they contribute to the communication with the model.

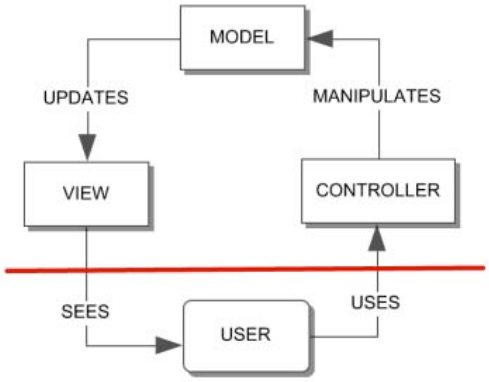

Figure 3. The MVC model

In the suggested VCC (View Cloud Controller) model, interaction and devices are strongly related. Interaction includes voice commands, conventional events or gestural events. However the graphical user interfaces must be developed to allow and support that each part of the system is under continuous development so to support more as well as future diverse interaction models. Therefore, the starting point can be a conventional application so to add tangible functionalities or voice command functionalities. The application inputs and outputs define the types of interaction into the system, so a flexible system needs to be designed and developed, not in regard to programming, but to promote adaptation with the new methods. For that reason a MVC model with a Cloud layer would speed up this type of applications development and would allow the inclusion of new interaction methods.

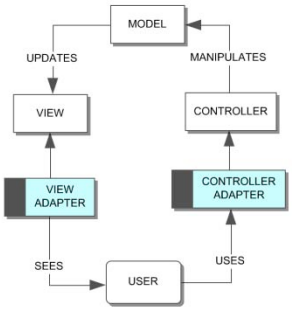

Figure 4. VCC model.

In the previous diagram (Figure 4) the VCC model is represented. Through integrated adapters on the Cloud this model allows to make a migration of input or output of the view or controller model, adapting it to the device's type.

One of the main goals for hosting this layer on the Cloud is that applications, isolated from devices or interaction methods, aid users to share information each other. With this fact the DUIs would provide an important step because usually the DUIs interact with applications of similar characteristics. Therefore, decisions on specific DUIs facilitate working with different type of devices [14].

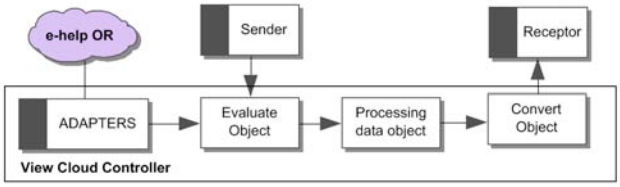

Figure 5. View Model Controller Functionality.

The previous figure 5 shows the ways the View Model Controller works. The adapters evaluate the input object type and the output device to which the information goes, producing a migration from the processed object to the required type of object by the receptor device of information. All this processing is carried out on the Cloud and is located into the presentation logic; the object does not suffer any modification in reference to its content, only on its structure to be interpreted by the receiver. Till now, objects are sent from one screen to another without modification, whereas the emitter and the receiver use the same protocol. Adding this layer, the emitter and receiver's independence is diversified, providing more flexibility to the DUIs in its use and device's independence. One of the goals in the computer software applications development is the operating systems portability. For that reason, it is important to reach portability among different interaction devices so to acquire more functionality for DUIs in the applicable environments.

PROOF OF CONCEPT

An authentic situation occurring in the operations room during the process of a surgical intervention is going to be described and presented. This is a mixed situation using DUI between two different devices and communication between users at the surgery. The users are registered by the surgeon who initiates the intervention and requests information on previously performed medical tests from the assistant and on continuous information delivery from a nurse.

First, the types of devices associated to each user are described as follows:

- The surgeon carries the Google Glass. The doctor requests information from the assistant, which is received, visually on the surgeon's Google glass visor.
- The nurse uses an available tablet to visualize the data and send them to the surgeon.

Figure 6. Nurse's Tablet

In the previous Figure 6, the patient's monitoring results are displayed. This application connects the measurement medical equipment and displays it as with the information on the device. In this case the monitoring works during the implantation of a pacemaker by the cardiologist, who requests from the nurse the medical data with the necessary parameters related to blood pressure and oxygen saturation. The nurse sends that information through the "Drag and Drop" event to the shared area of the screen that the doctor can see this during the intervention surgery. At the following Figure 7 we can observe a display's vision simulation.

Figure 7. Google Glass Visor

At the same time, the anaesthetist sends to his/her clock the Biespecral Index, because s/he has to attend to another operations room at the contiguous room. The nurse sends that information to the clock.

Figure 8. Anaesthetist's clock

CONCLUSIONS AND FUTURE WORK

In this paper, a DUIs medical solution is presented and discussed, providing information exchange between users with different type of devices within an operation room. The abstraction layer created on the Cloud follows the VCC model and adjusts DUIs closer to the specific hardware devices; this is necessary for objects conversion towards a posterior interpretation. This migration is done on the Cloud due to its flexibility, development capacity and processing power. In addition, this platform offers new functionalities as the devices evolve in time.

Here DUIs are directed towards "wereable" mobile devices, because these are usually used for notifications delivery, and also to send useful information. For that reason the proof of concept was successfully implemented deploying "wereable" devices that have the capacity to grow in regard to functionalities. Also new microprocessors can be incorporated increasing the power, as at the same time the WiFi communication will be improving.

REFERENCES

1. Google Glass. http://www.google.com/glass/start/
2. Philips. Soluciones profesionales de dictado y grabadoras de dictado. https://www.dictation.philips.com/es/como-hacemos-la-diferencia/news/el_reconocimiento_de_voz_y_el_flujo_de_trabajo/
3. Philips. Clinical Informatics & Patient Monitoring. http://www.healthcare.philips.com/main/products/hi_pm/products/index.wpd.
4. Fardoun, H. M., & Alghazzawi, D. M., Cipres A. P. (January 2013). Distributed User Interfaces: Usability and Collaboration. Distributed User Interfaces. Human–Computer Interaction Series 2013 (pp 151-163). ISBN 978-1-4471--5498-. Springer London.
5. Paules Ciprés, A., Fardoun, H. M., Alghazzawi, D. M., & Oadah, M. (October 2012). KAU e-health mobile system. In Proceedings of the 13th International Conference on Interacción Persona- Ordenador (p. 29). ACM.
6. Ciprés, A. P., Fardoun, H. M., &Mashat, A. (September2012). Cataloging teaching units: Resources, evaluation and collaboration. Federated Conference In Computer Science and Information Systems (FedCSIS), 2012 (pp. 825--830). IEEE. SCOPUS.
7. Fardoun,H.M.,Antonio P. Ciprés,Sebastian R. Lopez, Bassam Zafar (June 2012). CSchool Interactive Design. 1st international Workshop on Interaction Design in Educational Environments (IDEE 2012). ICEIS 2012 Conference. Proceedings of the 14th International Conference on Enterprise Information Systems ICEIS 2012. 1st International Workshop on Interaction Design in Educational Environments IDEE 2012. INSTICC, ISBN 978-989-8565-17-4, June 28, 2012. Wroclaw, Poland.
8. Galaxy NotePRO (12.2"). http://www.samsung.com/es/consumer/mobile-phone/tablets/pro-series/SM-P9050ZWAPHE
9. GALAXY Gear. http://www.samsung.com/latin/consumer/mobile-devices/galaxy-gear/
10. Kubicki, S., Lepreux, S., &Kolski, C. (2013). Distributed UI on Interactive Tabletops: Issues and Context Model. In Distributed User Interfaces: Usability and Collaboration (pp. 27-38). Springer London.
11. Pedro G. Villanueva,Ricardo Tesoriero,José A. Gallud: Is The Quality In Use Model Valid For DUI?. Proceedings of the 2nd Workshop on Distributed User Interfaces: Collaboration and Usability, DUI 2012 in conjunction with CHI 2012 Conference. ISBN 978-84-695-3318-5, pp 39-44. May 5th, 2012. Austin, Texas, USA.
12. Li, L., &Liu, J. (2012). An efficient and flexible web services-based multidisciplinary design optimisation framework for complex engineering systems. Enterprise Information Systems, 6(3), 345-371.
13. Krasner, G. E., & Pope, S. T. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, *1*(3), 26-49.
14. P. G. Villanueva, R. Tesoriero and J. A. Gallud. Revisiting the Concept of Distributed User Interfaces. In Distributed User Interfaces: Usability and Collaboration. Springer, Human-Coomputer Interaction Series. Eds. M. D. Lozano, J. A. Gallud, R. Tesoriero, and V. M. R. Penichet. ISBN: 978-1-4471-5498-3, pp. 1-15. 2013. url= http://dx.doi.org/10.1007/978-1-4471-5499-0_1

12 + 1 Questions in the Design of Distributed User Interfaces

Victor M. R. Penichet, Maria Dolores Lozano, Jose A. Gallud, Ricardo Tesoriero
Computer Systems Department
University of Castilla-La Mancha (UCLM), Albacete, Spain
{Victor.Penichet, Maria.Lozano, Jose.Gallud, Ricardo.Tesoriero}@uclm.es

ABSTRACT
Current visual display ecosystems raises new situations and new configurations regarding the way a user interacts with a system through the user interface. In a post-WIMP period, we can find coupled displays, multi-touch devices, and interactive table-tops, tablets, tangible user interfaces, eWatchs and many other devices often interconnected through the same applications. This scenario poses researchers new challenges in the design of distributed user interfaces. In this paper we raise a set of questions as guidelines to consider that may drive designers in their work.

Author Keywords
Distributed User Interfaces; DUI; design; Tangible User Interfaces; Multi-Device Environment.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces - *Graphical user interfaces, Input devices and strategies, Interaction styles, Screen design*.

INTRODUCTION
Distributed User Interfaces (DUI) are a novel research field in the Human-Computer Interaction area and plays an important role in the proper design of advanced visual interface display ecosystems when more than one device is used to perform tasks on the same application. This is a common scenario due to the large amount of different devices that users use in their everyday life. Computers have become part and parcel of our daily lives, therefore current applications are also adapted to such situation and provide mechanisms to interact with them from the various devices in multiple ways. New challenges in the design of these applications emerge and they have to be carefully studied by researchers to achieve higher quality applications.

DUI '14, July 01 2014, Toulouse, France
Copyright 2014 ACM 978-1-60558-724-0/14/07…$15.00
http://dx.doi.org/10.1145/2677356.2677664

In this paper, we present some guidelines through 12 + 1 questions taken from an in depth study of papers on DUI mainly presented at the DUI workshops held during the last three years. Researches from all over the world presented their ideas, proposals and innovations in the area. Here we synthetize and identify the most important conclusions, agreements and assumptions which may guide a designer thanks to the knowledge of experts in the field. These guidelines will make the designers think on several aspects to explicitly consider many important points when distributing a user interface. There is one final guideline; a question that designers should consider regardless of the kind of application they design; just a must.

The paper is organized as follows. Firstly, a number of advances concerning display technology and interaction are presented in Section 2. Then, a unified and detailed definition of Distributed User Interfaces is given in Section 3. Section 4 provides the twelve plus one questions we propose as guidelines to design DUIs. Lastly, some conclusions and final remarks are presented in Section 5.

ADVANCES IN DISPLAY TECHNOLOGY AND INTERACTION: CURRENT TECHNOLOGICAL ECOSYSTEM
Current technological ecosystem introduces important challenges to developers and researches on display technology as well as on interaction matters. The way a user interacts with the system has evolved in such a manner that new research opportunities arise. In the last decades, we have moved from single end user interfaces to a wide variety of interactions due to the emergence of different devices platforms, architectures, operating systems, space and/or time distribution, etc. We have moved from one user interacting with their own computer in a really *simple* way, to more complex situations. Moreover, even such complex situations have become further more complicated with the introduction of the wide variety of different devices, platforms, architectures, operating systems, space and/or time distribution, etc. *where users may interact despite the complexity of these settings.* User interface design and interaction have turned into important research fields plenty of topics to tackle.

There are many interaction techniques such as (a) *touching* which involves touching an object. Some projects using this technique can be found in [4, 11, 20]; (b) *scanning* through

the mobile device which is capable of scanning information and interacting with the system to provide a service to the user [21]; (c) approach&remove [16] which allows the user to control distributed user interfaces by approaching the mobile device to an object; and lastly, (d)*movement based interaction* through motion sensing input-output devices like Kinect and Leap Motion.

New scenarios have appeared such as *Multi-Device Environments* (*MDE*) with multiple and heterogeneous devices distributed in the environment along with screens and surfaces where user interfaces are displayed.

We also should mention an increasingly fashionable paradigm, *ubiquitous computing*, described by Mark Weiser [22] in 1991: "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it". Its main goal is to provide the user with advanced and implicit computing, capable to carry out a set of services but without being aware of it. In order to interact in these environments, user interfaces are required to allow an intuitive and simple interaction that help remove the barriers so far encountered, such as prior learning of the use of systems.

Another interesting vision lies in doing common objects interactive and perceptible. *Tangible User Interfaces* (*TUI*) refers to user interfaces which give physical form to digital information, making the parts directly malleable and perceptible [12].

Current technology offers us a wide variety of possibilities to make interaction richer. *RFID* (*Radio Frequency Identification*) is a system for storing and remotely retrieving data which allows the identification of an object from the distance with no contact. In the same way, *NFC* (*Near Field Communication*) standardizes the way in which smartphones and other mobile devices establish radio communication with each other by touching them together or bringing them into proximity. Regarding Web communication we could mention Web Services as a set of protocols and standards to exchange data between applications providing interoperability; however, Websocket goes a step further and provides full-duplex communication channels over a single TCP connection. It can be used by any client or server application providing real-time interaction through the Web.

HTML5, Android OS, Windows Phone, iOS as well as many other technologies, and research fields around them provide amazing challenges and design opportunities. User interface distribution is becoming a really interesting research topic because of the nature of such advances. Even more, we could talk about real necessity more than mere opportunities.

DISTRIBUTED USER INTERFACES

GUI (Graphical User Interfaces) formerly thought to be used for PC-based software applications and controlled by a mouse and a keyboard are no longer enough in the new scenarios where we have multiple devices and displays. In these cases, other kinds of interfaces are necessary. DUIs (Distributed User Interfaces) have been conscious or unconsciously used and defined in many different ways in the near past.

As Niklas Elmqvist [7], Jean Vanderdonckt [19] and other researchers state, the specialized literature is plenty of references highly related to distributing the user interface, as in [9, 2], among others . We can find *Migratory and Migratable Interfaces* [3] to describe applications capable of roaming freely on the network instead of being confined to a particular computer. *Plasticity* [18] is a concept defined as the capacity of a UI to withstand variations in both the device and its physical environment while preserving usability. The aforementioned *Multi-Device Environment* (*MDE*) consists of multiple, heterogeneous and distributed devices, displays and surfaces. Also *Ubiquitous Computing* was previously introduced; it integrates data and computation into everyday objects and activities. Other connected terms are *Multi-Device Interaction Techniques*, *Application and Content Redirection*. Lastly, Niklas highlighted CAMELEON-RT as a middleware software infrastructure for distributed, migratable, and plastic interfaces 1.

Some approaches to a definition of DUI may be found in recent past years [6, 15 19]. However, after several workshops on Distributed User Interfaces where researchers have deeply discussed about this novel research field, we have found the definition given by Niklas [7] as the most appropriate, also assumed by many other authors subsequently. DUI was then described as "a user interface whose components are distributed across one or more of the dimensions *input* [so called input redirection], *output* [so called display or content redirection], *platform* [i.e., architectures, operating systems, networks, etc.], *space* [i.e., co-located or remote interactive spaces], and *time* [synchronous or asynchronous]". The wording in square brackets corresponds to some comments he made after the definition.

In order to get an in-depth understanding of the need of distributing user interfaces, Donatien Groulaux, Jean Vardendonckt and Peter Van Roy illustrated us with a really appropriated metaphor: a painter painting a scene [10]. In such scenario, the painting is the main focus of attention, while the rest of tools remain secondary. If we consider a software tool for painting, the colour palette, the pencil, the painting tools, etc. are allocated on the screen in different positions. Although they are well-grouped, the user interface collapse with so many information altogether, and it is not considered natural [13]. It is true that many possible configurations are available so that the user can

modify the layout of toolbars when needed, but still no natural and uncomfortable. We could design a more natural interface making use of different displays. Each display allocated in a similar way as it would be allocated in a real scenario. The main objective would be the same: painting. There would be still only one application. But several user interfaces in different devices could make it more natural. This is just a small example to show the power of distributing the user interface. Applying this minor example into our daily reality is much more complicated. That is the reason why researching on DUIs is more and more trendy. If we put together the complexity of nowadays applications and the aforementioned dimensions of DUIs (input, output, platform, space and time), it is not only a matter of quality, but a matter of necessity. We also depicted the distribution of user interfaces according to the users' mental models 8, splitting the interface of collaborative games on a projector to be displayed more clearly. Mobile device interface were used as interaction devices between the main interface and several tangible user interfaces.

Up to this point, the concept of DUI has been adequately explained. However, if we would like to go further on, we also could consider how to dynamically deal with such user interfaces in the developed applications. To do so, Grolaux [10] defines a set of properties as the basis of what they call a detachable user interface: "*detachability* [any UI component of the interactive application of interest can be detached from its host UI], *migratability* [the detached UI component is migrated from the source computing platform […] to another target platform], *plastifiability* [5] [the migrated UI component is adapted according to the new constraints posed by the new target computing platform], *attachability* [the plastified UI component is attached to any UI running on the target computing platform, if needed]." The wording in square brackets corresponds to some comments he made.

Although it is practically not considered [7] in the definition of DUI, *collaboration* is still a highly important concept to take into account. It was not included since users were not considered as a distribution dimension; nevertheless, space/time dimensions of CSCW are specifically mentioned in the definition. Besides, due to the nature of nowadays applications, technology and the use of devices by users, we also consider collaboration as a key issue to keep in mind. Empirical studies [14] indicate that the distribution of shared and private workspaces to support balanced participation in face-to-face collaboration is very important in collaborative settings.

GUIDELINES TO DESIGN DUIS

As stated above and according to the visions presented in previous editions of the workshop on DUIs and our own expertise in this area, we may summarize a number of key points to be considered as important subjects to cope with in the development of DUIs. It is important to notice that the main perspective we consider regards *Human-Computer Interaction* research field. As can be imagined, the distribution of the user interface implies many other fields which are also well appreciated. *Interaction techniques* such as *touching*, *scanning*, *approach&remove* and *movement based interaction* through motion sensing input-output devices, among others, determine the success of these systems. Traditional interaction techniques could be also good ones; however, analyzing the most adequate interaction in such a specific system becomes a key factor. Q1: What are the most appropriate interaction techniques?

Tangible User Interfaces (*TUI*) is also closely linked to interaction as implies a new way of interacting with software applications. Everyday objects take part in the distribution of user interfaces and interactions, which makes the development of systems more difficult, but makes the interaction and participation easier for many users. TUIs and the big amount of new devices that invades our everyday life drive us to consider the principles of *Multi-Device Environments* (*MDE*) as well as the idea of *Ubiquitous computing*: anytime, anyplace, anywhere, if possible. Q2: What are the devices working in the whole system? Q3: How do they communicate with each other? Q4: Where are they allocated?

In accordance with the assumed definition, we should consider the mentioned dimensions: "a user interface whose components are distributed across one or more of the dimensions *input*, *output*, *platform*, *space*, and *time*". Some of these basics are partially included in the previous questions; however, we may still wonder about substantial points. Q5: What parts of the user interfaces should we distribute? Q6: What are the main features of the different platforms, mainly according to compatibility, interoperability, etc.? Q7: What tasks are thought to be performed synchronously and what asynchronously? Q8: Are all de UIs thought to be co-located in the near space or in different spaces?

For a more in-depth consideration, Grolaux et al. [10] defined a set of properties as the basis of what they called a "detachable user interface": *detachability*, *migratability*, *plastifiability*, *attachability*. If considered, we also may pose another question. Q9: What components of a UI may we compose and decompose?

Sangiorgi et al. describe a set of challenges for distributing the user interface [17]. They finish the discussion with an interesting message: "The list of challenges presented on this paper is intended to bring the discussion of "old" problems of collaborative systems to the contemporary context". That is what, beyond the collaboration concept, we propose with this paper. Anyhow, once more we should realize about the importance of considering what users do in the whole system: *collaboration* as well as *awareness*. The four challenges they introduce for a distributed sketching system take into account user *awareness* as a first issue: "make users aware of each other's activities […]. Furthermore, by having such a large number of possible

devices and an "infinite" workspace to work on, it is hard to keep track of which devices are observing specific parts of the wall." Then the problem of *pointing remotely*: "how to "point" at something remotely?" Other important factors are *concurrency* and *conflicts*. Lastly, what they introduce as a novel subject is considering "*the right tool for the job*: Not all the devices have the same resolution or performance […] which devices are suitable for the […] activities". Q10: In what way users collaborate, coordinate and communicate? Q11: How can the user be aware of what the other users of the system do? Q12: Where may we find concurrency problems and other conflicts? The problem of "pointing remotely" and "the right tool for the job" are somehow included in previous questions, mainly regarding interaction techniques.

Lastly, keeping always in mind that technology evolves quickly, the state of *current technology* should also be considered as a fundamental issue. Technology such as *RFID* (*Radio Frequency Identification*), *NFC* (*Near Field Communication*), HTML5, Android OS, Windows Phone, iOS enriches interaction and provides new ways in the distribution of the user interface. Designers and researchers need to keep abreast of new developments in the field. The last question is Q13: Are we using technology properly or is there any other that could solve the problem more accurately? This last question is a really generic one, which could be used in the design of any kind of system; however, how not mention it?

CONCLUSIONS

In this paper we have presented twelve questions as guidelines that designers might consider when addressing the design of applications based on Distributed User Interfaces. These questions pose a set of problems or situations that designers should take into account in these environments. In this way, key aspects regarding would be explicitly addressed. Another last guideline just highlights the importance of using the appropriate and cutting-edge technology. These 12 + 1 questions are a synthesis which comes from the expertise of a number of researchers who are experts on DUIs. These guidelines do not intend to be a must, but just another piece of the puzzle that may be helpful in this research field.

ACKNOWLEDGMENTS

This work has been partially funded by project TIN2011-27767-C02-01 from the Spanish Ministry of the Economy and Competitiveness and by project TSI-100101-2013-147 from the Spanish Ministry of Industry, energy and Tourism. Special thanks to everyone who participated and/or made possible past DUI workshops.

REFERENCES

1. Balme, L.; Demeure, A.; Barralon, N.; Coutaz, J.; Calvary, G.: CAMELEON-RT: A software architecture reference model for distributed, migratable, and plastic user interfaces. In Proceedings of the Symposium on Ambient Intelligence, volume 3295 of Lecture Notes in Computer Science, 291–302. Springer, 2004.
2. Bandelloni, R.; Paterno, F.: Flexible interface migration. In Proceedings of the ACM Conference on Intelligent User Interfaces, 148–155, 2004.
3. Bharat, K. A. and Cardelli, L. Migratory applications. In Proceedings of the ACM Symposium on User Interface Software and Technology, 133–142, 1995.
4. Broll ,G. Graebsch ,R., Holleis ,P. , Wagner,M. Touch to play: mobile gaming with dynamic, NFC-based physical user interfaces, Proceedings of the 12th international conference on Human computer interaction with mobile devices and services, September 07-10, 2010, Lisbon, Portugal
5. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target UI. Interacting with Computers 15,3
6. Demeure, J. Sottet,S. Calvary,G. Coutaz,J. Ganneau,V. and Vanderdonckt,J. The 4C reference model for distributed user interfaces. In Proceedings of the International Conference on Autonomic and Autonomous Systems, 61–69, 2008
7. Elmqvist, N. (2011). Distributed User Interfaces: State of the Art. Workshop on Distributed User Interfaces 2011 (DUI) at the 29th ACM CHI Conference on Human Factors in Computing Systems 2011, ISBN: 978-84-693-9829-6, Vancouver, Canada.
8. Guía,E. Lozano,M.D. Penichet.V.M.R. Interaction and Collaboration Supported by Distributed User Interfaces: FromGUIs to DUIs. In Proceedings of of the 13th International Conference on Interacción Persona-Ordenador. ACM, Article No. 53. ISBN: 978-1-4503-1314-8 doi>10.1145/2379636.2379688Elche, Alicante, Spain, Oct. 3-5, 2012
9. Grolaux, D.; Roy, P. V.; Vanderdonckt, J.: Migratable user interfaces: Beyond migratory interfaces. In Proceedings of the IEEE/ACM Conference on Mobile and Ubiquitous Systems, 422–430, 2004.
10. Grolaux, D., Vanderdonckt, J., Van Roy, P. Attach me, Detach me, Assemble me like You Work. 10th IFIP TC 13 Int. Conf on Human-Computer Interaction INTERACT'2005 (Rome, 12--16 September 2005). M.-F. Costabile, F. Paternò (Eds.). Lecture Notes in Computer Science, Vol. 3585, Springer-Verlag, Berlin, 2005, pp. 198--212.
11. Hardy,R., Rukzio,E. Touch & interact: touch-based interaction of mobile phones with displays, Proceedings of the 10th international conference on Human computer interaction with mobile devices and services, September 02-05, 2008, Amsterdam, The Netherlands

12. Ishii, H.; Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 234–241, New York, NY, USA. ACM Press
13. Jacobson, J.: Configuring Multiscreen Displays with Existing Computer Equipment. In: Proc. of Conf. on Human Factors HFES'2002.
14. Looi, C.K.; Lin, C-P.; Liu, K-P. (2008). Group Scribbles to Support Knowledge Building in Jigsaw Method. IEEE Transactions on Learning Technologies, July-September 2008 (vol. 1 no. 3), pp. 157-164
15. Melchior,J. Grolaux,D. Vanderdonckt,J. and Roy,P. V. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In Proceedings of the ACM Symposium on Engineering Interactive Computing System, 69–78, 2009
16. Romero,S., Tesoriero,R., González,P. , Gallud,J. A., Penichet, V. M. R.: Sistema Interactivo para la Gestión de Documentos Georeferenciados basado en NFC. Interacción 2009, X Congreso Internacional de Interacción Persona-Ordenador. Barcelona. Septiembre 2009. ISBN-13:978-84-692-5005-1
17. Sangiorgi, U. B., Zen, M., Motti, V. G. & Vanderdonckt, J. (2013). Challenges on Distributing a Collaborative Sketching System Across Multiple Devices. In M. D. Lozano, A. S. Mashat, H. M. Fardoun, J. A. Gallud, V. M. R. Penichet, R. Tesoriero & J. Vanderdonckt (eds.), DUI@EICS (p./pp. 50-53). ISBN: 978-84-616-4792-7
18. Thevenin, D.; Coutaz, J.: Plasticity of user interfaces: Framework and research agenda. In Proceedings of IFIP INTERACT, 110–117, 1999.
19. Vanderdonckt, J. Distributed user interfaces: How to distribute user interface elements across users, platforms, and environments. In Proceedings of the International Conference on Interaccion, 2010
20. Vandervelpen, Ch., Vanderhulst, K., and Coninx, K. Light-weight Distributed Web Interfaces: Preparing the Web for Heterogeneous Environments. Proc. of 5th Int. Conf. on Web Engineering ICWE'2005 (Sydney, July 25--29, 2005). Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2005.
21. Want, R., Fishkin, K.P., Gujar, A., Harrison, B.L.: Bridging physical and virtual worlds with electronic tags. In: Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, ACM Press, Pittsburgh, Pennsylvania, United States, 1999.
22. Weiser, M. The computer of the 21 century. Scientific American Special Issue on Communications, Computer, and Networks, 1991

Performance Evaluation of Proxywork

Pedro G. Villanueva
University of Castilla-La Mancha
Campus Universitario s/n
(02071) Albacete, Spain
pedro.gonzalez@uclm.es

Ricardo Tesoriero
University of Castilla-La Mancha
Campus Universitario s/n
(02071) Albacete, Spain
ricardo.tesoriero@uclm.es

Jose A. Gallud
University of Castilla-La Mancha
Campus Universitario s/n
(02071) Albacete, Spain
jose.gallud@uclm.es

ABSTRACT
Proxywork is a system that allows users to distribute user interface components of any Web application among a set of devices. In order words, it allows to transform any Web application in a Web application with Distributable User Interface. The distribution is controlled by the user through a set of primitives (clone, copy and migrate) attached to Web page components. Proxywork injects these operations automatically into the Web page components in runtime, so Web pages do not require any extra information in order to be distributed among different devices. This paper presents an evaluation of Proxywork productivity to perform defined tasks. This evaluation demonstrates that, the productivity is greater when we use Proxywork instead of without Proxywork when performing certain tasks.

Author Keywords
Distributable User Interfaces; Web; Proxywork; Quality in Use; HCI

ACM Classification Keywords
H.5.2 [Information interfaces & presentation]: User Interfaces; H.5.3 [Information interfaces & presentation]: Group and Organization Interfaces.

INTRODUCTION
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 "distribute" 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.

The term Distributed User Interface or DUI has been defined in many different ways, some of them [3, 4, 6]. But as it is mentioned in [1], it is not possible to find a single formal definition that can be considered as the reference. This concept is redefined in [7] where the new concept of Distributable User Interface (DeUI) is presented.

Proxywork system offers the ability to transform Web applications designed to run on a single display into Web Applications running on a DeUI. This transformation is performed at runtime by 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. This modification attaches a menu that allow users to clone, copy or migrate UI components to the rest of the browsers that are connected to the Proxywork proxy.

In this work, we present a user study whereby we want to demonstrate whether productivity when certain tasks are proposed, is greater using Proxywork than without Proxywork.

This article is organized as follows: Section 2 introduces the concept of DeUIs. Section 3 briefly describes the Proxywork system. Later on, section 4 shows the quantitative evaluation of the Proxywork system. Finally, conclusions and future work are presented.

THE DISTRIBUTABLE USER INTERFACE CONCEPT
A formal definition of the concept of distribution of the user interface is present in [7]. That article proposes a new concept called Distributable User Interface (DeUI).

The DeUI concept is defined in an informal way as follows: A User Interface is distributable, if and only if, there is at least one *interaction object* which can be in more than one *platform*.

The Platform entity is the combination of Hardware (CPU and I/O devices) and the operating system, which supports the running application. In the case of Web applications, the browser is part of the platform. Some examples of different platforms are smartphones, tablets, laptops, desktop computer, etc.

The Interaction Object entity (hereafter referred to as IO) is defined as the Abstract Interaction Object (AIO) described in [5]. Some examples are: an application window, a button,

DUI '14, July 01 2014, Toulouse, France
Copyright 2014 ACM 978-1-60558-724-0/14/07..$15.00
http://dx.doi.org/10.1145/2677356.2677665

a layout, a textbox, etc. There are three types of interaction object: Interaction Component, Interaction Container and Interaction Surface.

The Interaction Component entity represents the basic elements of the user interface. An InteractionComponent cannot contain other elements. Some examples are: a button, a label, an icon of a RFID panel, etc.

The Interaction Container entity represents user interface elements able to contain other elements. Some examples are: a grid including two buttons, a region of a RFID panel that contains some RFID icons, etc.

The Interaction Surface entity represents user interface elements that may contain other elements, but that cannot be contained in another element. Some examples are: a windows desktop, a panel of RFID, an Activity on *Android*, a Page on *Windows Phone*, a View on *iPhone*, etc.

DISTRIBUTING WEB APPLICATIONS: PROXYWORK

Proxywork is a system that supports Web-based DeUIs. This system is a proxy that transforms any Web Application into a Web Application with a distributable user interface. This system is the first that allow users to interact on true DeUI environment. Figure 1 shows the overview of the Proxywork architecture that is explained in the next paragraphs.

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, e. g. http://www.yyy.com, departs from the device browser, and arrives to the Proxywork proxy. The Proxywork system requests the Web resource to the Web server where the application is hosted. 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 will be able to perform the distribution of UI components. Proxywork returns the modified Web page supporting the distribution primitives to the device Web browser that sent the request. Finally, the Web browser shows the distributable Web page (http://www.yyy.com).

Distribution Granularity

The distribution 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.

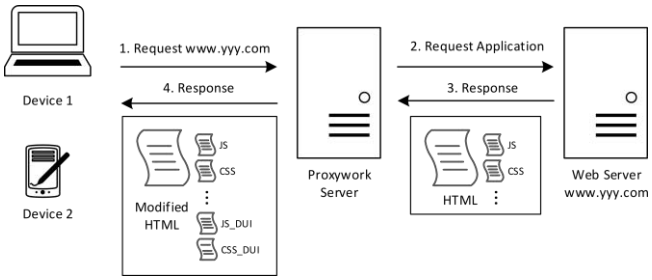

Figure 1. Overview of the Proxywork architecture.

Distribution primitives

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

The **Connect** primitive 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 primitives that require a target device Web browser (Clone, Copy and Migrate).

The **Disconnect** primitive 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 primitives.

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

The **Copy** primitive allows users to copy UI components from one device Web browser to another one connected to the same distribution environment. This primitive requires the target device Web browser as parameter. Besides, the interaction with this component in the source or target device, affects to the source device.

The **Clone** primitive allows users to copy UI components from one device Web browser to another one connected to the same distribution environment. This primitive requires the target device Web browser as parameter. Besides, the interaction with this component in the source device, affects to the source device, and the interaction with this component in the target device affects to the target device.

The **Migrate** primitive sends UI components from one device Web browser to another one connected to the same distribution environment. This primitive requires the target device Web browser as parameter. The component disappears from the source device and appears on the target device. Besides, the interaction with this component in the target device, affects to the source device.

To illustrate the difference between Copy and Migrate let us suppose that A and B are two device Web browsers. If a user performs a Copy primitive on a UI component, called X, running on A being the target device B, all subsequent primitives performed on X do not affect B in any way. However, if a user performs a Migrate primitive on a UI

component, called Y, running on A being the target device B, the Y UI component disappears from A and appears on B. Note that all primitives performed on Y are targeted to A.

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

When users click the right button of the mouse on any UI component, a distribution menu with the Clone, Copy and Migrate primitives appears. For each primitive there is a list of devices that are connected to Proxywork at that time. The Figure 2 illustrates this with an example.

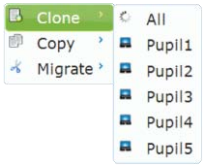

Figure 2. Distribution menu with Clone, Copy and Migrate primitives and five connected devices.

USABILITY EVALUATION

Given the Proxywork system, a quantitative evaluation was carried out. Our goal was to investigate whether the productivity when performing certain tasks is greater using Proxywork than without Proxywork. Therefore, we have focused mainly on the total time of the distributed task and the total time of the task without distribution.

The scenario simulates a primary school classroom and our participants are professors. The participants were asked to complete the next task: the user must to show, in the PC of five students (pupils), the first video of *Youtube* that appears when the user searches the string "children's stories" in the search engine of *Youtube*.

For this experiment we have set a null hypothesis, which allows us to evaluate the productivity of Proxywork for the proposed task. The null hypothesis is described as follows:

H_0: There is no significant improvement in the total time of the distributed task with Proxywork respect to the total time of the task without using Proxywork.

Participants

Five volunteer participants (2 female) were recruited from the local university campus. Participants ranged from 29 to 55 years (mean = 37.4, SD = 10.33). All were daily users of computers, reporting 4 to 12 hours usage per day (mean = 7.4, SD = 3.13). All were weekly users of *Youtube* Web site, reporting 5 to 20 videos per week (mean = 11.2, SD = 5.97). Participants had no prior experience with Proxywork system.

Apparatus

The hardware consisted of six *Windows* hosts, one of them managed by participants (2.8 GHz *Intel Core i7* with 8 GB of RAM) (see Figure 3.a) and the rest of them busy by pupils (2.4 GHz *Intel Core i5* with 6 GB of RAM) (see Figure 3.b). All hosts ran *Windows 7* and used *Google Chrome* browser. Hosts were connected to local university Ethernet.

Procedure

The experiment was performed in a quiet room. The user had their own computer and other 5 computers available for pupils. All computers showed initially a different page of *Youtube*. Each user should perform first the proposed task without using Proxywork, and subsequently, had to do the same task from his/her computer using Proxywork (within subjects). In the first case, the user should work in each computer searching and starting the video. In the case of Proxywork, the user made use of the clone primitive to all computers from her/his own computer (see Figure 2).

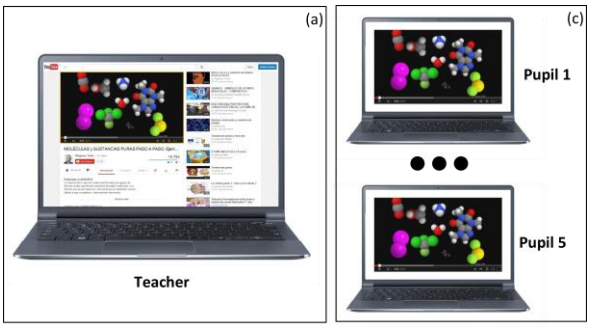

Figure 3. Hardware for experimentation.

Each session of the experiment took half an hour, and during that time, the user performed the task with both conditions. The first step at the beginning of the session was to check all systems worked correctly: the computers, the Proxywork system and the measuring device. Later, we explained the experiment and the task to be carried out to each participant. Once the user had a clear idea about the experiment, we gave him/her a training session, and after that, the user performed the task with both conditions. During the testing we measured the times, and recorded the process.

Once all the sessions were made, we collected the task times without Proxywork and the task times with Proxywork. Errors with each condition were also collected.

Design

Independent and dependent variables can be obtained from the null hypothesis. The independent variable in this case is the technique used, with two possible conditions (with Proxywork or without Proxywork). The dependent variable is the task time. Thus, by having a single independent variable, the experiment has a basic design instead of a factorial design (more than one independent variable).

We decided to carry out the experiment with a "within group" design instead of a "between groups" design because the number of users selected to perform the experiment is reduced. Besides, the learning effect can be considered null. The advantage of this design is that you need fewer users, but the limitation is the impact due to the fatigue of the users.

RESULTS AND DISCUSSION

A dependent variable was measured through the course of the experiment. Results for the basic metric of the task time is presented first. This is followed by additional investigations on a linear regression of the task time respect of the number of computers.

Task time

In user studies involving multiple conditions, the ultimate objective is to find out whether there is any difference between the conditions. Since the data points contributed by the same participant are related, a paired-sample *t* test should be used [2].

The results for the task time without Proxywork are (mean = 81.8, SD = 11.61) and for the task time with Proxywork are (mean = 30, SD = 2.34).

After applying the paired-sample *t* test on the set of data collected in the experiment, we obtained a statistical test (*t*) 10.91 with the specific degrees of freedom (df = 5), which yields a *p-value* of 0.00000562, with which there is strong evidence, with a 95% confidence, that the task time without Proxywork is significantly greater than the task time to the using Proxywork, (t (5) = 10.91, $p < 0.05$).

You can also be 90% confident that the true difference between the means is between 41,679 and 61,921, according to the uncertainty associated.

Linear regression

We measured the time users spent to show the video on different number of computer used (1 to 5 computers), and a linear regression has been applied to make a prediction of how long it would take to show the video on N computers. The result is the linear equation f (x) = 1.200 + 16.04 x. This equation allows us to predict the time that it would take to accomplish the task without Proxywork considering x is number of computers. Number of computers does not affect the task time performed with Proxywork (f (x) = 30). The Figure 4 shows the average task time for both conditions of the independent variable according to the number of computers.

CONCLUSION

An paired-sample *t* test suggests that there is significant difference in the total time of the task distributed with Proxywork respect to the total time of the task without using Proxywork (t(5) = 10.91, p<0.05).

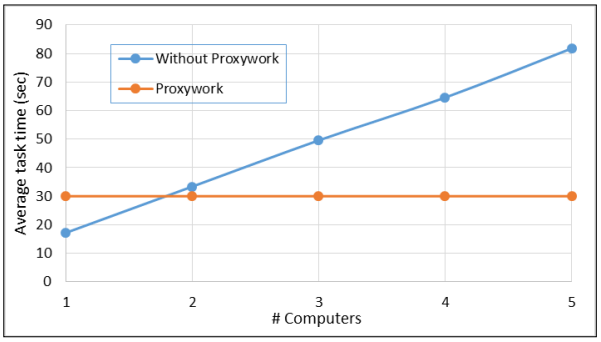

Figure 4. Average task time for both conditions of the independent variable according to the number of computers

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).

REFERENCES

1. Elmqvist, N. Distributed User Interfaces: State of The Art. *CHI 2011 Workshop* (2011), 7-12.
2. Lazar, J., Heidi J., and Hochheiser, H. Research Methods in Human-Computer Interaction. Wiley Publishing (2010). ISBN: 978-0-470-72337-1.
3. López, J.J., Gallud, J.A., Lazcorreta, E., Peñalver, A., and Botella, F. Distributed User Interfaces: Specification of Essential Properties. *Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem*. Springer (2011). ISBN 978-1-4471-2270-8. Chapter 2, 13-21.
4. Melchior, J., Grolaux, D., Vanderdonckt, J., and Van, R. P. 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*. ACM Press (2009), 69-78.
5. Vanderdonckt, J., and Bodart, F. Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. *Proc. of InterCHI'93 "Bridges Between Worlds"*. ACM Press (1993), 424–429.
6. Vandervelpen, Ch., Vanderhulst, K., Luyten, K., and Coninx, K. Light-weight Dis-tributed Web Interfaces: Preparing the Web for Heterogeneous Environments. *Proceeding of 5th Int. Conf. on Web Engineering. ICWE'2005*. Springer-Verlag (2005).
7. Villanueva, P. G. Revisiting the Concept of Distributed User Interfaces. *Distributed User Interfaces: Usability and Collaboration*. HCI Series (2013). ISBN 978-1-4471-5498-3. Chapter 1. 1-15.

Real Time Public Transport Location and Time Services for mobile users

Habib M. Fardoun, Daniyal M. Alghazzawi, Lorenzo Carretero Gonzales
King Abdulaziz University
Jeddah, Saudi Arabia
{hfardoun, dghazzawi, lgonzales}@kau.edu.sa

ABSTRACT
At the people's daily routine is usually included the option of using the public transport to reach their destination. However, in most cases, they have to arrive at a specific time to the expected place, calculating the necessary time of the whole trip. In this proposal we are going to introduce a platform, which unifies the services that the public transport companies have, to provide to the user, by mean of a mobile device, an accurate notion of the time to the destination and of the transport's location every moment.

Author Keywords
Distributed User Interfaces; Public Transport; Mobile Devices; Web Services.

ACM Classification Keywords
H.2.4. Systems. *Distributed Databases*.

H.5.0. Information Interfaces and Presentation. *General*.

INTRODUCTION
At current society, the use of the public transport is, in most cases, a problem due to the lack of parking places in the main cities. In addition, this fact provokes certain worry on people because they don't know how much time they need to park or if they will find a parking for their vehicle. This brings the possibility of arriving late to the working place or to an important meeting [1]. For that reason, users usually going out early from their houses to avoid problems like the previous ones, which brings a loss of personal time for sleeping or for being with the family. [2]

The administrations take into account these factors and for that reason they offer to the citizens' public transports for carry them to near places to their destination, without expending time worrying about where they have to park or the time to make it. Thus, transports like the metro, the bus, the tram or the train, which pass each a specific time for determined stops, available for users. Moreover, the price of this kind of transports is usually cheap to provide a wide access to everyone. However, although all of this facilitates the citizens' life, they have to continue estimating the time to arrive to a specific destination. Therefore, they continue using more time than the necessary with the target of not to arrive late to the meetings or work.

Due to previous facts we propose a solution that helps to the users to know, with a small margin of error, the time to expend from one stop to another one in real time, with the possibility of seeing in a map where the transport is in a determined moment, and the necessary time on each section of the trip. Thus, after consulting the available application at the mobile device, the user will be able to estimate meetings with a bigger accurate, avoiding the fact of getting out early from his/her home and promoting the punctuality at the meetings, by means of Distributed User Interfaces [3].

At next sections we are going to talk about the services available by the public transports, about the system's architecture, the methodology used to carry the platform out and how the different users interact with the system to obtain the requested information.

STATE OF ART
The companies of public services usually provide through their web page information about all related with the transport: stops, schedule, times, etc. Thus, the user, after navigating through the web, introduces the street origin and destination, and the system provides the route and the type of transport that can be used beside with estimated times.

http://dx.doi.org/10.1145/2677356.2677666

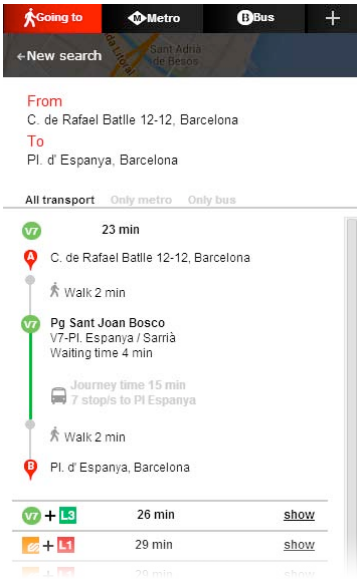

Figure 1. TMB (Metropolitan Transport of Barcelona)

In addition to the previous comments, the company Google[1] with its product Maps[2] [4], also provides transport's routes, showing different trips that can be used, by mean of a personal vehicle or by public transport, beside with information about the estimated times to reach the destination.

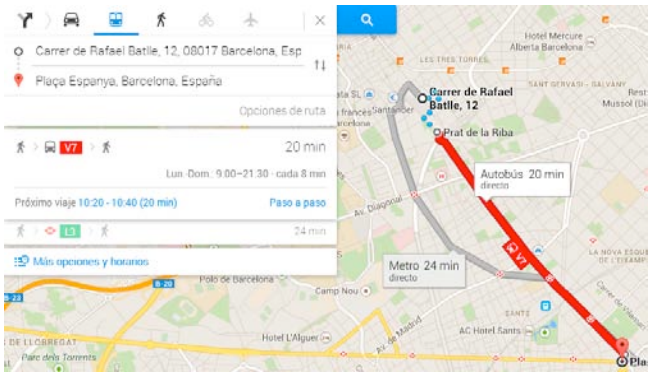

Figure 2. Google Maps.

Once we are physically located at the different stops, we are able to obtain more accurate information about the time to arrive our next transport thanks to the digital devices that show the estimated time of arrival. However, we must to be physically there to know this information, for that reason it would be very useful to consult it before going out.

ARCHITECTURE

This system starts from the different user interfaces distributed among the operators' terminals of each of the implicated public transport companies. With the information about the times and location that they have available, we open read-only outputs through web services for a posterior treatment at the centralized platform. This platform is in charge of unifying all the obtained information and providing to the users of the necessary data to control the transport times. Below we show the architecture taking as a reference the public transport of the city of Barcelona.

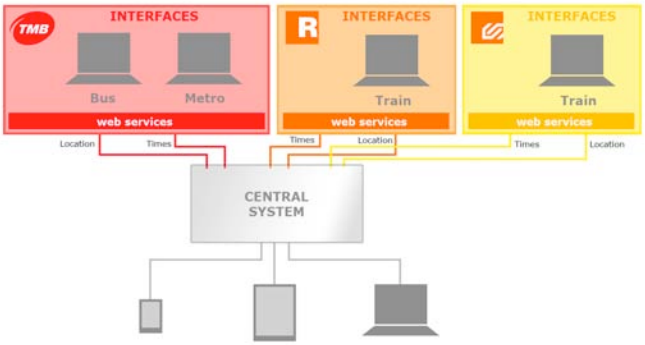

Figure 3. System's Architecture.

As we can see at the Figure 3, we have available various distributed user interfaces among the different public transport companies.

- In red we have the TMB (Transporte Metropolitano de Barcelona), in English: Metropolitan Transport of Barcelona, related with the transport by bus or metro [5].
- In orange are defined the "Rodalies de Catalunya" that makes reference to the regional train, which connects with the farthest parts of the city and with surrounding towns [6].
- In yellow we have the FGC, the railways of Cataluña, which go from the centre to the outskirts of the city [7].

Taking into consideration that these companies have all the information related with their transports, for example the location, they can estimate the time to arrive the next transport to a specific stop. If we can consult that data through read-only web services, we will have the necessary information for users can consult their itinerary in a more accurate way. To avoid a dispersion of all that information

[1] Google. North American multinational enterprise specialized in products and services related with Internet, software, electronic devices and other technologies.

[2] Maps. Google's service that allows to visualize the world through satellite images, maps or a combination of both.

into different web pages and/or mobile applications, we provide a central kernel where all the data is stored. Therefore, we centralize the data providing to the users the information of the different transport means of the city.

To provide a wider accessibility, we made a mobile application to access and visualize the data. Thus, with only a mobile device we have all the information related with the location of the next transport and with the times for arriving to the next destination.

METHODOLOGY

Basing on the services offered by the public transport companies to the users and on the information provided by Google Maps, we are going to obtain a system, which unifies and improves the requested information by those users. Below we show the services that each of the companies provides:

- TMB.
  - The Metropolitan Transports of Barcelona are related with the bus and metro, offering routing services and estimated times in function of a street origin and a street destination.
- Rodalies.
  - The "Rodalies de Catalunya" related with the regional train, offer routing services in function of a stop origin and a stop destination, and they have an exact schedule about departures.
- FGC.
  - "Ferrocarrils de la Generalitat de Catalunya" are train services similar to metro and show routes from a stop origin and a stop destination with determined departure times.
- Google Maps.
  - Offers detailed services for routing by walk, with public transport or private transport.

In all of them, there are digital posters on each station or stop where the arrival time of the next transport is indicated, although not in every bus stop this service is offered.

Having available all this information, our proposal treats to improve and to unify all the information that is distributed on different control interfaces to be more accessible for the user and to be more useful. Thus, we will use the info of Google Maps [8][9] to know the time that we need to arrive to the stop by walk (added in future versions). In addition the TMB, FGC and Rodalies provide the data to know about how much time we have to wait till the transport's arrival. Besides, the application shows the transport's location with the help of the GPS technology and the used one when the train goes under the floor [9]. In addition, for the transfer among stops, we are going to use the info provided by the TMB to know the walking time. Finally, once the user has arrived to the destination stop, we use the services of Google Maps again to calculate the last stretch to the destination.

With all that information we obtain a good estimation about the time we have to use from one point to another of the city using the public transports.

APPLICATION

The installed application at the mobile devices is easy due to the main target is that the user late as less as possible to obtain the route and the estimated time. This is so, because in general, people use this type of services when they have hurry. For that reason the number of screens to show is small and with only the necessary information.

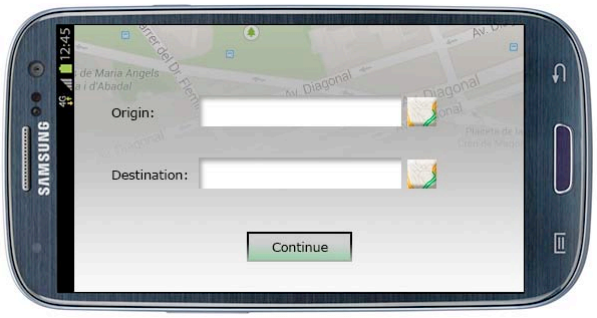

Figure 4. Selection Screen.

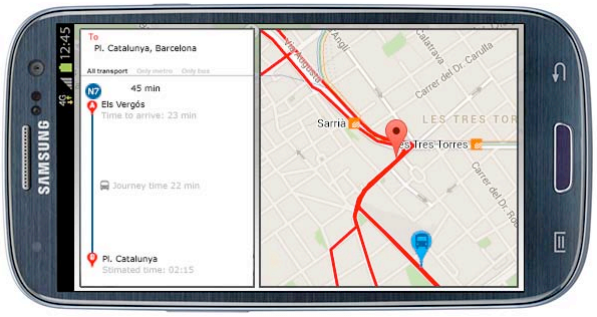

Figure 5. Trip Screen.

As we can see at the previous figure, the amount of information screens is negligible. We only have to provide few data to obtain the route and thus, to get the information shortly. Thus, in a glance the user knows what s/he needs.

CONCLUSION AND FUTURE WORK

With this paper we have stepped forward to get a more accurate estimation about the time necessary to reach a destination by mean of the services provided by the public transport of a specific city (in this case Barcelona). Thus, the users will have a more concrete notion about the time they have to use to move themselves from a point to another and thus, they can schedule appointments with a smaller error margin.

In this particular case we have treated the time needed from a public transport stop to the final stop. In this route are also included the transfers and the time among changes. However, we haven't taken into consideration the time needed to arrive to those stops, therefore in future extensions we will take the services provided by Google to control it and thus, to obtain even more details about the walking time from a determined point to another. For example, from our house to the first stop and from the last stop to our destination.

REFERENCES

1. Luigi dell'Olio, Angel Ibeas, Patricia Cecin. The quality of service desired by public transport users. Transport Policy, Volume 18, Issue 1, January 2011, Pages 217–227. http://dx.doi.org/10.1016/j.tranpol.2010.08.005
2. Niels van Oort, Daniel Sparing, Ties Brands, Rob M.P. Goverde. Optimizing Public Transport Planning and Operations using Automatic Vehicle Location Data: the Dutch Example. Models & Technologies for Intelligent Transportation Systems. 2013.
3. P. G. Villanueva, R. Tesoriero and J. A. Gallud. Revisiting the Concept of Distributed User Interfaces. In Distributed User Interfaces: Usability and Collaboration. Springer, Human-Coomputer Interaction Series. Eds. M. D. Lozano, J. A. Gallud, R. Tesoriero, and V. M. R. Penichet. ISBN: 978-1-4471-5498-3, pp. 1-15. 2013. url= http://dx.doi.org/10.1007/978-1-4471-5499-0_1
4. Google Maps. www.google.com/maps
5. TMB (Transports Metropolitans de Barcelona). http://www.tmb.cat/en/home
6. Rodalies de Catalunya. http://www20.gencat.cat/portal/site/rodalies/?newLang=en_GB
7. FGC (Ferrocarrils de la Generalitat de Catalunya). http://www.fgc.cat/eng/index.asp.
8. Gabriel Svennerberg. Beginning Google Maps API 3 (Expert's Voice in Web Development). 2010. ISBN-13: 978-1430228028.
9. Evangelos Petroutsos. Google Maps: Power Tools for Maximizing the API. 2014. ISBN-13: 978-0071823029.
10. Masaki Ito, Satoru Fukuta, Takao Kawamura and Kazanuri Sugahara. A Precision Navigation System for Public Transit Users. Distributed, Ambient, and Pervasive Interactions, Lecture Notes in Computer Science Volume 8028, 2013, pp 302-308. Print ISBN 978-3-642-39350-1. DOI 10.1007/978-3-642-39351-8_33.

Interaction Modality Mapping Service for devices in a P2P network

João Paulo Delgado Preti
Instituto Federal de Ciência e Tecnologia de Mato Grosso (IFMT)
CP 78.005-200 MT BR
joao.preti@cba.ifmt.edu.br
+55 65 8409-6882

Lucia Vilela Leite Filgueiras
Escola Politécnica da Universidade de São Paulo (EPUSP)
CP 05.508-900 SP BR
lfilguei@usp.br
+55 11 3091-5200

ABSTRACT
Distributed user interfaces are a trend in human-computer interaction, supporting applications in ubiquitous and collaborative computing. Yet, interoperability is achieved by the use of several protocols. In order to strengthen interoperability, a greater degree of standardization is still needed. In this paper, we propose a interaction modality mapping service for peer-to-peer service-oriented approach to attend cross-device and distributed user interaction. As a proof-of-concept, we developed the service and GUI components that allows devices services to be invoked through different interaction modalities. We concluded that the architecture is feasible and provide a rapid construction of solutions that exploit the interaction among multiple devices, multiplatforms and multiple users spontaneously, as devices enter and leave the network.

Author Keywords
Transitions, SOA, P2P, distributed user interface.

ACM Classification Keywords
H.5.2. User Interfaces: Input devices and strategies.

INTRODUCTION

The emergence of a variety of connected devices make up an infrastructure that encourages the natural growth of new interaction models. Computational devices tend to connect, applications, rather than compete, are working together, making user interaction more rich and complex. Talks-about new communication models and new human computer interaction demands are observed in studies by distributed user interfaces in [6], [7] and [8]. The paper [10] shows the relationship of the various media with the cloud computing and the importance of transparency device to compose a favorable scenario for the human cloud interaction.

Integrate different devices and systems and the skills to do it efficiently may present a barrier to use the potential of information systems and the various connected objects (devices and applications) disseminated in the environment.

Aiming to meet the requirements already presented in [13], an architecture with the following characteristics is presented:

- service oriented;
- DPWS stack based;
- organized as a Web Service Framework;

The use of SOA in the proposed architecture is suitable because:

1. There is tendency to implement SOA, in particular Web services standards (SOAP, WSDL, DPWS, etc..) directly on the devices [11], which enables a P2P network architecture;
2. There is the prospect of applications integration, i.e. distributed information can act in cooperation with corporate systems [3], which enables cross-application;
3. Standards used in the household are trying to incorporate the technology of Web Services, as the UPnP 2.0[1] and Microsoft[2] with its invisible computing platform, which enables cross-device;
4. There is a convergence of mobile and stationary systems where applications become a resource not litmited by the constraint of device and location [12].

DUI '14, July 01 2014, Toulouse, France
Copyright 2014 ACM 978-1-60558-724-0/14/07⋯$15.00
http://dx.doi.org/10.1145/2677356.2677667

[1] UPnP (*Universal Plug and Play*) is a distributed system, open network architecture, where devices are directly connected with each other at home, office and public spaces.
http://en.wikipedia.org/wiki/Universal_Plug_and_Play

[2] It is a research prototype for making small devices part of the seamless computing world. http://research.microsoft.com/en-us/um/redmond/projects/invisible/

This research assumes that Web services technologies are increasingly widespread in the platforms of heterogeneous devices over Internet Protocol. Nevertheless, the typical use of Web Services is not suitable, as it should be taken into account the limited processing capabilities and bandwidth of resource constrained devices.

The use of SOA in the devices is becoming common and refers to the Device as a Service (DaaS) concept. This concept is addressed in several studies, such as [1], [2], [4] and [9], and several of them point the DPWS specification as a promising one.

For the proposed architecture in this research we chose a direct communication model (P2P) because:

1. gives the device the ability to spontaneously connect with each other to perform an activity;
2. promotes a facilitated communication, plug and play style, without many configurations;
3. there is no hierarchy among communication nodes and there is not a compromising failure point, each device can be a consumer or provider, with no common dependence in the architecture to a single device.

This paper presents a service-oriented architecture that assist in the communication of devices that need to coordinate a distributed interaction using more than one interaction modality.

SOA technologies for devices has emerged like UPnP and Jini, but a promising technology for compatibility with Web Services (WS) is proposed by OASIS through Device Profile for Web Service (DPWS) specification [5].

Interaction Modality Mapping Service (IMMS) is being built on top of DPWS specification as architecture base for discovery and communication between devices. All devices in the distributed environment must have an implementation of DPWS specification.

Despite DPWS stack provide specifications for service description, location, security and events, it does not include some features needed for a distributed interaction scenario. For this reason, DISS (Distributed Interaction Support Service) existence becomes necessary. DISS was already presented in [13] and raises a set of services needed to organize a distributed interaction. In this paper we present IMMS, one part of DISS service architecture responsible for multimodality. IMMS rationale is described in the following section.

INTERACTION MODALITY MAPPING SERVICE (IMMS)

Given the different modes of interaction for each device type, we visualize an organization of services for its type. The existing arrangements for a smartphone interaction can be very different from those found on a TV. There are smartphones that allow interaction via keyboard, voice, video camera, multitouch screen and through sensors, such as accelerometer and proximity. TV interaction occurs almost always at the remote control keypad.

The IMMS is the service responsible for registering how an interaction, performed on a given device, invokes the services offered by another device. To achieve this behavior, the IMMS must have recorded the events that can be launched and act as an adapter for the correct invocation of the desired service. The relationship between actions and services is a many to many cardinality.

An example is shown in Figure 1, which is possible to visualize a smartphone device interested in the services of another device whatsoever. Actions taken on the smartphone represents user interaction with this device. These actions, trigger events that are intercepted by IMMS, which is responsible for interpreting the event and properly invoke the desired service on another device. A user action can be, for example, a multitouch gesture "to enlarge". This action can be configured to increase the volume of the TV, or zoom in on a display, or increase the acceleration of an engine.

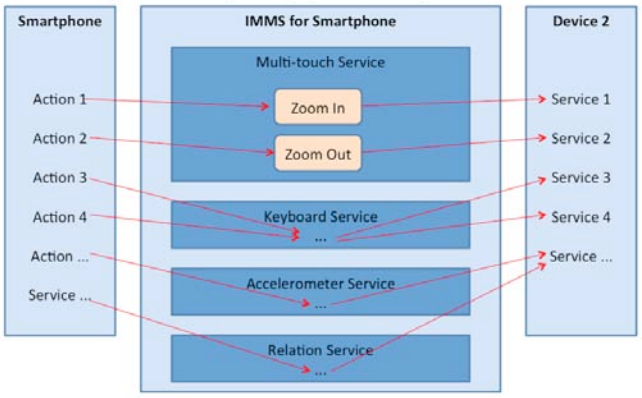

Figure 1. IMMS mapping user actions on a smartphone to another device.

Table 1 presents a hypothetical example of interaction possibilities of the use of a smartphone with the services offered by a TV and vice versa.

Table 1: Interaction between Smartphone and TV

SMARTPHONE	TV
Spreading two fingers	Increase volume
Approaching two fingers	Decrease volume
Swipe right	Channel up
Swipe left	Channel down
Double tap the screen	Change screen format (16:9;4:3)
Drag three fingers down	Show TV program schedule
TV	**SMARTPHONE**
Press up arrow	Go to previous contact
Press down arrow	Go to next contact
Press ENTER/OK	Show detailed contact data
Press Info	Show smartphone location

At the end of Figure 1 is also possible to observe the existence of a Relation Service, which is necessary to allow connections between services to be constituted. It allows chaining a set of devices in a single interaction. The call to a service can trigger a call to another service, allowing an action to take effect in more than one service (present in the same device or not), causing a cascading effect. We visualize the possibilities of using SOAP messages over UDP or stream strategy for events that can be fired continuously. The DPWS specification allows you to send SOAP messages over UDP or TCP, and also the realization of stream, which provides the flexibility to categorize events according to their nature and use a more optimized protocol or strategy if necessary.

The WS-Eventing specification allows creating complex topologies for events submission and processing, allowing the event source and the receiver to be decoupled.

Figure 2 shows how the IMMS is organized.

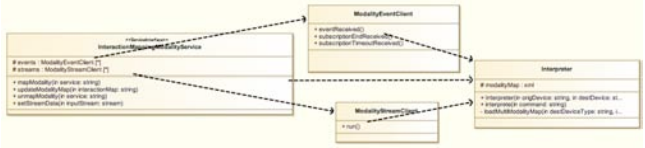

Figure 2. IMMS structure for events and stream processing.

The mapModality operation allows loading the modality formats recorded in a XML configuration file for the service the user wants to access.

The ModalityEventClient and ModalityStreamClient classes represents the communication format of the interaction, whether in the form of events, whether in the form of stream, respectively.

The Interpreter class is responsible for storing and retrieving the XML configuration file that structure the interaction modality map, as the example in Figure 3.

In Figure 3 is possible to verify that the operations setMouse (which sends mouse data streamed to other devices) of MouseService and okEvent (which sends data typed on the keyboard when pressed OK in the smartphone and ENTER on the desktop) of KeyboardService, are being mapped to TV devices services. It is possible to check that three mouse and one keyboard events were mapped:

- simple click on the left button (LEFT_ONE_CLICK): invokes the nextChannel TV operation only once (occurrence value = "1");
- simple click the scroll button (SCROLL_ONE_CLICK): invokes the showInfo TV operation only once (occurrence value = "1");
- hold down the left button (LEFT_PRESSED): invokes the operation volumeUP several times (occurrence value = "*"). The number of calls to volumeUp operation will depend on the positive variation of mouse x axis;
- type CHN: invokes the setChannel operation (changes the TV channel), the numeric argument entered after the CHN command sets the new value of the channel.

```
<multimodalDevices>
    <TV>
        <br.ufmt.xdevice.MouseService>
            <operation name="setMouse">
                <type>stream</type>
                <event type="LEFT_ONE_CLICK">
                    <service>br.ufmt.xdevice.TVService</service>
                    <operation name="nextChannel">
                        <occurrence value="1" />
                    </operation>
                </event>
                <event type="SCROLL_ONE_CLICK">
                    <service>br.ufmt.xdevice.TVService</service>
                    <operation name="showInfo">
                        <occurrence value="1" />
                    </operation>
                </event>
                <event type="LEFT_PRESSED">
                    <condition attribute="x" operator="GREATER_THAN" value="0">
                        <service>br.ufmt.xdevice.TVService</service>
                        <operation name="volumeUp">
                            <occurrence value="*" definedBy="x" />
                        </operation>
                        <condition>
                </event>
                <operation>
        </br.ufmt.xdevice.MouseService>

        <br.ufmt.xdevice.KeyboardService>
            <operation name="OkEvent">
                <type>event</type>
                <event type="CHN">
                    <service>br.ufmt.xdevice.TVService</service>
                    <operation name="setChannel">
                        <occurrence value="1" />
                    </operation>
                </event>
            </operation>
        </br.ufmt.xdevice.KeyboardService>
    </TV>
</multimodalDevices>
```

Figure 3. XML document that maps TV services to mouse and keyboard events.

The sequence diagram in Figure 4 shows an example of IMMS using event-driven mechanism.

In this mapping, the keyboard service is mapped to interact with TV through text commands. Already Figure 5 shows an example using IMMS via stream.

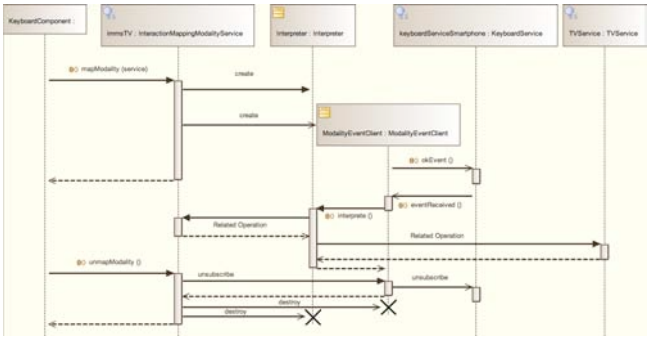

Figure 4. Keyboard operations mapped to TV operations through events.

In this other example the mouse service is mapped to interact with the TV via mouse click actions. The keyboard and mouse actions of the two samples can be investigated in the XML document described in Figure 3.

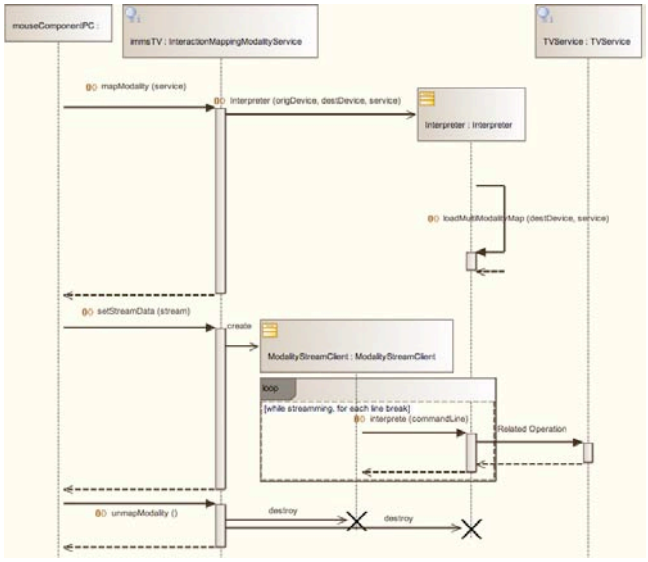

Figure 5. Mouse operations mapped to TV operations through stream.

For demonstration purpose take the example of our XMouse GUI component. This component is responsible to create a virtual cursor in the destination device, allowing controlling a remote cursor when the cursor service exists in the destination device.

Figure 5a shows the implemented XMouse desktop GUI component. Figure 5b shows the result of virtual cursors in another desktop device controlled by XMouse component.

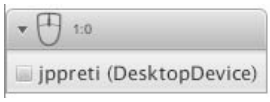

Figure 5a. XMouse component.

Figure 5b. Cursors controlled by XMouse.

XMouse in its initial version was able to communicate only with devices that support the creation of virtual cursors (like desktop devices). With IMMS, XMouse component is able to communicate with devices for other intents than the virtual cursor (as controlling TV operations).

CONCLUSION

The experiments presented in this paper are helping in the definition of a service that allows services to be linked with different interaction modalities with a distributed interaction nature.

Device services in conjuction with IMMS can significantly ease the implementation of a planned and distributed interaction and offers benefits of interoperability, usability, reusability and deployability due to its service nature. However, we understand the need of standardization of devices services.

REFERENCES

[1] Andreas Muller et al. 2009. An assisted device registration and service access system for future home networks. (Dec. 2009), 1–5.

[2] Bohn, H. et al. 2006. SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains. *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on* (Apr. 2006), 43.

[3] Bronsted, J. et al. 2010. Service Composition Issues in Pervasive Computing. *IEEE Pervasive Computing*. 9, 1 (2010), 62–70.

[4] De Deugd, S. et al. 2006. SODA: Service Oriented Device Architecture. *Pervasive Computing, IEEE*. 5, 3 (Sep. 2006), 94 –96.

[5] Devices Profile for Web Services Version 1.1: 2009. *http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html*. Accessed: 2014-01-28.

[6] DUI 2011. DUI 2011 Distributed User Interfaces. (Vancouver, BC, May 2011), 109.

[7] DUI 2012. DUI 2012 Distributed User Interfaces. (Austin, TX, USA, May 2012), 74.

[8] DUI 2013. DUI 2013 Distributed User Interfaces. (London, UK, Jun. 2013), 75.

[9] Jammes, F. et al. 2005. Service-oriented device communications using the devices profile for web services. *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*. (2005), 1–8.

[10] João Paulo Delgado Preti and Lucia Vilela Leite Filgueiras 2010. Interação em Nuvens. (Belo Horizonte, MG, Oct. 2010).

[11] Karnouskos, S. et al. 2009. Towards the real-time enterprise: service-based integration of heterogeneous SOA-ready industrial devices with enterprise applications. *Proc. of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'09)* (2009), 2127–2132.

[12] Marc McLoughlin 2009. ECSCW Workshop: Bridging Interaction Clouds: Exploring collaborative interaction across assemblies of mobile and embedded technology. *ecscw workshop: bridging interaction clouds*.

[13] Preti, J.P.D. et al. 2014. Transitions: a crossmedia interaction relevant aspect. (Hawaii, USA, Jan. 2014).

Non-Functional Requirements for Distributable User Interfaces in Agile Processes

Mohamed Bourimi
MT AG – Business by Integration
Balcke-Dürr-Allee 9
(40882) Ratingen, Germany
mohamed.bourimi@mt-ag.com
+49 173 8605799

Ricardo Tesoriero
University of Castilla-La Mancha
Campus Universitario de Albacete
(02071) Albacete, Spain
ricardo.tesoriero@uclm.es
+34 967599200 (ext. 2295)

ABSTRACT

This paper presents a two-folded approach to deal with non-functional requirements for distributable user interfaces (DeUIs) in agile processes. This proposal employs a conceptual agile framework that ensures earlier consideration of non-functional requirements and stakeholders' involvement to solve tensions among agility, requirements engineering practices and continuous system architecture adaptation. Besides, it improves the step of continuous architecture adaptation as established in the DeUI field by employing model-driven architectures. Thus, while this approach profits from the conceptual framework by means of continuous feedback on how to technically better support the classical tension between agility and requirement engineering; it also takes advantage of model-driven architecture to cope with the tension between agility and distributable user interface architecture changes.

Author Keywords

Agile methodologies, Distributable User Interfaces, Model-driven development, Scrum, AFFINE.

ACM Classification Keywords

H.5.2 User Interfaces: User-centered design, H.5.2 User Interfaces: Graphical user interfaces (GUI), I.6.5 Model Development

MOTIVATION

The cost reduction of digital displays has encouraged the use of them almost everywhere. They can be both, stationary (i.e. Smart TVs, advertising displays, interactive information point displays, etc.) or mobile (i.e. Smartphone, Tablets, etc.).

Due to the advances in the communication infrastructure, these devices can be easily connected to define a display ecosystem [1] where users are able to interact with more than one interaction display at the same time.

One way of taking advantage of the resources that are part of a user interface (UI) Ecosystem is the distribution of UI components among devices that are available within the ecosystem. According to [2], 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. Some cases of study where DUIs were successfully developed are exposed in [3] and [4].

The DUI concept was redefined in [5] to describe a state of a UI, instead of a type of UI. Consequently, the distributable user interface (DeUI) concept emerges to define UIs that can be distributed among different platforms in runtime. The DeUI concept is defined using a metamodel that describes the distribution characteristics of UIs in terms of metaclass instances that specify

4th Workshop on Distributed User Interfaces and Multimodal Interaction
DUI'14, July 01 2014, Toulouse, France
Copyright 2014 ACM 978-1-60558-724-0/14/07$15.00
http://dx.doi.org/10.1145/2677356.2677660

the UI distribution model [6]. Some examples of DeUIs can be found in [25, 26]

There are different approaches to develop DeUIs; however, they are leaving the research status to become more mature in order to satisfy sensitive industrial requirements [7]. Moving interfaces from one screen to another screen or device as we know them from famous movies (e.g. Minority Report, Paycheck or Avatar) are no more fiction but becomes a reality.

The distribution aspect, as defined above, allows the distribution of information among different displays or UIs running on different platforms, which are often used in collaborative settings, adding so another dimension of complexity. Indeed, this "*distribution*" aspect in a given usage situation needs addressing emerging issues (e.g.; related to trust, security and privacy requirements). Especially when DeUIs are used within collaborative settings, further requirements such as group and social awareness, usability as well as tailorability (i.e. adaptability) requirements have to be considered. The majority of these requirements are primarily related to non-functional requirements (NFRs) [29].

The major challenge from the software engineering perspective consists since decades in properly addressing those requirements in (i) early system design as well as (ii) continuously aligning them with the resulting system architecture in latter adaptation/maintenance phases of emerging systems and applications. DeUIs and socio-technical systems (i.e. groupware or social software) in general demand frequent evaluation, collection of feedback and experiences with respect to unexpected human behavior. Nowadays, user-centered and participatory design metho(dologie)s with different degree of agility are used to build such socio-technical systems. Agility and earlier end-users' involvement increases the frequency of change requests on both; business processes and development level (business process agility; agile development). In this paper, we address agility at both levels. The meaning of agile processes is put here on developing systems as well as considering dependencies to business in general (i.e. organizational management levels). Agile method(ologi)es often just focus on development activities while business process agility demands stronger consideration of constraints emerging from organizational dependencies to IT Risk Management, IT Compliance, and Multi-project Controlling, and stakeholders involvement concerns across the organizational units etc. This increases tensions among agility, requirements engineering practices and continuous system architecture alignment to changing business needs.

In this paper, we present a two-folded approach to deal with NFRs in early stages of agile development of distributable user interfaces (DeUIs).

To cope with this challenge we employ the characterization of the UI distribution presented in [5]. This characterization is based on the definition of metamodels that are part of a model-driven architecture (MDA) designed to develop DeUIs. The MDA presented in this paper is an extension of [8] that allows the modeling of NFRs such as, privacy and security, of traditional UIs.

The remainder of this paper is structured as follows; it starts with the analysis of the problem that presents NFRs when developing DeUIs in agile processes. Then, it presents how to address this problem using a model-driven architecture and discusses the first findings of adopting this approach. Finally, we expose conclusions and future work.

PROBLEM ANALYSIS AND RELATED WORK

Background information

End-users are more and more taking control over the designed systems. For instance, the range of devices to be supported in different business scenarios increased rapidly over the last years. A single end-user disposes nowadays of many (mobile) devices, with different capabilities (UI, operating system, etc.) and uses all of them interchangeably along the day depending of its leisure or professional needs.

Another emerging need in industrial settings is to support all categories of users (young or elderly persons, lay or expert users, etc.) when providing applications that act as touch points to the targeted audiences. One could consider for instance multi-channel marketing (MCM), e.g., by using social media where the designed system must support responsive design for different devices/platforms. Furthermore, the direction of interaction is no more from marketers to consumers but should also be supported in the opposite way, so that consumers are enabled to provide information (e.g.; commenting pictures) or trigger further interaction (i.e. as a reaction on an offer by chatting with the marketers or their agents).

The reader may get an impression on which complexity the system could reach when considering that the generated context-rich content should support responsive design capabilities (for different platforms), maybe track the user interaction/navigation within the content (on page or click level) in order to provide analytics to the marketers, and last but not least, include anchors for bidirectional interactivity, e.g., in order to support commenting artifacts, setting location spots or (instantly and securely) communicating with others, etc.

As mentioned before, the design of interactive and collaborative systems is lately focusing on agile methodologies as a way to efficiently involve all stakeholders from the beginning when building such systems. In industrial settings agility is being followed to reach faster release cycles in order to meet "time to market" challenges. Latter are increasingly influenced by change requests of the customers (e.g., getting car parking tickets with my handy, shopping by scanning barcodes while walking through a corridor in a subway, etc.). In industrial settings, e.g. for the MCM example above, increasing users' numbers within a successful campaign could produce in performance bottlenecks and require availability improvements. Furthermore, MCM involves different stakeholders from marketing, content management creation teams (e.g., layout designers), usability and user experience engineers, business analysts, as well as approval committees (legal and information security, and data protection officers, for checking content on IT compliance, IT product managers for allocating IT operations capabilities, etc.).

Tension field and identified needs

In the following we accurately highlight the tension fields that emerge in such settings when building sophisticated DeUI and collaborative systems as we experienced in our (academic and industrial) project settings by referencing related work:

- The complexity of the business domain by satisfying business agility NFRs while considering multi-laterality of stakeholders and their (often conflicting) goals is needed (**Tension Field 1**; **TF1**). For instance, Cremers and Alda mention in organizational requirements engineering that "project management issues (costs, time, schedule) are often considered as non-functional requirements as well"; however, at the project organizational/management level [9].
- The majority of literature addresses NFRs from the development perspective. NFRs at development levels are known as a classical problem area [10]. A very important finding is that NFRs are often conflicting and their nature as well as alignment differs according to involved (research) areas within a given business case in particular or project in general (Usability, Security and Data Protection, etc.). For instance, confidentiality could not be handled like availability and integrity within the Security research area [24]. Furthermore, there are findings that (N)FRs' alignment problems cannot be fully solved using automations [11] (**TF2**).

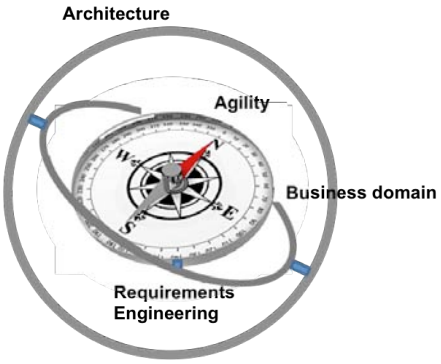

Figure 1: Tension filed in agile processes

- Mature industrial software is, however, a final product of a software life (production) process (s. **TF1**). Producing complex software should align the dimensions depicted in Figure 1 [12]. Regarding the tension between agility and requirements engineering (RE), practitioners of agile methodologies stress, that adequate support for NFRs is not provided in agile method(ologie)s in general (e.g., it is not easy to consider NFRs in user stories). Paetsch et al. denotes in [13] that: *"Requirements engineering, […] is a traditional software engineering process with the goal to identify, analyze, document and validate requirements for the system to be developed. Often, requirements engineering and agile approaches are seen being incompatible: RE is often heavily relying on documentation for knowledge sharing while agile methods are focusing on face-to-face collaboration between customers and developers to reach similar goals"*. In agility, refactoring hell with respect to continuous Software Architecture (SA) alignment in agile settings is well-known example. (**TF3**)
- Development methodologies and processes in general try to reach an engineering-oriented approach (automatable, measurable, etc.). Model-driven development offers an opportunity to reach this. This could be in our case of benefit since current research work show that MDA could be profitable towards engineering DeUI. However, considering NFRs and their continuous alignment could be a challenge especially when following agile methods (cf. **TF3**, Agility vs. Architecture Design). In our case an engineering-oriented automation level should be well integrated in DeUI design, implementation and evaluation since we already elaborated first steps in this direction (e.g., in [8]). The new need consists in supporting it in agile processes for industrial settings we are involved in. Catching interdependencies between organizational and development levels as well as solving NFR issues that could not be fully automated is a main requirement thereby. (**TF4**). Note that even though agile method(ologi)es could produce a qualitatively good product backlog, there will be still unpredictable breakdowns in the planning, which results for instance from decision to be taken at development level. Another ones are often not related to technological challenges such as decisions to freeze or abort projects due to budget restrictions or re-prioritization.

With simple words in our case, concretely for building complex DeUIs within agile (industrial) processes, it is required to address NFR related problems at both levels; organizational one, e.g., reaching consensus among stakeholders' multi-lateral goals; as well as at development level, i.e., for conflicting or contradicting NFRs and FRs (**Need 1**; **N1** with respect to **TF1** and **TF2**). Thereby engineering the process as much as possible by refining the metamodels of our emerging DeUI MDA framework by considering won best practices and/or lessons learned (**N2** with respect to **TF3** and **TF4**).

APPROACH

The big picture

The industrial adoption of DeUIs is reflected in our two main needs (N1 and N2) requires a two-folded approach. Tremendous literature from

different research communities try to overcome drawbacks identifies agile methodologies (NFR, RE vs. Software Architecture) contemplated from each other (representing so micro-approaches[1]). We argue that a methodological macro-approach, integrating NFR early consideration and alignment for DeUI in agile processes should be/ build the first part of any approach (i.e., by also considering/integrating organizational NFRs, so for agile processes within complex business domains, e.g., MCM in general[2]) (N1). Indeed, many projects fail in industrial settings due to organizational NFRs and their dependencies to other levels (development, IT operations, IT compliance, changes due to emerging regulations and legal directives, etc.). The second part of our approach towards engineering-oriented improvements consists on MDA techniques addressing NFRs for DUIs in agile processes (N2). In this paper, we present a proposal on how a MDA approach addresses NFRs, such as privacy, security and UI distribution (in continuation of our previous work described in [9]).

The methodological part

One could argue that our identified needs (N1-2) could be seen as high-level requirements to any software development process. Indeed, however, and as mentioned before, agility and the software engineering as disciplines are still young. A process and meth(odologie)s tailoring respective development processes are taking place in different research fields, industrial settings with different focuses. Such processes' tailorings or/and improvements (based on existing ones, e.g., integrating CMMI and Scrum) lead to increasing process complexity. We argue that this is contradicting in some degree the philosophy of "agility in software engineering" (mainly at development level). For this, we integrate the Agile Framework For Integrating Non-functional requirements Engineering (AFFINE) as an integral part of our approach [14]. We argue that, AFFINE is not just another agile method(ology); instead it defines a conceptual framework consisting in its turn of two integral parts: (a) the method and (b) its supporting technology (i.e. architecture). In addition, AFFINE conceptually handles all NFRs at the earliest stage of the development process, which in effect follows a kind of a "big-front up effort" approach (cf. [12]). As result, the backlog considers all NFRs, involves experts for all NFRs of interest, and helps to constitute the adequate capability of teams while ensuring traceability of requirements (which is not often explicitely done in documentation of agile methods; **N1**).

One of the main design rationales of AFFINE consists in considering human factors (i.e. stakeholders and experts involvements) and integrating NFR consideration as well as negotiation techniques and artifacts into Scrum (see [14]) to ease the communication across the whole phases of the software life cycle among involved stakeholders. The current version of AFFINE extends used practices to ease the consideration of dependencies to organizational NFRs. However, keeping the simplicity of the agile methodology intact so far as possible. So, inherent dependencies to often-competing NFRs of various degrees and at different levels (strategic, organizational, development, and operational) are considered by keeping boundaries to other Frameworks such as ITIL[3].

---

[1] For clarification: Approaching problems by dividing them in research is legitimate proceeding (divide and conquer philosophy), however, for industrial settings and since DeUIs are in adoption/maturity evaluation process, a coarser approach is needed.

[2] The reader may consider again the footnote one with respect to agile processes vs. agile methodologies (as we perceive them in this contribution).

[3] IT Infrastructure Library as a set of best practices for IT service management (ITSM), which focuses on aligning IT with business needs/requirements.

As we argued earlier in this paper, a macro-approach is needed to address our needs. The big picture of our approach is depicted in Figure 2.

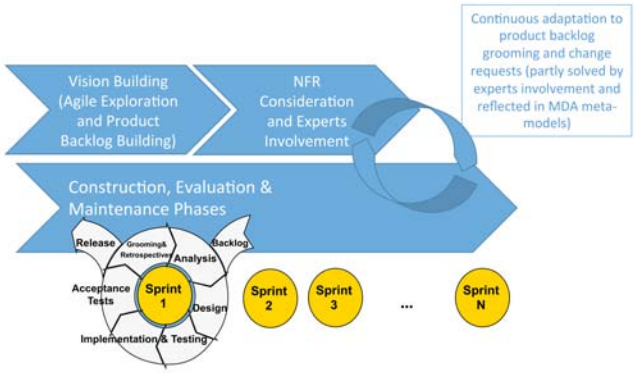

Figure 2. Agile Addressing NFRs for DeUIs

Figure 2 depicts the concrete parts of our conceptual framework towards integrating the best –so far as feasible- from agility, RE and SA by focusing on software process improvement (SPI) requirements specific to a given business domain (DeUI for MCM for instance). Again, we argue that in our case the need for such conceptual convergence-driving approach at this macro-level is crucial since existing approaches are (legitimately) focusing on specific aspects between both extremes (i.e. the SA's Up-Front and the Just-So-Far-As-Needed agile philosophies). Related work address often bottom-up- or top-down-focused tunings but not sufficiently their convergence. Figure 2 is based on collected crucial SPI requirements (SPIRs) by using AFFINE within DeUI as well as other projects (i.e. building tailorable groupware). It coarsely shows how we systematically try to enforce consideration of our needs within two integral phases of our framework: (P1) a vision alignment phase (see. Figure 2), and (P2) the phase of agile solution's requirements mapping onto the agreed vision (see Figure 2, Construction, Evaluation, and Maintenance Phases). While P1 is performed as an agile quality-driven sub-process enforcing the earlier consideration of organizational and requirements engineering SPIRs, P2 incorporates established best practices and anti-patterns for ensuring continuous SA validation at development and operational levels; and this while considering continuous changes in NFRs agreed in P1 in a balanced and customer's needs oriented way, namely, by using MDA for DeUI while considering identified NFRs (TF1-4).

The role played by MDA in P1 is involves the definition of the virtual (e.g. buttons, entry field, canvas, etc.) and physical (e.g. displays, smart-panels, etc.) resources that are part of the display ecosystem as well as NFRs such as, principals, access control and anonymity attributes, etc. In P2, the MDA defines a set of marking models that relates NFRs to resources. The aim of these models is filling the gap among MDA models that cover different concerns of the DeUI system in order to generate source code for catching up dependencies across involved levels, the a single (however, versioned) document is used (e.g. traceability of NFR alignment decisions).

The Figure 2 depicts therefore a conceptual framework that can be tailored with means of P1 and P2 in order to meet concrete customers' situation (in terms of a balanced agile and existent architecture-considering SPI implementation). Thereby P1 and P2 can be considered as concrete SPIRs alignment and dependency checking mechanisms at the meta-level. In our case, we used MDA in order to reach the big-front effort approach in P1 in various projects (high-fidelity rapid prototyping with MDA for building a common vision among stakeholders, identifying NFRs of interests, needed experts, staffing capabilities, e.g. for development teams, etc.) [3,4, 15].

Considering the interplay between agile methodology and MDA

Some NFRs could be broken down into requirements that can be automatically checked (e.g. availability could be verified in terms of performance response times, integrity could be assessed by means of unit tests, etc.). However, other NFRs are difficult to be qualitatively or quantitatively evaluated.[4] For the formers, we

[4] For instance, usability, confidentiality and consideration of privacy as well as trust

showed in [9] how metamodels could address some issues regarding breakable NFRs in DeUIs. However, as we have mentioned in the Motivation section, there are non-breakable NFRs that introduce tensions in the development process between (a) requirements engineering and agility and (b) between agility and architecture design (N2).

The use of a MDA to define different concerns of DeUIs allows the definition of different metamodels for each aspect of the system to be developed (e.g. NFRs such as, security and privacy). Even though, models that are part of the MDA are used to generate semi-automatically the source code of the system, these models can also be validated according to constraints defined in the metamodels. Consequently, it allows the validation of NFRs in early stages of the development process (event before the system is built).

Our macro-approach consisting in following the proposed steps of our conceptual framework (see Figure 2) suggests the involvement of experts in order to early and adequately addresses NFRs (N2). Due to the nature of some NFRs that could not be addressed with fully automated techniques (the engineering approach), involvement of experts reduces the complexity of this problem. For catching up dependencies to later changes within the construction, evaluations, and maintenance phases, AFFINE is used (cf. consensus finding iteration by circulating a communication artifact in form of an AFFINE document [14]).

The continuous alignment with means of MDA in our case targets considering (a) tensions between requirements engineering and agility and (b) between agility and architecture design (distributed system architecture are expensive to continuously be maintained and tailored each time) (**N2**). Considering NFRs for DeUI with means of MDA is described in the following.

---

issues/requirements within the designed DeUI system.

The engineering part: Metamodeling DeUI NFRs

With respect to N2, from a MDA perspective, the software is specified at different levels of abstraction using the same or different Domain Specific Languages (DSLs).

The architecture usually divides the software representation into 3 layers. The highest level of abstraction model is the Computational Independent Model (CIM) that employs a DSL that is close related to the problem domain. This model is usually transformed into Platform Independent Model (PIM) that is a lower level of abstraction model that represents a computational solution of the problem. This model is independent of the implementation platform. Then, the PIM is linked to the Platform Model (PM) using a Marking Model (MM) to generate the Platform Specific Model (PSM) of the software including implementation platform details. Finally, the PSM is transformed into source code (Implementation Specific Model) using a model-to-text transformation.

Regarding the implementation, metamodels were defined in ECORE [16] enriched with constraints in OCL [17] following the OMG [18] standards. Model editors were developed as Eclipse plugins using the EMF [16] and GMP [19] technologies. Models are represented in using the XML Model Interchange (XMI) [20] language also defined by the OMG. Model-to-model transformations are defined in ATL [21] and model-to-text-transformations are defined in MOFScript [22].

The Cameleon Reference Framework (CRF) [23] is a unifying reference framework to model traditional multi-target UIs where the Tasks and Concepts layer of the framework is the CIM of the UI, the Abstract UI layer of the framework is the PIM of the UI, the Concrete UI layer is the PSM and the Context of Use is a transversal model that affects all layers (see Figure 3).

The metamodel exposed on Figure 4 was defined in [6] to introduce security and privacy issues into the MDA specified by the CRF.

Models conforming the PriS metamodel are linked models conforming the Task and Domain

metamodels using a marking model that associates Activity instances (defined in the PriS metamodel) to Task instances (defined in the Task metamodel) and Resource instances (defined in the PriS metamodel) to Class instances (defined in the Domain metamodel).

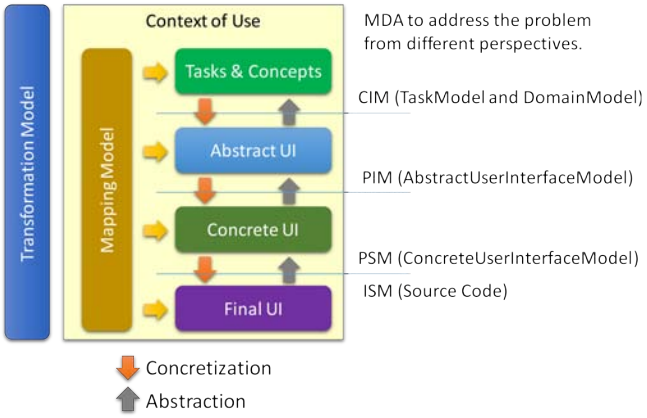

Figure 3. The Cameleon Reference Framework

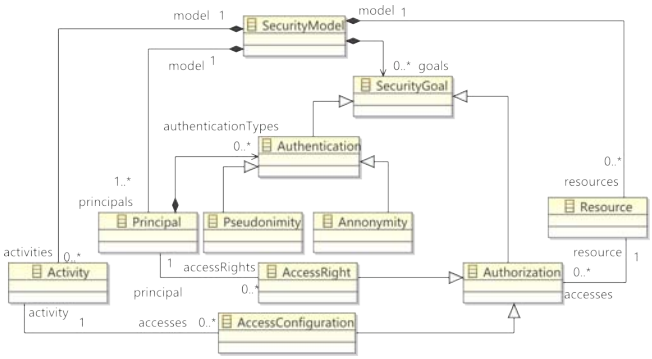

Figure 4. The PriS-oriented security metamodel

This approach is valid for UIs running in a single platform system because security or privacy issues are defined for the whole platform. However, in a DeUI environment platform conditions vary during UI system lifetime.

The Figure 5 shows the metamodel that describes the UI distribution characteristics of a DeUI [5]. Models conforming this metamodel describe the UI Distribution features which are related to AUI models using a marking model that links Interaction Object instances (defined in the UI Distribution metamodel) to Abstract Interaction Object instances (defined in the Abstract User Interface metamodel).

According to the UI distribution metamodel, a Distributed User Interface (DUI) as a state of a UI instead of a type of UI. A UI ecosystem reaches the Distributed State, if and only if, it defines at least 2 Interaction Surfaces that are hosted on different Platforms. As Interaction Surfaces are Interaction Containers, they host at least one Interaction Component each at a given time. It means that UI components belonging to a Distributed UI, which is running on a UI ecosystem, are distributed among UIs running in different runtime platforms. Therefore, a UI ecosystem is Distributable, if and only if, exists at least one Interaction Object that can be hosted in at least 2 Interaction Surfaces implemented on different Platforms. This concept of UI Distribution is a challenge from different perspectives.

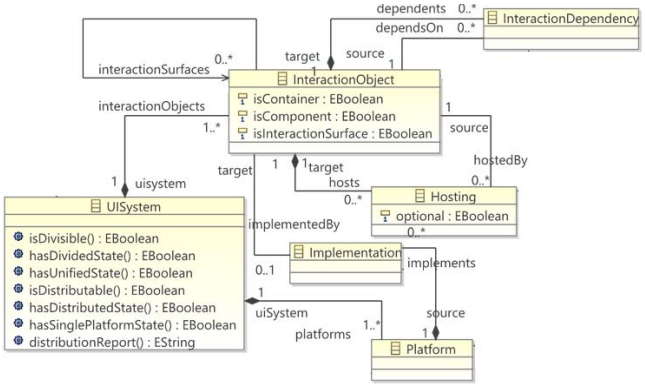

Figure 5. The UI Distribution metamodel.

Note that the UI Distribution model is defined at the PIM layer of the MDA because in a DeUI interaction objects can be distributed among different platforms at runtime. Therefore, in a DeUI, the PriS model affects both, Platform and Interaction Object instances.

To model this situation, we define a new marking model that relates Platform and Interaction Object instances (defined in the UI Distribution metamodel) to Resources instances (defined in the PriS metamodel).

Consequently, the proposed MDA provides traceability in terms of security and privacy among: data (as PriS Resource instances), tasks (as PriS Activity instances), interaction objects and platforms (also as PriS Resource instances).

Although this approach covers most of security and privacy issues, there still exist some

particularities that cannot be modeled using this MDA. For instance, those issues exposed in the Introduction section of this paper.

DISCUSSION

The following contribution goes beyond last research carried by us (i.e. [14, 9, 15]) in the following directions:

- Primarily considering organizational NFRs (such as aligning multi-lateral stakeholders' or technological requirements that are also often conflicting), which is increased need for DeUI within their industrial adoption process.
- Reflecting the result of our research in form of a conceptual agile framework (macro-approach) in order to catch up important dependencies (following the anti-pattern approach, especially because agility and software engineering are still considered as young disciplines in the research and industrial fields).
- Targeting an engineering-oriented approach by using MDA for NFRs in DeUI projects in agile processes.

However, it is difficult to qualitatively and/or quantitatively compare our approach to other ones. However, first collected experiences show the benefits of our approach, so:

- Our approach helped to early catch up dependencies by building DeUI prototypes for SocialTV, Cloud Computing or Distributed Decision Making settings where NFRs such as Authentication, Anonymity, and Pseudonymity are crucial to the success of the application. For instance, experts' involvement helped to identify conflicting NFRs (awareness in a collaborative UI and privacy concerns of the end-users) or design decision that could to violations (e.g. information security violations when users share information using a shared display, i.e. a reflection on a wall). In various scenarios, experts analyzed different situations based rapidly generated MDA prototypes and were able to detect violations, design balanced solutions in consensus among stakeholders.
- Reducing for instance the refactoring hell and accumulating knowledge in the developed metamodels (which could be reused/extended for different business domains, e.g. MCM with DeUI instead of SocialTV or Cloud Computing, etc.). Using MDA for high-fidelity rapid prototyping approach in various projects (according to the big-front effort proceeding) helped to sharpen and reduce efforts in general in the agile development phases. However, limitations in our settings (fixed budget, single track development, etc.) are not representative enough to emphasize the relevance of observations made in this direction.

Strictly speaking, from the modeling point of view, UI distribution requirements should be part of the CIM; however, as both, the Abstract UI and the UI Distribution metamodels share the Interaction Object concept, UI distribution characteristics are introduced at the PIM.

Nevertheless, linking a potential UI Distribution metamodel defined at the CIM level to the models defined at the Tasks and Concepts layer of the CRF would introduce semantic relationships that are relevant to the user experience.

As Tasks and Concepts models provide information of the system from a higher level of abstraction, there is information that could be derived from these models to improve the reusability and maintainability of the software.

For instance, suppose that the Task model of a DeUI system defines the "Enter Name" and we perform a Task to Abstract UI model transformation [27] that generates 2 Interaction Objects representing an input component to introduce the information and an output component that identifies the input component. By using the metamodel defined in [5, 6], you

have to create 2 Interaction Objects to link them (one per Interaction Component); however, if a UI Distribution model is defined at the CIM level, you have to define only one artifact to link the task. Consequently, the transformation would automatically create 2 Interaction Objects linked to both Interaction Components.

Using a MDA approach to develop applications increases the productivity by incrementing the software reuse, decrementing maintenance costs, increasing development speed, etc. [28].

Apart from some conceptual issues, we consider that metamodeling for DeUIs is mature at the moment; therefore, we are not discussing in this paper issues that emerge in large scale integration projects.

By employing MDAs, this approach deals with the tension between agility and architecture design. To cope with the tension between agility and requirement engineering, we employed the AFFINE methodology.

Just to explain how our approach helped in detecting trade-offs among just 3 NFRs (awareness, confidentiality and maybe privacy). Imagine the complexity for distribution and other NFRs of relevance.

Allowing for multiple identities in general and showing such identities at a given location breaks confidentiality in the means of linkability.

Suppose that a mobile application allows mobile anglers to see who is angling at watercourses around them (i.e. river, sea, etc.). It presents anglers mapped onto the map within the application, as a way of workspace awareness, and you see two persons in your application map. However, when you look into the watercourse with your eyes you see just one person. Therefore, you infer that this person has two identities.

To solve this problem, the chat application was designed to support just one identity. However; another problem arises.

When you chat often with a person using two identities (A and B) without you knowing it, you perceive that every time identity B is logging in to the chat, identity A is logged off. It is the consequence of the chat restriction to support just one identity of the same person at the same time. Thus, you infer that identity A is linked to identity B.

To solve this problem, logging out pseudo-delays were adopted to avoid the simultaneousness of logging in and out, which makes the linkability more difficult.

Strictly speaking, from the modeling point of view, the use of a MDA supports the NFR modeling and the development process. While the representation of NFRs (i.e. security and privacy) are modeled using the PriS metamodel, the development process is supported by the CRF metamodels and the UI distribution metamodels.

Regarding the NFR representation, this MDA allows the validation and verification of models to ensure both, the inter-model coherence and the intra-model consistency. On the other hand, the representation of the UI characteristics allows the use of transformation engines to generate UI system source code.

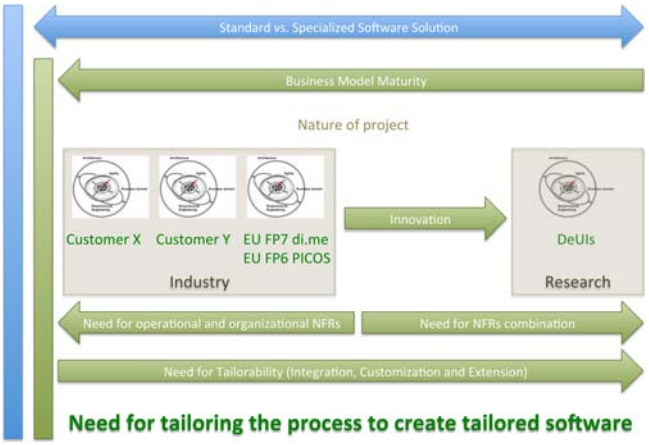

Figure 6: Context and contribution of the proposal

This proposal was successfully applied with agile evaluation and user-centered design to different cases of study. To shows the viability of the approach we present 2 projects where AFFINE and metamodels were applied [30].

The Figure 6 shows a diagram that summarizes the contribution of this paper. It describes the context of the contribution in the field of specialized software, concretely speaking, in the field of DeUIs. The maturity of the business model presented in this paper is supported by industrial and research projects. From the industrial perspective, this approach was employed in two commercial and two EU projects (di.me and PICOS). From the research perspective, it supports the DEINUDI National project which is focused on the application and development of DeUIs. The proposal provides a framework to support the customization of operational and organizational NFRs as well as the combination of NFRs from different concerns in order to satisfy the need for tailoring the methodology and the process to create tailored software.

CONCLUSION AND FUTURE WORK

The interleaving of the two parts results in a big benefit on the software development process because: (a) it supports frequent changes on the distributed UI architecture efficiently and (b) AFFINE conceptual framework considers earlier NFRs by building a circle which can be repeated along evolution of systems with technical means for solving NFR challenges.

As consequence, this proposed approach in this contribution avoid tensions between agility and architecture changes as well as agility and requirement engineering. Therefore, those aspects, which are not covered by the MDA, are covered by Affine (i.e. not fully automatable solutions to address NFR issues). To show the validity of our proposal we presented and explained anecdotally (by referencing our projects, collected experiences, and selected relevant related work) cases where the approach was successfully applied.

As future works, from the MDA perspectives, we are working on a UI distribution metamodel at the CIM level of the MDA in order to model distribution concerns at the Tasks and Concepts Layer of the CRF.

ACKNOWLEDGMENTS

We thank the CICYT-TIN 2011-27767-C02-01 Spanish project. Further thanks are due to Thomas Barth, Peter Heintzen and Thomas Eschke for discussions with respect to organizational requirements dependencies.

REFERENCES

1. Terrenghi, L., Quigley, A. and Dix, A. A taxonomy for and analysis of multi-person-display ecosystems. *Personal and Ubiquitous Computing*, 13, 2009, 583–598.
2. Vanderdonckt, J. Distributed User Interfaces: How to Distribute User Interface Elements across Users, Platforms, and Environments, in *XI INTERACCIÓN*, (Valencia, Spain, 2010), 20-32.
3. Barth, T., Fielenbach, T., Bourimi, M., Kesdogan, D. and Villanueva, P. Supporting distributed decision making using secure distributed user interfaces, in Gallud, J.A., Tesoriero, R., Penichet, V. M. Eds. *Distributed User Interfaces*. Human-Computer Interaction Series. Springer London, 2011, 177–184.
4. Heupel, M., Bourimi, M., Schwarte, P., Kesdogan, D., Barth, T. and Villanueva, P. G. Enhancing the Security and Usability of DUI Based Collaboration with Proof Based Access Control, in Lozano, M. D., Gallud, J, A. Tesoriero, R. and Penichet, V. M.R. Eds. *Distributed User Interfaces: Usability and Collaboration*. Human–Computer Interaction Series. ISBN: 978-1-4471-5498-3. Springer London, 2013, 95-105.
5. Villanueva, P. G., Tesoriero, R. and Gallud, J. A. Revisiting the Concept of Distributed User Interfaces., in Lozano, M. D., Gallud, J. A., Tesoriero, R, and Penichet, V, M.R. Eds. *Distributed User Interfaces: Usability and Collaboration* Human Computer interaction Series. ISBN: 978-1-4471-5498-3, 2013, 1-15.
6. Villanueva, P. G. *Distributable User Interfaces*. PhD Thesis. Universidad de Castilla-La Mancha. June 2014.
7. Bandelloni, R. and Paternò, F. Flexible interface migration, in *9th international*

conference on Intelligent user interfaces (IUI '04). ACM, (New York, USA, 2004) 148-155.

8.Bourimi, M., Tesoriero, R., Villanueva, P. G., Karatas, F. and Schwarte, P. Privacy and Security in Multi-modal User Interface Modeling for Social Media, in *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE 3rd International Conference on Social Computing (SocialCom)* (Boston, MA, USA, 2011), 1364-1371.

9.Cremers, A. B. and Alda, S. Non-functional Requirements. Organizational Requirements Engineering. Chapter 9. URL = http://www.iai.uni-bonn.de/III/lehre/vorlesungen/SWT/RE05/slides/09_Non-functional%20Requirements.pdf (last visit 2014).

10.Chung, L. and Nixon, B.A. Dealing with non-functional requirements: three experimental studies of a process-oriented approach, in *ICSE 1995*. ACM, (New York, USA, 1995).

11. Chung L. and Prado Leite, J. C. On Non-Functional Requirements in Software Engineering. In Conceptual Modeling: Foundations and Applications in Alexander T. Borgida, Vinay K. Chaudhri, Paolo Giorgini, and Eric S. Yu (Eds.). LNCS 5600. Springer-Verlag, Berlin, Heidelberg (2009) 363-379.

12. Babar, M. A., Brown, W. and Mistrik, I. *Agile Software Architecture: Aligning Agile Processes and Software Architectures*, 1st edition. Morgan Kaufmann Publishers Inc. December 2013.

13.Paetsch, F., Eberlein, A., Maurer, F.: Requirements engineering and agile software development, in *12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003*, 2003, 308–313.

14. Bourimi, M., Barth, T., Haake, J. M., Ueberschär, B. and Kesdogan, D. AFFINE for enforcing earlier consideration of NFRS and human factors when building socio-technical systems following agile methodologies, in Bernhaupt, R., Forbrig, P., Gulliksen, J. Lárusdóttir , M. (Eds.) *3rd International Conference on Human-centred software engineering (HCSE'10)*. Springer-Verlag, Berlin, Heidelberg, (Reykjavik, Iceland, 2010), 182-189.

15. Bourimi, M. and Kesdogan, D. Experiences by using AFFINE for building collaborative applications for online communities, in Ozok, A. A. and Zaphiris, P. (Eds.) *5th International Conference on Online Communities and Social Computing (OCSC'13)*, Springer-Verlag, Berlin, Heidelberg, 2013, 345-354.

16.The Eclipse Modeling Framework (EMF). URL: http://www.eclipse.org/modeling/emf/

17.The Object Constraint Language (OCL). URL: http://www.omg.org/spec/OCL/2.3.1/

18.The Object Management Group (OMG). URL: http://www.omg.org/

19. Graphical Modeling Project (GMP). URL: http://www.eclipse.org/modeling/gmp/

20.The XML Metadata Interchange. URL: http://www.omg.org/spec/XMI/

21.The Atlas Transformation Language. URL: https://eclipse.org/atl/

22.The MOFScript. URL: http://eclipse.org/gmt/mofscript/

23.Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. *Interacting with Computers* 15(3), 2003, 289–308.

24. Santen, T. Security Engineering: Requirements Analysis, Specification, and Implementation. Habilitation, Fakultät Elektrotechnik und Informatik, Technische Universität Berlin, 2006.

25. Villanueva, P. G., Tesoriero, R. and Gallud, J. A. Distributing web components in a display ecosystem using Proxywork, in Love, L., Hone, K. MacEwan, T. (Eds.) *27th International BCS Human Computer Interaction Conference (BCS-HCI '13)* British Computer Society, (Swinton, UK, 2013). Article 28, 6.

26. Villanueva, P. G., Tesoriero, R. and Gallud, J. A. Proxywork: Distributing User Interface

Components of Web Applications, in Lozano, M. D., Mashat, A., S., Fardoun, H. M., Gallud, J. A., Penichet, V. M. R., Tesoriero, R. and Vanderdonckt, J. (Eds.) *3rd Workshop on Distributed User Interfaces: Models, Methods and Tools, DUI 2013. In conjunction with ACM EICS 2013 Conference*, (London, UK, 2013), 58-61.

27. Tran, V., Vanderdonckt, J., Tesoriero, R. and Beuvens, F. Systematic generation of abstract user interfaces, in *4th ACM SIGCHI symposium on Engineering interactive computing systems (EICS '12)*. ACM, (New York, NY, USA), 101-110.

28. Mellor, S. J., Scott, K., Uhl, A. and Weise, D. *MDA Distilled: Principles of Model-Driven Architecture*. Addison-Wesley. 2004.

29. Ambler, S. W. *Beyond Functional Requirements on Agile Projects*. URL: www.ddj.com/architect/210601918

30. Mostafa, D. Maturity Models in the Context of Integrating Agile Development Processes and User Centred Design. PhD Thesis, University of York. 2013.

31. Karatas, F., Bourimi, M., Kesdogan, D., Villanueva, P. G., Fardoun, H. M. Evaluating Usability and Privacy in Collaboration Settings with DUIs: Problem Analysis and Case Studies in Lozano, M. D., Gallud, J. A., Tesoriero, R. Penichet, V. M.R. (Edrs.) *Distributed User Interfaces: Usability and Collaboration*. Springer London. Human–Computer Interaction Series. ISBN 978-1-4471-5498-3. (2013) 119-127.

Organizers

King Abdulaziz University

Belgian Lab. of
Computer-Human
Interaction

Université catholique
de Louvain UCL

Sponsors

4th Workshop on Distributed User Interfaces and Multimodal Interaction 2014

DUI 2014 ,1st July 2014, Toulouse, France

 ACM ISBN: 978-1-60558-724-0