"Distributed User Interfaces: Models, Methods and Tools"

Lozano, María Dolores ; Mashat, Abdulfattah S. ; Fardoun, Habib ; Gallud, José ; Tesoriero, Ricardo ; Vanderdonckt, Jean

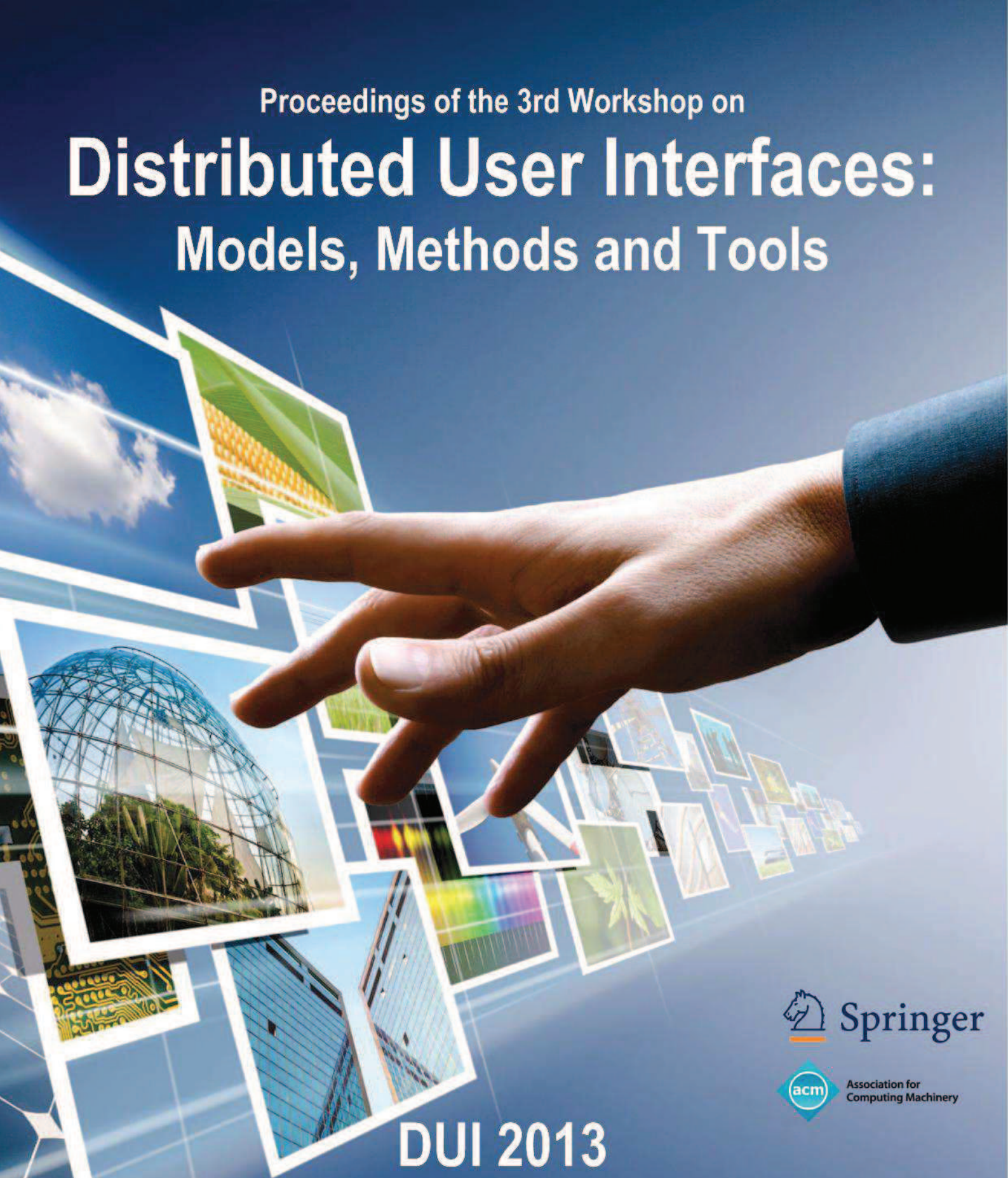
Abstract

Distributed User Interfaces (DUIs) have recently become a new field of research and development in Human- Computer Interaction (HCI). The DUIs have brought about drastic changes affecting the way interactive systems are conceived. DUIs have gone beyond the fact that user interfaces are controlled by a single end user on the same computing platform in the same environment. This new interaction mechanism affects the way these novel systems are designed and developed. New features need to be taken into account from the very beginning of the development process and new models, methods, and tools need to be considered for the correct development of interactive systems based on Distributed User Interfaces. Therefore, the goal of this workshop is to promote the discussion about the emerging topic of DUIs, answering a set of key questions regarding their development: How current UI models can be used or extended to cover the new features of DUIs? What new features should be considered and ho...

Document type : *Monographie (Book)*

Proceedings of the 3rd Workshop on

Distributed User Interfaces: Models, Methods and Tools



Springer

Association for Computing Machinery

DUI 2013

In conjunction with ACM EICS 2013 Conference

London, UK
June 24th, 2013

María D. Lozano - Abdulfattah S. Mashat - Habib M. Fardoun - José A. Gallud
Víctor M. R. Penichet - Ricardo Tesoriero - Jean Vanderdonckt (Editors)

Proceedings of the 3rd Workshop on

Distributed User Interfaces:
Models, Methods and Tools

DUI 2013

In conjunction with ACM EICS 2013 Conference

London, UK
June 24th, 2013

María D. Lozano - Abdulfattah S. Mashat - Habib M. Fardoun - José A. Gallud
Víctor M. R. Penichet - Ricardo Tesoriero - Jean Vanderdonckt (Editors)

Editors

Prof. Dr. María D. Lozano
University of Castilla-La Mancha
Computing Systems Department
Edificio Infante Don Juan Manuel
Campus Universitario
02071, Albacete, Spain
maria.lozano@uclm.es

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

Prof. Dr. Víctor M. R. Penichet
University of Castilla-La Mancha
Computing Systems Department
Edificio Infante Don Juan Manuel
Campus Universitario
02071, Albacete, Spain
victor.penichet@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. Abdulfattah S. Mashat
King Abdulaziz University
Information Systems Department
Faculty of Computing and Information Technology,
Jeddah, Saudi Arabia
asmashat@kau.edu.sa

Prof. Dr. José A. Gallud
University of Castilla-La Mancha
Computing Systems Department
Edificio Infante Don Juan Manuel
Campus Universitario
02071, Albacete, Spain
jose.gallud@uclm.es

Prof. Dr. Ricardo Tesoriero
University of Castilla-La Mancha
Computing Systems Department
Edificio Infante Don Juan Manuel
Campus Universitario
02071, Albacete, Spain
ricardo.tesoriero@uclm.es

3rd Workshop on Distributed User Interfaces 2013: Models, Methods and Tools
London, UK - June 24th, 2013
Cover designed by Gabriel Sebastián (Photo: Shutterstock)

ISBN-10: 84-616-4792-0
ISBN-13: 978-84-616-4792-7

Acknowledgements

This publication has been funded with support from the King Abdulaziz University. And special thanks for ***His Excellency Prof. Osama SadikTayeb President of King Abdulaziz University***, for His support in all areas to ensure the success of raising the level of scientific research, including this workshop. Also, researchers from ISE Research Group of the University of Castilla-La Mancha, and researchers from the Belgium Lab of Computer-Human Interaction of the Université Catholique de Louvain, supported the workshop.

DUI 2012: Models, Methods and Tools

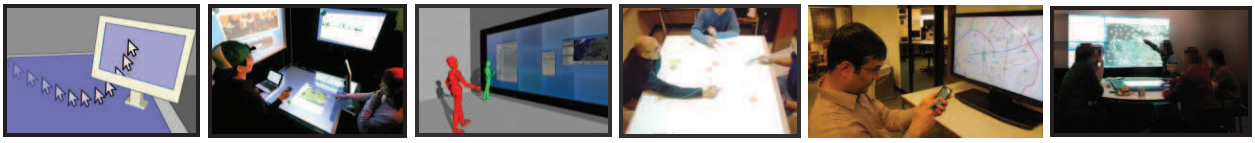
Distributed User Interfaces (DUIs) have recently become a new field of research and development in Human-Computer Interaction (HCI). The DUIs have brought about drastic changes affecting the way interactive systems are conceived. DUIs have gone beyond the fact that user interfaces are controlled by a single end user on the same computing platform in the same environment. This new interaction mechanism affects the way these novel systems are designed and developed. New features need to be taken into account from the very beginning of the development process and new models, methods, and tools need to be considered for the correct development of interactive systems based on Distributed User Interfaces. Therefore, the goal of this workshop is to promote the discussion about the emerging topic of DUIs, answering a set of key questions regarding their development: How current UI models can be used or extended to cover the new features of DUIs? What new features should be considered and how should they be included within the development process? What new methods and methodologies do we need to develop DUIs in a correct way following the quality standards for interactive systems?.

These articles were submitted to the *3nd Workshop on Distributed User Interfaces: Models, Methods and Tools* that was held on June 24th, 2013 in London, UK as part of the fifth ACM SIGCHI Symposium on Engineering Interactive Computing Systems EICS 2013 conference.

Table of Contents

The Broken Mirror: Space, Attention and Collaboration in Co-located Distributed User Interfaces

Miguel A. Nacenta
SACHI, School of Computer Science
University of St Andrews
St Andrews, Scotland, United Kingdom
mans@st-andrews.ac.uk



ABSTRACT
The promise of detaching applications from the constraints of a single computer or a single display has been a topic of interest to many groups in the last three decades. I have been working in the area of co-located, multi-display user interfaces for almost ten years, with a focus on perception and new interaction techniques. In this talk I will discuss my experiences building, designing, and studying MDEs. My intention is to start a discussion on the aspects of user interface prototypes that need to be taken into account when creating tools and models to implement them, and how some of these concepts apply as well to distributed scenarios.

Author Keywords
Co-located collaborative systems, Multi-display Environments, Distributed User Interfaces.

ACM Classification Keywords
H.5.2. Information interfaces and presentation: User Interfaces – Graphical user interfaces.

DESCRIPTIONS
Distributed User Interfaces (DUIs) can be *distributed* in at least three senses: across different physical locations, across different computers, and across different displays. In my previous research I have focused on interfaces that are distributed across displays. Due to the heterogeneity of computing environments, this usually also implies building systems that are distributed across different computers.

The use of DUIs for co-located work and play is promising because different devices can be good for different purposes. Using a multitude of heterogeneous displays/devices together has the potential to create cross-device interfaces that are more useful than the sum of their individual components. Unfortunately current multi-display systems are still a "broken mirror" in the sense that they feel like a random combination of flat pieces that do not reflect a consistent, easy to use interface.

My focus on the perceptual aspects of multi-display environments has led me to study a number of perceptual issues derived from the fragmented and heterogeneous nature of these multi-device and multi-display environments. In this talk I will discuss previous experimental and system building work that addresses some of the spatio-perceptual and social aspects of this "broken mirror" problem.

I will discuss problems related to the following topics:

- Input: how best to transfer objects from one display to another [8,2,5,3] and how input is affected by very large displays [1]
- Visuals: how the fragmentation of multi-display environments [9] causes the problem of split attention [10, 11]
- Socials: how the separation between personal and social spaces affects group interaction [6, 7, 4].

REFERENCES
1. Ricardo Jota, Miguel A. Nacenta, Joaquim A. Jorge, Sheelagh Carpendale, and Saul Greenberg. 2010. A comparison of ray pointing techniques for very large displays. In Proceedings of Graphics Interface 2010 (GI '10). Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 269-276.
2. Miguel A. Nacenta, Dzmitry Aliakseyeu, Sriram Subramanian, and Carl Gutwin. 2005. A comparison of techniques for multi-display reaching. In Proceedings of the SIGCHI Conference on Human Factors in Computing

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

Systems (CHI '05). ACM, New York, NY, USA, 371-380. DOI=10.1145/1054972.1055024

3. Miguel A. Nacenta, Carl Gutwin, Dzmitry Aliakseyeu, and Sriram Subramanian. There and Back Again: Cross-Display Object Movement in Multi-Display Environments. Human-Computer Interaction 24, 1 (2009), 170–229.

4. Miguel A. Nacenta, Mikkel R. Jakobsen, Remy Dautriche, Uta Hinrichs, Marian Dörk, Jonathan Haber and Sheelagh Carpendale. The LunchTable: a multi-user, multi-display system for information sharing in casual group interactions. Proceedings of the 2012 International Symposium on Pervasive Displays, ACM (2012), 18:1–18:6.

5. Miguel A. Nacenta, Regan L. Mandryk, and Carl Gutwin. 2008. Targeting across displayless space. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08). ACM, New York, NY, USA, 777-786. DOI=10.1145/1357054.1357178

6. Miguel A. Nacenta, David Pinelle, Carl Gutwin, and Regan Mandryk, Individual and Group Support in Tabletop Interaction Techniques. In Tabletops-Horizontal Interactive Displays. 2010, 303–333.

7. Miguel A. Nacenta, David Pinelle, Dane Stuckel, and Carl Gutwin. 2007. The effects of interaction technique on coordination in tabletop groupware. In Proceedings of Graphics Interface 2007 (GI '07). ACM, New York, NY, USA, 191-198. DOI=10.1145/1268517.1268550

8. Miguel A. Nacenta, Samer Sallam, Bernard Champoux, Sriram Subramanian, and Carl Gutwin. 2006. Perspective cursor: perspective-based interaction for multi-display environments. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06), Rebecca Grinter, Thomas Rodden, Paul Aoki, Ed Cutrell, Robin Jeffries, and Gary Olson (Eds.). ACM, New York, NY, USA, 289-298. DOI=10.1145/1124772.1124817

9. Miguel A. Nacenta, Satoshi Sakurai, Tokuo Yamaguchi, Yohei Miki, Yuichi Itoh, Yoshifumi Kitamura, Sriram Subramanian, and Carl Gutwin. 2007. E-conic: a perspective-aware interface for multi-display environments. In Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07). ACM, New York, NY, USA, 279-288. DOI=10.1145/1294211.1294260

10. Umar Rashid, Miguel A. Nacenta, and Aaron Quigley. 2012. The cost of display switching: a comparison of mobile, large display and hybrid UI configurations. In Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12), Genny Tortora, Stefano Levialdi, and Maurizio Tucci (Eds.). ACM, New York, NY, USA, 99-106. DOI=10.1145/2254556.2254577

11. Umar Rashid, Miguel A. Nacenta, and Aaron Quigley. 2012. Factors influencing visual attention switch in multi-display user interfaces: a survey. In Proceedings of the 2012 International Symposium on Pervasive Displays (PerDis '12). ACM, New York, NY, USA, , Article 1 , 6 pages. DOI=10.1145/2307798.2307799

BIOGRAPHY

Miguel Nacenta is a Lecturer at the University of St Andrews, Scotland, United Kingdom. He co-founded SACHI, the HCI research group at the School of Computer Science. He is a Telecommunication Engineer (Ingeniero Superior de Telecomunicación) by the Polytechnical University of Madrid (U.P.M.), Spain, and a doctor in Human-Computer Interaction by the University of Saskatchewan, Canada. His research focuses on perception applied to human-computer interaction, CSCW, and novel information visualization techniques. His work has been featured in New Scientist, FastCo.Design, Wired.co.uk, MIT Technology review and many others. This year Miguel is program co-chair of the Interactive Tabletops and Surfaces 2013 conference and local chair of UIST 2013.

Tangible and Distributed User Interfaces to Improve Cognitive Abilities of People Affected by Alzheimer's Disease

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

María Dolores Lozano
Computer Systems Department
University of Castilla-La Mancha, Albacete, Spain
maria.lozano@uclm.es

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

ABSTRACT

The world's population is becoming older. As a consequence, certain diseases as Alzheimer are gaining special importance in our society. We can take advantage of the evolution of new technologies to develop applications with the aim of enhancing and stimulating the cognitive abilities of people suffering from Alzheimer's disease in order to slow down the progress of the disease. Co-Brain Training is a collaborative and interactive game based on distributed and tangibles user interfaces that provides an intuitive and simple user interaction designed and developed with NFC and mobile technologies.

Author Keywords

Tangible interaction; NFC technology; Distributed User Interfaces; Collaboration; Alzheimer disease

ACM Classification Keywords

H5.2. Information interfaces and presentation: User Interfaces. – Graphical user interfaces

General Terms

Design, Human Factors, Experimentation

INTRODUCTION

In recent years life expectancy has increased thanks to the great advances in the field of medicine. Although physical body's health has improved, brain deterioration is still a challenge for researchers and clinicians. Neurodegenerative diseases such as Alzheimer's disease (AD) are the most common form of dementia among older people. According to the World Health Organization [4], it is estimated that in 2005, 0.379% of the world's population had dementia, and that the prevalence would increase to 0.441% in 2015 and 0.556% in 2030.

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

Alzheimer's disease is characterized by progressive loss of memory and other mental abilities; this loss results in atrophy of the affected regions. There are different solutions to improve and slow down the disease, one of these is cognitive stimulation therapies.

Technology is a tool that provides multiple opportunities for people when conducting psychosocial therapies based on cognitive stimulation [1]. Specifically the games are a way for users to perform tasks and learn in a fun, entertaining, and motivating way and help them to develop basic abilities such as attention, memory, language and executive functions.

Video games and serious games [2] simulate real situations for people with cognitive disabilities, such as shopping in the supermarket and include training programs to improve cognitive abilities. The advantages offered by these systems are numerous. They enhance positive attitudes in users while being appealing and encouraging, while providing information quickly. However, the user needs a minimum knowledge of computer use. Not everybody can use a computer and some devices, like the mouse or the keyboard, as they are not very intuitive for older people, they may need someone to help them.

In order to overcome this barrier between older people and technology, we have developed Co-Brain Training system, based on cognitive stimulation games. The system is easy to use, intuitive and accessible thanks to new technologies such as NFC (Near Field Communication). Our proposal also includes the use of Tangible User Interfaces (TUI) within a Distributed User Interface (DUI) setting. TUI refers to user interfaces which provide physical support to digital information, thus making the objects directly malleable and perceptible [3]. In the following sections, we present the system and its main features.

CO-BRAIN TRAINING SYSTEM

Co-Brain Training (Collaborative Brain Training) is an interactive and collaborative game designed to stimulate cognitive abilities such as memory, attention and language skills in people with Alzheimer's disease. The collaborative game is based on the distribution of interfaces

and devices to facilitate the user interaction. It integrates a new form of human-computer interaction which consists in approaching the mobile device to a tangible interface. The functionality of the system is as follows. The main game interface is executed in a computer and is projected on the wall, thus improving collaboration among users since the game interface can be displayed more easily and everybody can see it from any point. The users with the mobile device that incorporates the NFC reader can interact with the game through the tangible user interfaces which they have to manipulate. The interactive game is composed of different tangible interfaces representing the objects to manipulate in the game. The size of each tangible interface is 420x297 mm, the same as a din-A3 paper. It shows the images needed to handle the game and each image has a NFC tag incorporated that describes its functions. All the devices are connected through a Wi-Fi access point to the server where the methods associated with the games are implemented (See Figure 1).

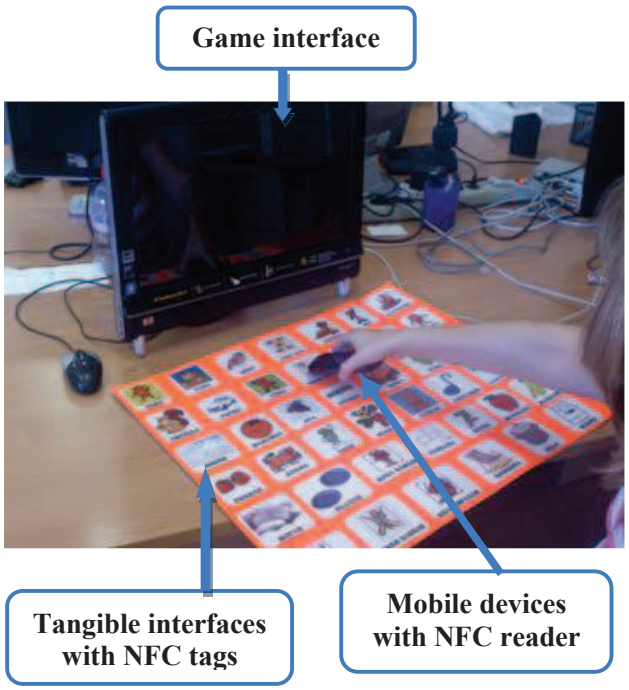


Figure 1. Co-Brain Training system composed of the Game interface, Tangible User Interface based on NFC tags and Mobile devices with NFC reader incorporated

Tangible and Distributed Interfaces

This project supports distributed user interfaces for flexibility and easy use. The two types of interfaces supported by the system are described as follows.

- Main Game User Interface. It is the main interface of the system. It displays the information graphically, with animations, text and sounds. At all times, it shows the game process and course.

-Tangible User Interfaces. These are common physical objects used as interaction resources to interact with the game. Each image contains an NFC tag incorporated (See Figure 2b) that indicates the function corresponding to the image. The interaction with the system is easy and intuitive as it is only necessary that the user brings the mobile device closer to the corresponding image in the tangible user interface.

The mobile device is used as an interaction device. It is responsible for reading only the information contained in the NFC tag and communicates it to the server and the game interface.

(a)

(b)

Figure 2: (a) Interaction Style. The user approaches the mobile device to the tangible interface. (b) NFC tag integrated within each image of the Tangible interface.

Co-Brain Training Tools

The system consists of two tools: Co-Brain Training Admin and Co-Brain Training Games.

Co-Brain Training Admin

Therapists, psychologists, teachers or parents are in charge of saving a game and checking the evolution of patients with Alzheimer's disease. This module of the system is focused on providing support for the person responsible for users, who can check their improvements. In addition, statistics and charts regarding the evolution of the users are provided. This part is a web application and may be displayed on any kind of device.

Co-Brain Training Games

This is the module of the system used for end users, i.e. patients with Alzheimer's disease:

• VeoVeo Game

VeoVeo is one of the games that has been designed to improve memory skills. This game consists in identifying objects that have been previously displayed on the game interface, as depicted in Figure 3a.

(a)

(b)

Figure 3: (a) Main User Interface and (b) Tangible user interface corresponding to the VeoVeo Game

The procedure is as follows. An image is displayed on the projector for a limited time and users have to concentrate and memorize the set of objects appearing in the image, depending on the game level. After thirteen seconds, the image disappears and users have to remember these objects. Then, they have to identify them among all the images shown as tangible user interfaces and approach the mobile device to the correct image, then, the NFC tag of that image is identified and the system checks if it is the correct one or not and then a success or a failure message is displayed on the projector, accordingly.

• Spelling Game

The aim of this game is to improve linguistic and vocabulary skills. First, a word is shown to the users and then, they have to identify the letters that make up this word. They have to choose the letters in the correct order and approach the mobile device to the letters represented as tangible user interfaces (See Figure 4).

(a)

(b)

Figure 4: (a) Main User Interface and (b) Tangible User Interface corresponding to the Spelling Game

System Architecture

Co-Brain Training is a client-server system, designed as follows. The client system is the user mobile device with NFC reader. It is connected to the server application through a wireless network and communicates with the tangible interfaces via NFC when the user approaches the mobile device to the tangible interface. A tag is integrated inside each image of the tangible interface; each tag describes a function. When the NFC reader in the mobile device is brought closer to the chosen representative image, the NFC tag is excited by electromagnetic waves sent by the NFC reader and then, the mobile device executes the corresponding method in the server. It maps this information in the database and executes the steps necessary to return the information to the main game interface (See Figure 5).

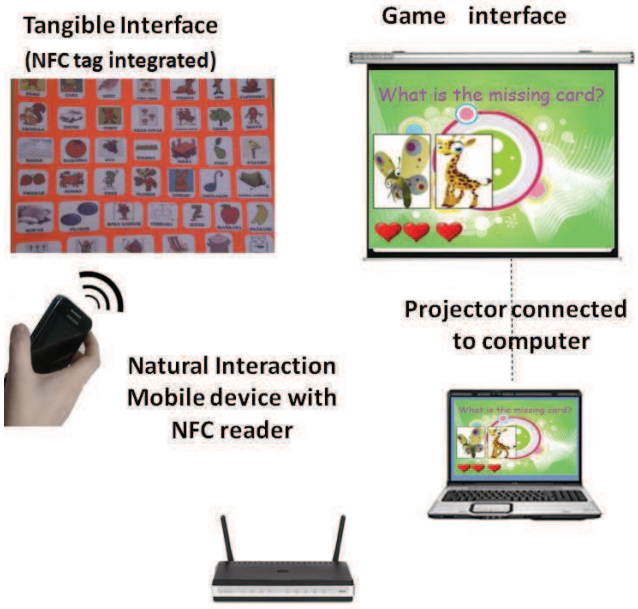


Figure 5:System architecture

CONCLUSIONS

Co-Brain Training (Collaborative Brain Training) is an interactive and collaborative game based on distributed and tangibles user interfaces developed with emerging technologies such as NFC and mobile devices. The main objective is to stimulate and improve cognitive abilities of people with Alzheimer's disease. In order to interact with the system it is necessary to approach the mobile device (that incorporates NFC reader) to the tangible interface that integrates NFC tags inside, and then the results and associated information are projected on the wall. This interaction style is simple and intuitive; its purpose is to eliminate the technological barrier for older people.

ACKNOWLEDGMENTS

This research has been partially supported by the Spanish research project TIN2011-27767-C02-01 and the regional projects with reference PII10-0300-4174and PII2C09-0185-1030. We would especially like to thank Francisco Vizcaino Garcia for his collaboration in this project.

REFERENCES

1. Annema,J.H., Verstraete,M. , Vanden Abeele,V., Desmet, S., Geerts, D. Video games in therapy: a therapist's perspective. International Journal of Arts and Technology, 2011
2. Bouchard, Bruno and Imbeault, Frederick and Bouzouane, Abdenour and Menelas, Bob-AntoineJ. Developing Serious Games Specifically Adapted to People Suffering from Alzheimer. Serious Games Development and Applications Book, ISBN:978-3-642-33686-7, 2012
3. Ishii ,H. Ullmer ,B. Tangible bits: towards seamless interfaces between people, bits and atoms, Proceedings of the SIGCHI conference on Human factors in computing systems, p.234-241, March 22-27, 1997, Atlanta, Georgia, United States [doi>10.1145/258549.258715]
4. World Health Organization (WHO). World report on disability; Enero2013.

Information Concepts for Cross-device Applications

Michael Nebeling, Christoph Zimmerli, Maria Husmann,
David Simmen and Moira C. Norrie
Institute of Information Systems, ETH Zurich
CH-8092 Zurich, Switzerland
{nebeling|zimmerli|husmann|simmen|norrie}@inf.ethz.ch

ABSTRACT

Cross-device application development poses different requirements ranging from the adaptation and distribution of user interfaces, over the migration of application state to the management of data as it is moved around and shared between devices. In particular, the evaluation of multi-device user interfaces becomes a challenge when they are distributed across a range of interaction resources. We present our ongoing work towards a platform for the design and evaluation of cross-device applications that promotes the central concept of a *cross-device session* useful for both design and evaluation. Our concept enables the tracking and management of the *data* part of a distributed, multi-device application as it is viewed and updated on different devices as well as the *metadata* describing the interactions and how the data evolved as a result of them. We illustrate the benefits of our approach for a first cross-device application and discuss how we plan to address the remaining challenges.

Keywords

Multi-user/multi-device interaction; cross-device development platform; session concept.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Input devices and strategies, Interaction styles*

1. INTRODUCTION

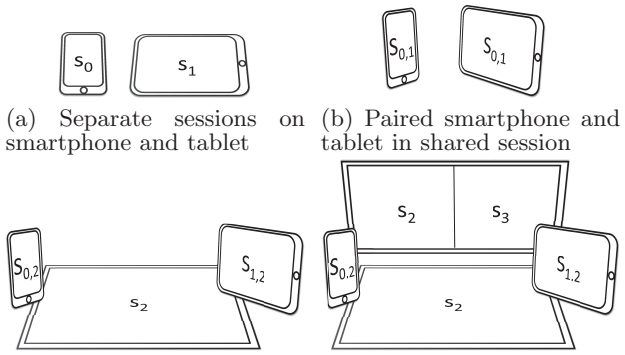
It has become increasingly common that users perform their tasks using various devices, ranging from traditional desktop computers to new types of mobile devices with different multi-modal interaction resources. Yet, most applications today still assume a single device for carrying out the tasks. Recent research has started to focus on settings around multiple devices, developing new forms of interaction in both individual and collaborative usage scenarios [1, 3]. However, this requires new design guidelines and techniques for sharing information or even devices, which is generally not supported by current applications.

Figure 1: Multi-user, multi-device scenario around a tabletop—we present information concepts for the design and evaluation of cross-device applications

Cross-device development requires design decisions and implementation effort on different layers, involving the adaptation, distribution and migration of both user interfaces and data across devices. Several frameworks have been proposed to support multi-device development based on different model-based [6], object-oriented [3] and data-oriented approaches [1]. The common goal to reduce design effort is often anticipated by using different abstraction layers and models to describe the user interface and interactions [9]. However, in our own experience and based on a systematic literature review, we have identified three major problems:

Need for Rapid Prototyping Tools. Existing solutions provide little support for rapid prototyping of multi-device, distributed user interfaces. Rather, the focus is on powerful and expressive models to support systematic multi-device development. However, the increased flexibility of cross-device applications in terms of the use context often requires experimentation with alternative designs. Current GUI builders support the design of user interfaces for a single device, but provide no specific support for user interfaces distributed over multiple devices. There are partial solutions that focus on either adapting interfaces for different devices [4] *or* distributing them across devices [5]. The exceptions are model-based, multi-device authoring environments which however start from an abstract representation rather than concrete interface more common to designers [9].

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



(a) Separate sessions on smartphone and tablet

(b) Paired smartphone and tablet in shared session

(c) Mobile devices paired with tabletop in shared session

(d) Mobile devices paired with tabletop and wall-size display in parallel session

Figure 2: Different scenarios and session modes

Different Software Architectures and Implementation Methods. There is an increased proliferation of new devices with very different hardware and software characteristics. Developers hence require a wide range of skill sets and programming experience with different languages and software development kits. Many solutions have been built for a fixed environment to support specific interaction techniques [2]. More dynamic support for handling additional devices often requires applications to be built using special middleware and programming models [1, 3]. An exception is the service-oriented architecture in [10] that is however constrained to message-based applications. Web-based approaches to overcome device differences are promising, but often constrained by current browser support [7].

Limited Support for User Studies. Finally, while recent research tends to focus on exploring new cross-device interaction techniques, the current state is to build special instruments for each study [2, 11]. There is insufficient tool support for user studies spanning multiple devices in co-located, remote or mixed usage scenarios.

In this paper, we present our ongoing work towards a cross-device development platform supporting pre-configured physical spaces as well as ad-hoc, dynamic "walk-up and share" mobile scenarios. The goal is to provide flexible support for interactive development on the target devices themselves and dynamic distribution with additional devices. In a first step, this paper presents a novel concept for session management that we have started to exploit for enabling both the design and evaluation of cross-device applications. Our concept facilitates different modes for sharing and managing interfaces as well as data across multiple devices monitoring the data evolution in a central history. We will sketch how it can also be configured for user testing in mixed co-located and asynchronous, multi-device scenarios, and how we plan to support studies based on session playback and analysis.

2. SCENARIOS

There are many possible scenarios in which multiple devices could be used to achieve a task—individually or in collaboration, with one or multiple devices per user [9]. We use the setting in Figure 1 to motivate our session concept.

Figure 2a illustrates a first scenario in which the smartphone and tablet are used individually with no interaction across devices. Here, each device is configured with its own session, S_0 and S_1, manipulating data in isolation.

Figure 2b illustrates a second scenario in which the two mobile devices are paired to participate in a shared Session $S_{0,1}$. As a result, the interactions carried out with one device are no longer independent of the other. The session can be configured so that the interface is distributed across the devices. In this scenario, it is possible that users manipulate their own data or the same data part of the shared session.

Based on this scenario, Figure 2c illustrates another one in which both mobile devices are instead connected to the tabletop and therefore indirectly part of the same Session S_2. Here, the mobile devices effectively function as a remote control and interactions on either device are reflected on the tabletop. However, direct manipulations on the tabletop are not synchronised with Sessions S_0 and S_1. This could be enabled by pairing the tabletop as in the previous scenarios.

Figure 2d extends on the previous scenario in that all manipulations of Session S_2—be it through interactions on the mobile devices or the tabletop itself—are now also reflected on the wall-size display paired with the tabletop. The wall display is also running a separate Session S_3 in parallel.

Likewise, the person sitting with the laptop on the left in Figure 1 may follow the actions performed on the table from her device by configuring Session S_2, and could also do this remotely.

To facilitate user studies, each session could be configured to record the input and performed actions on each device. Additionally, devices such as Kinect for 3D skeletal tracking could be configured as part of Session S_2 and record the user interactions around the tabletop in physical space.

3. MODEL AND ARCHITECTURE

The goal of our platform is to enable the design, distribution and evaluation of multi-device applications—both at the user interface and data level—in a uniform way based on the session concept. Below we present a simple model of the concepts and how they relate to each other. We also present the features of our platform and the architecture of applications. This is followed by an example of how an existing single-device application could be extended for multi-device scenarios based on the concepts.

3.1 Model

As illustrated in Figure 3, our concept of a Session S is defined as $S =< U, D, I >$ linking the concepts of User U, Device D and Information I represented as data and metadata. Data refers to the information managed in a session, while metadata describes the interactions and how the data was manipulated during the session.

Figure 3 depicts the core model. Sessions are created and maintained by one or multiple users for one or multiple devices. A `Session` is by default public and accessible to everyone. The platform supports a `PrivateSession` where the initiating user can directly control which other users get access to it. All interactions with the data of an application are made through sessions. For each interaction, our platform

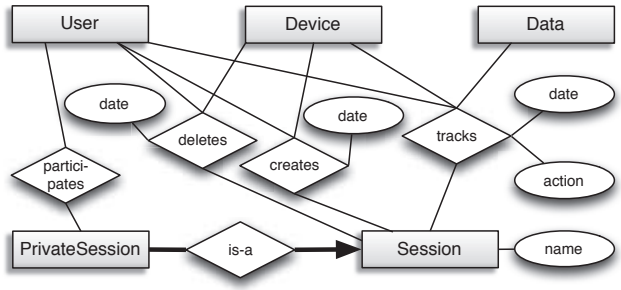


Figure 3: Core model linking the different concepts behind our platform

Action	Description
add	adds data to this session
edit	change the data in this session
copy	add the data to another session
delete	delete the data in this session
move	copy the data and delete it in this session
log	write arbitrary data to the session log

Table 1: Session API

tracks the `User`, `Device` and `Data` with `date` and `action`. The possible actions are summarised in Table 1. The type of data is application-specific and the actual data is not manipulated by our platform. However, the platform enforces a unique identifier for each data instance, while edit actions create new versions of the data. Since the id of a resource stays from creation on over all edit actions and multiple sessions, we are able to trace the evolution of all data in the application over time. Sessions can be managed through the Session API. Note that, in order to keep up the traceability of all actions (e.g. when the session is configured for user studies), the session and its data will never actually be deleted, but only marked as deleted if requested.

3.2 Features

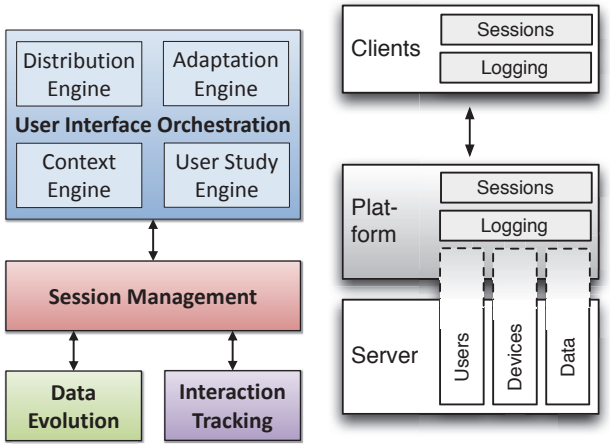
Based on our session concept, our platform provides the following features for cross-device design and evaluation.

Simple User and Device Registration. Our platform enables simple registration of users and devices. For ad-hoc scenarios, automatic device detection is supported based on a device description repository.

Each session involves at least one device, but multiple devices can participate in the same session. As illustrated in Figure 2d, it is also possible to control multiple sessions from a single device, allowing interactions between sessions.

Implicit and Explicit Session Management. By default, session management is implicit, e.g. each device runs its own session. The platform supports explicit session management, e.g. to allow certain users and devices to view and participate in other sessions. This also supports asynchronous and remote collaboration.

Evolution Tracking and History. The metadata collected as a result of the interactions during a session can be used for tracking the data evolution across multiple users and devices. The history is useful for version management and migration of data between both devices and users.



(a) Platform components (b) Extended architecture

Figure 4: Platform and application architecture

Custom Logging and Session Playback. Our platform supports custom logging of actions during a session to track additional events not directly triggered by the application. This can for example be useful when additional devices such as Kinect are used for tracking additional information like the user interaction in physical space.

The platform also supports session playback. This feature was primarily developed to migrate the data between users and devices and for asynchronous and remote scenarios. However, it is also useful for user studies allowing evaluators, not only to trace the data and how it evolved, but also to view the interactions across multiple devices.

Interaction Tracking and Analysis. In related projects, we have started to develop novel interaction tracking and analysis tools that support multi-device development in particular for mobile touch devices [7, 8]. We are currently working on integrating the techniques with our platform.

3.3 Architecture

To support multi-device design and evaluation, our platform consists of the four major components illustrated in Figure 4a. First, **user interface orchestration** is enabled based on a **context engine** for keeping track of users and devices, an **adaptation engine** for adapting interfaces for different form factors and input modalities, a **distribution engine** for rendering (parts of) an interface on different devices and a **user study engine** for conducting user evaluations in multi-device scenarios. Second, the **data evolution** component provides basic functionality for monitoring the application state and keeping a history of the data. Third, the **interaction tracking** component offers techniques for monitoring user behaviour. Finally, the **session management** component at the core of the platform is used to feed the different components with the use context information.

The platform acts as a middleware to track and maintain the data on different devices independent of the application. Figure 4b illustrates the extension of an existing application with our platform. Assumed is a client/server application managing some kind of data. Linking the application to

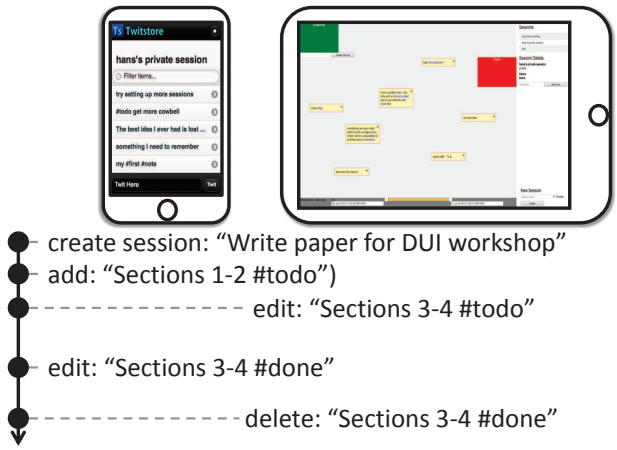


Figure 5: Extended note-taking application with session-based management of notes between users and devices

our platform requires a mapping from the concepts of user, device and data in the platform to available counterparts in the application. If some concept is not present in the application, it is managed solely by the platform. In order to fully profit from the advanced features and logging facilities offered by the platform, clients need to use the Session API.

3.4 Implementation

Our platform is currently implemented based on web technologies using jQuery in combination with jQMultiTouch [7] on the client side and a Java server with an object database backend. The Session API is exposed via a REST interface supporting asynchronous communication using AJAX. MobileESP[1] is used for automatic device detection.

4. FIRST APPLICATION

As a first proof-of-concept, we extended an existing note-taking application developed in another project with the concepts presented in the previous sections. The application already had the notion of users and their private notes with data being stored on a server and two different web-based clients. One client is targeted at mobile devices with smaller screens and shows the notes in list form. The other is intended for desktop and tabletop settings allowing users to organise and group notes spatially on the screen. Both clients and the server required minimal extensions enabling multi-device sessions as illustrated in Figure 5.

So far, we have used the extended application in group meetings and informally tested it with multiple users and different devices such as laptops, tablets, smartphones and a tabletop similar to the scenario in Figure 2c. We have observed how notes are often taken and managed in private sessions before and after meetings, typically while on the go. During the meetings themselves, selected notes were then mostly shared and discussed with others in a shared session on the table. The notes on the table were frequently refined, merged and sometimes also discarded independent of the private copies. Towards the end of meetings, the changes and notes created in the meeting were then copied over from the common to private sessions and merged as necessary. Our platform will enable more formal user studies and allow us to develop and test new multi-device features.

[1]http://blog.mobileesp.com

5. CONCLUSION AND FUTURE WORK

We have presented a cross-device session concept and our first prototype of a platform that supports the design, distribution and evaluation of multi-device applications—both at the user interface and data level. Our approach integrates the different concerns in contrast to existing works that tend to focus only on one of these aspects. For example, [9] only considers aspects related to the user interface, and [3] assumes a single information landscape displayed by all devices. Our first application allows users to manage information separately and arbitrarily connect devices for ad-hoc sharing. Implementing a first multi-device scenario allowed us to overcome many challenges, but several parts remain as future work. We are currently working on a cross-device GUI builder and tools enabling interactive development of new and simple adaptation of existing applications for multi-device environments. Particular attention is paid to the programming model, where we make sessions a first-class concept for static, dynamic and mixed distribution.

6. REFERENCES

[1] T. Gjerlufsen, C. N. Klokmose, J. Eagan, C. Pillias, and M. Beaudouin-Lafon. Shared Substance: Developing Flexible Multi-Surface Applications. In *Proc. CHI*, 2011.

[2] M. Haller, J. Leitner, T. Seifried, J. R. Wallace, S. D. Scott, C. Richter, P. Brandl, A. Gokcezade, and S. E. Hunter. The NICE discussion room: integrating paper and digital media to support co-located group meetings. In *Proc. CHI*, 2010.

[3] H.-C. Jetter, M. Zöllner, J. Gerken, and H. Reiterer. Design and Implementation of Post-WIMP Distributed User Interfaces with ZOIL. *IJHCI*, 28(11), 2012.

[4] J. Lin and J. A. Landay. Employing Patterns and Layers for Early-Stage Design and Prototyping of Cross-Device User Interfaces. In *Proc. CHI*, 2008.

[5] J. Melchior, D. Grolaux, J. Vanderdonckt, and P. V. Roy. A Toolkit for Peer-to-Peer Distributed User Interfaces: Concepts, Implementation, and Applications. In *Proc. EICS*, 2009.

[6] J. Melchior, J. Vanderdonckt, and P. V. Roy. A Model-Based Approach for Distributed User Interfaces. In *Proc. EICS*, 2011.

[7] M. Nebeling and M. C. Norrie. jQMultiTouch: Lightweight Toolkit and Development Framework for Multi-touch/Multi-device Web Interfaces. In *Proc. EICS*, 2012.

[8] M. Nebeling, M. Speicher, and M. C. Norrie. W3Touch: Metrics-based Web Page Adaptation for Touch. In *Proc. CHI*, 2013.

[9] F. Paternò and C. Santoro. A Logical Framework for Multi-Device User Interfaces. In *Proc. EICS*, 2012.

[10] J. S. Pierce and J. Nichols. An Infrastructure for Extending Applications' User Experiences Across Multiple Personal Devices. In *Proc. UIST*, 2008.

[11] T. Seyed, C. Burns, M. C. Sousa, F. Maurer, and A. Tang. Eliciting Usable Gestures for Multi-Display Environments. In *Proc. ITS*, 2012.

Making Distributed User Interfaces Interruption-Resistant: A Model-Based Approach

Félix Albertos, Víctor Penichet, José Gallud
Computer Systems Department
University of Castilla-La Mancha
Albacete, Spain
{victor.penichet,jose.gallud,felix.albertos}@uclm.es

Marco Winckler
ICS-IRIT team
Université Paul Sabatier
Toulouse, France
winckler@irit.fr

ABSTRACT
Distributed User Interfaces (DUIs) have gone beyond the fact that traditional user interfaces run on the same computing platform in the same environment. This new interaction paradigm affects the way these novel systems are designed and developed. New features need to be taken into account from the very beginning of the development process and new models and tools need to be considered for the correct development of interactive systems based on DUIs. The starting point of this paper is that DUI-based systems are susceptible of being interrupted in several ways as they are dependent on connectivity. In this proposal this issue is assessed from a conceptual point of view, asking the question of what new features should be considered and how should they be included within the development process? The model-based approach presented provides developers with means to make DUIs resilient or resistant to interruptions.

Author Keywords
Work interruption, caching modeling, model-based approach, distributed user interface.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Connectivity is one assumption that in the last years has been commonly assumed. But there are a lot of scenarios where this assumption may not be totally guarantee. Travelling by flight, natural disasters, areas where the connection is not available or limited, are only a small set of scenarios where the lack of connection has to be taken into account. When developing systems connected through the Internet, deal with disconnection problems is an open issue, and have to be included when modeling these systems. Some examples of systems susceptible of being interrupted are those based on the Distributed User Interfaces paradigm.

A DUI is a user interface whose components are distributed across one or more of the dimensions input, output, platform, space, and time [2]. The interface may be restricted to the same physical (and geographic) space, or can be distributed geographically. In the last scenario, interfaces may use the client-server paradigm to get connected. This paradigm is followed by the web-based distributed user interface [6], where Internet is used to connect the DUIs. Following the paradigm of the web, a connection to the server must exist in order to interact with the system. But, what happen if there is no connection to the server? In this scenario, an interruption due to the lost of connectivity to Internet arises this question: are users able to continue interacting with the system without being affected by the system interruptions? Our aim is to make DUIs resilient to interruptions, providing continuity of service for DUIs connected over the Internet, dealing with the problematic of interruptions, as shown in the Figure 1. The term resilient is often used to address systems that are able to recover from failures, but in the present context it is used to qualify systems that can prevent from the occurrence of interruptions, help users to resume from interrupted tasks, and/or ensure a minimum level of service for performing a task despite of the interruption [8] in DUIs environments.

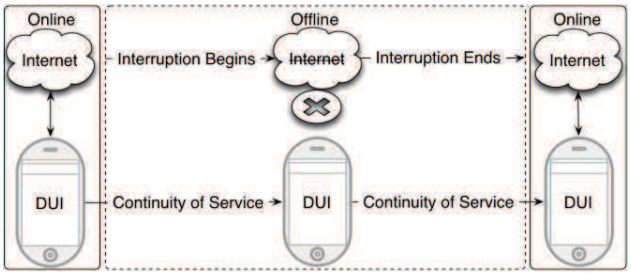


Figure 1: Providing Continuity of Service.

STATE OF ART
McFarlane [7] proposed a general, interdisciplinary, theory-based definition of human interruption, which says that human interruption is "the process of coordinating abrupt changes in people's activities." Interruptions occur when a

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

person is working on a primary task (usually long-lasting) and an alert for a secondary task occurs.

There are previous works to systematically detect and deal with interruptions when doing task on interactive systems [8]. They try to model and to predict the impact of interruptions on those systems.

There are some DUIs approaches susceptible of being interrupted due to the lack of connectivity. In [4] is presented DUIs to enhance decision making in disaster situations. One of its main goals is to be fault-tolerant and to take into consideration the requirements from all stakeholders. Therefore, a lot of devices and architectures must be supported, and the following of international standards may be welcomed to design the system that supports them, such us the HTML standards, widely adopted.

HTML5 is the latest version of the HTML standard. One of the new ideas present in the Web development on these days is a set of specifications known as "Web Applications" [9]. The goal of this specification is to enable improved client-side application development on the Web. Due to the variety of scenarios where Web Applications can be used, new ways are needed to support the development of applications. One of the aspects to take into account is the network connection availability or the interruptions while using Web Applications. In order to enable users to continue interacting with Web Applications and documents even when their network connection is unavailable authors can provide a manifest that lists the files needed for the Web Application to work offline and which causes the user's browser to keep a copy of the files for use Offline Web Applications [10]. Other set of technologies allows Web Applications to store locally information. Keeping a copy of the files used in Web Applications is not always enough to make them interruptions resilient. Webstorage introduces two related mechanisms for storing name-value pairs on the client side. However, it does not provide in-order retrieval of keys, efficient searching over values, or storage of duplicate values for a key.

Earlier works has identified the problem of disconnection in Web environments. They proposed models to deal with this issue, such us [1, 3, 5]. The main drawbacks on these works are that most of them don't use standard technologies, restricting the target platforms to use, and the difficulties or the impossibility to be adapted to existent Web Applications. Also, the main functionalities are focused in only catching the information locally, not to deal with the consequences of the interruption. Finally, most of them are hard to implement and involve the programmatically implementation of the provided features.

MAKING WEB APPLICATIONS RESILIENT TO INTERRUPTIONS

When designing Web Applications there are several model-based approaches, as aforementioned. But our proposal deals with an issue that has been neglected when modeling Web Applications: how the application behaves when it is interrupted and how to recover from interrupted work. The proposed conceptual model-based approach combines explicit representation of end-user navigation and local information storage. It provides users with information about which Web site's contents are available when they are interrupted and how they can get easy access to local cache content.

The proposed model represents the *static* properties of a Web site, as well as it *behavior*. The *static* properties define the structure of the Web site. The *behavior* defines how the system will react to the events and how it will change over the time. Web pages are the basic elements in the World Wide Web. They are also the basic elements in the offline model. They are defined as *nodes*. A *node* is an element in the model that may be connected to other *nodes*. They are connected to each other to conform what is known as Web projects. A *Web project* is a superset of *nodes* and it is the upper level of abstraction in the model. We can refer Web projects as Web sites.

The model shows both the *static* properties as well as the *behavior* of the Web site. To represent these properties, nodes have associated properties, represented as *node states*. We propose three state levels for *nodes*: *static*, *navigational* and *data*. *Static* state is defined according to the requirements of the site and the site structure. Static states are *normal*, *precacheable*, *nocacheable*, *initial* and *external*. *Navigational* state changes over the time when users use the Web site. *Navigational* states are *novisited* and *visited*. *Data* states are the result of the combination of the two previous states. Possible values for data states are *cached* and *nocached*. ***Static*** states are defined to answer the following questions. The first question is: is the node internal or external? External nodes are set to *external* state. This state excludes other static states for the node. The second question is: is the node initial? Initial node is set to *initial* state. It will be always accessible in offline mode and *precacheable*. *Initial* nodes are unique within the Web project. The third question is: will be the node cached when the Web site is visited? *Precacheable* nodes will be always cached when the Web site is visited, but *nocacheable* nodes will not be cached ever. ***Navigational*** state is set when users navigate through the site. It changes when the site is used. It is a dynamic state. Have been defined two states: *novisited* and *visited*. The question answered with this state is: has been the node visited? When a node has been visited, it is set to *visited*. Meanwhile, the node is set to *novisited*. ***Data*** state is the result of the combination of the two previous states. As a result, a node can be *cached* or *nocached*. When a node is *cached*, it will be available when the site is interrupted. When a node is *nocached*, it wouldn't be available when the site is interrupted.

The model introduces means to deal with interruptions due to offline navigation. One of the mechanisms used to

support offline navigation is the transformation of the elements of Web pages. These transformations may act removing or altering the content of Web pages. Available transformations are *element disabling* and *alternative link destinations*.

When using Web pages, some of the elements may not be available for users in offline mode. This restriction may be due to several reasons. One of the reasons could be that part of the Web page requires a connection with some external resource. Since the fact that there is no connection to the server, the element wouldn't work. An example of this scenario is when a form is used to send information to a remote server. Other scenario is when linking to an external resource. Since the site is in offline mode, the action couldn't be performed. Another reason for disabling an element is when it shows information retrieved from an external server. An example of this scenario is when using Web pages to show online maps or *Facebook* walls. To overcome these situations, the model allows *element disabling*. Through this technique, any element in the Web page could be disabled, belonging it to our server or to an external server, preventing it to be presented in the Web page when it is in offline mode. Since Web pages are described in HTML and most elements can be nested, when disabling an element, all the elements enclosed within this element will be disabled too. Another available transformation is the *alternative link destination*. When using the Web in offline mode, some destination will not be reachable due to lack of connectivity or for design constraints. To prevent the problems associated with the lack of connectivity and to support the design constraints, the model allows giving an alternative destination to any link in Web pages. As a result, when in offline mode, alternative links will work instead of the original.

INTERRUPTION-RESILIENT DUI APPROACH

In the Web Application's Offline Model the properties of the system define the behavior of Web Applications. The main elements are the Web pages that form the Web site and the HTML elements that describe those Web pages. They are independent of each other. Links are the only connection between Web pages. But within DUIs, there is an important issue to be addressed. When an interruption occurs on DUIs environment, it is not only affected the actual interface being displayed, a DUI, but also is affected the overall DUI system. For example, it is not the same a DUI that only show information than other DUI that is used to input important data within the Web Application. The first one may not be critical for the overall system; meanwhile the second may be vital for the overall system task. Therefore, new mechanisms have to be introduced to deal with this kind of scenarios.

To make DUIs resilient to interruptions we have included in the Web Applications Offline Model the DUI Offline Model. In this Model, DUI-based systems have two kinds of elements, *soft* and *strong*. A *soft element* in a DUI system is a non-critical element of the UI (called *soft DUI*). A *strong element* in a DUI system is an element that host critical information (input or output) and interruptions can affect the system (called *strong DUI*). Each type of DUI behaves depending of it connection status: online (non-interrupted) or offline (interrupted), and specific mechanisms are provided to deal with interruptions. These mechanisms, allow users to interact with interrupted DUIs and to synchronize offline work. Mainly these mechanisms provide caching features, content removal and transformation and change link destinations.

The first type of DUIs is the *Soft DUI*. These DUIs do not have a special behavior in the model. From the point of view of the overall system, when they are not connected is assumed that they have been disconnected from the system. From the point of view of these kinds of DUIs, they do not provide any special mechanism to deal with the interruption. These DUIs are represented in the model with dotted lines, as depicted in the Figure 2.

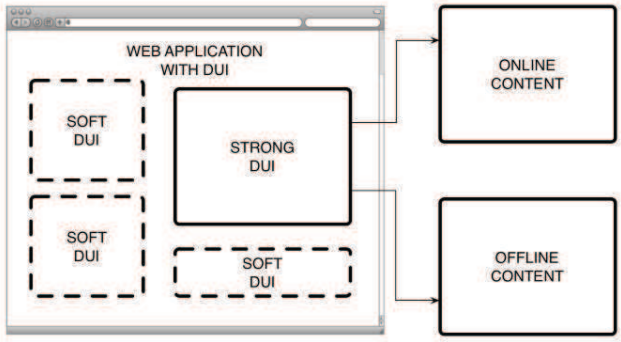


Figure 2: Content transformation on *strong DUIs*

The second type of DUIs is the *strong* DUI. These DUIs are provided with mechanism for offline operation when they are interrupted. They are represented in the model with solid lines, as depicted in the Figure 2. From the point of view of the overall system, the connection interruption with a *strong* DUI doesn't means that the DUI has been disconnected from the system. To disconnect the DUI there must be an explicit disconnection operation. As a consequence, within the status of this type of DUIs, the connection status must be represented in the model. When the *strong* DUI is created, it is represented within the system with two possible statuses: *online* or *offline*. *Online* indicates that the DUI has not been interrupted meanwhile the *offline* status shows that the DUI is interrupted, but has not been an explicit disconnection from the overall system. From the point of view of *strong* DUIs, they are always cached for offline operation. The mechanisms involved in this task have been described previously in the Offline Model for Web Applications. When an interruption arises, end users can use the DUI and the mechanisms for offline work have to be defined, as follows. During the interruption, the content is susceptible of been transformed for its operation, as depicted in the Figure 2. Available

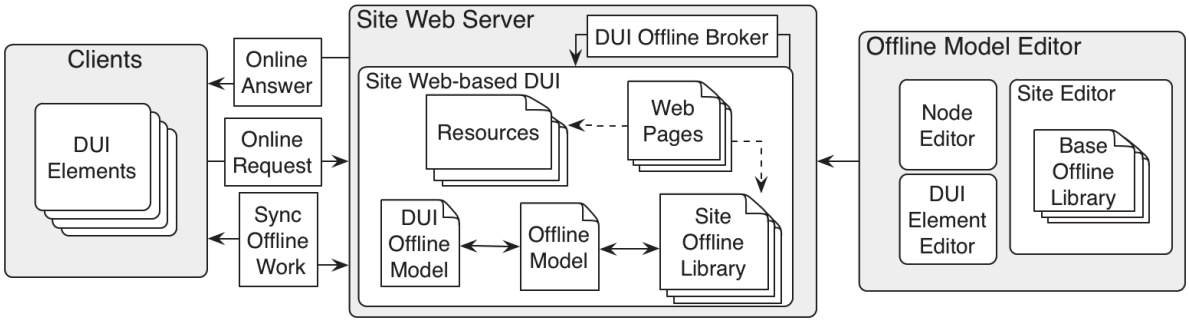


Figure 3: Overall architecture of the offline model approach for DUI Web Applications.

transformations are *element disabling* and *alternative link destinations*. These transformations have been described within the Offline Model for Web Applications.

The architecture of the offline model approach for DUI Web Applications is depicted in the Figure 3. The architecture includes the DUI Offline Model. It contains information about the DUIs elements, such us the type of DUIs that conform the system and the content transformation policies.

CONCLUSION AND FUTURE WORK

In this proposal, interruptions of DUIs in Web Applications are assessed since a modelling perspective, which constitutes an extension of the aforementioned Offline Model. The main goal is to provide with means to make DUIs resilient to interruptions. The provided conceptual mechanism includes the definition of two types of DUI elements, *soft* and *strong* for dealing with interruptions. Also, useful information is presented to describe the behavior of DUIs in Web Applications. As future work, we are working on the creation of a proxy server to annotate web applications on the fly. With this feature, the Offline Model will be included without modifying the original Web Application, simplifying the Offline Model management.

ACKNOWLEDGMENTS

This research has been partially supported by the CICYT TIN2011-27767-C02-01 project and the regional project with reference PPII10-0300-4174.

REFERENCES

1. Bret Cannon. 2011. Minimizing Resource Access and Management Disparities Between Desktop and Web Applications. PhD thesis, January 2011.
2. Elmqvist, N. 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, May 7--12, 2011.
3. E. Goncalves and A. M. Leitao, “Implementing offline work in web applications for rich domains,” in Proc. of the 11th Int’l Symposium on Web Systems Evolution (WSE). IEEE, 2009, pp. 79–82.
4. M. Heupel, M. Bourimi, D. Kesdogan, T. Barth, P. Schwarte and P. G. Villanueva. Enhancing security and usability of DUI based collaboration with proof based credential systems. Proceedings of the Distributed User Interfaces 2012 CHI Workshop, held in conjunction with 2012 CHI conference, 23-26, ISBN-10: 84-695-3318-5.
5. Yung-Wei Kao, ChiaFeng Lin, Kuei-An Yang, and Shyan-Ming Yuan. 2012. A Web-based, Offline-able, and Personalized Runtime Environment for executing applications on mobile devices. Comput. Stand. Interfaces 34, 1 (January 2012), 212-224. DOI=10.1016/j.csi.2011.08.006 http://dx.doi.org/10.1016/j.csi.2011.08.006.
6. C. Lee, F. A. Musyaffa, Y.-M. Kwon, "Collaborative Social Authoring Technology Using Web-based Distributed User Interface (DUI)," Int'l Conf. CHI 2012, Workshop on DUI (Distributed User Interfaces), May 2012.
7. McFarlane, D. C. (1997). Interruption of people in human-computer interaction: A general unifying definition of human interruption and taxonomy (NRL Formal Report NRL/FR/5510–97–9870): Naval Research Laboratory, Washington, DC.
8. Philippe Palanque, Marco Winckler, Jean-François Ladry, Maurice H. ter Beek, Giorgio Faconti, and Mieke Massink. 2009. A formal approach supporting the comparative predictive assessment of the interruption-tolerance of interactive systems. In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems (EICS '09). ACM, New York, NY, USA, 211-220. DOI=10.1145/1570433.1570473 http://doi.acm.org/10.1145/1570433.1570473
9. W3C. Web Applications Working Group Charter. http://www.w3.org/2012/webapps/charter/ (Last access Apr. 2013).
10. WHATWG. Offline Web applications. http://www.whatwg.org/specs/web-apps/current-work/#offline (Last access Apr. 2013).

Decoupled Distributed User Interface Development Framework for Ambient Intelligence Systems

G. Varela, A. Paz-Lopez, J. A. Becerra , R. J. Duro
Integrated Group for Engineering Research, University of A Coruña
C/ Mendizábal S/N, 15403, Ferrol, A Coruña, Spain
gervasio.varela@udc.es

ABSTRACT
Ambient Intelligence (AmI) systems are required to be adapted to each scenario and use the available devices to interact with the user. Because of the variability of devices and the diversity of scenarios, they must support a wide number of devices and technologies, thus introducing a lot of complexity in the implementation of the UI.

This paper proposes an AmI UI development framework called Dandelion. It allows the development of highly portable AmI UIs by allowing developers to implement them using high level models that are transformed, at deploy time, into a selection of end devices. This transformation is driven by a novel device abstraction technology that encapsulates devices behind a generic set of distributed user interaction actions, isolating the devices, conceptually and physically, from the system's logic.

Author Keywords
User Interfaces; Ambient Intelligence; Distributed User Interfaces; Hardware Abstraction.

ACM Classification Keywords
D2.2 [**Software Engineering**]: Design Tools and Techniques – *User interfaces*. H5. [**Information interfaces and presentation**]: User Interfaces – *Interaction styles, Input devices and strategies, user interface management system (UIMS)*.

General Terms
Human Factors; Design.

INTRODUCTION

Every Ambient Intelligence (AmI) system aims to integrate itself in the daily life of its users, providing its functionality in an unobtrusive and natural way. Two features are critical to achieve this goal [1, 2]: rely on context information to adapt the behavior of the system to the specific characteristics of the environment and users, and enrich the environment with devices, taking advantage of them to interact with the user and the environment itself.

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

This last feature makes AmI UIs very dependent on their ability to use the sensing, actuation and appliance devices available in an environment [2]. These devices can vary considerably from one scenario to another; they may have very different natures, technologies, and paradigms of interaction with the user. Support this wide variety of devices in a system would be quite complicated. For this reason, many AmI systems are built for a specific set of devices in a particular environment and, therefore, they are very hard to deploy in a different scenario.

The work presented in this paper aims to alleviate this problem by providing an UI development framework that, relying on model-driven engineering techniques and distributed hardware abstraction technologies, allows the development of UIs decoupled from the technologies and locations of the devices chosen to interact with the user. This framework, called Dandelion, has two main components. A model-driven UI management system allows the description of the UI using high level models, whose abstract elements are connected, at deploy time, with a selection of distributed devices. This connection is implemented by the Generic Interaction Protocol (GIP), a device abstraction technology that, implemented as a distributed agent communication protocol, encapsulates the specific behavior of sensor and appliance devices behind a generic interface of distributed interaction actions.

Model-driven approaches have been very successfully within the Human-Computer-Interaction (HCI) community since Thevenin and Coutaz [4] proposed a model-driven approach to support UI adaptation to context changes. Many authors in the field of multiplatform UI development have used similar approaches [5, 6, 7], and it has also been used in Ubiquitous Computing (UC) and Ambient Assisted Living (AAL) systems [7, 8]; which are closely related to the AmI field. Nevertheless, those works have been mainly focused in the adaptation of graphical user interfaces to multiple platforms and displays. The work presented in this paper takes a novel approach by abstracting sensor and appliance devices behind a common set of interaction actions to build AmI UIs that are independent of the specific hardware devices.

This paper is organized as follows. Section 2 provides an overview of the Dandelion system. In section 3, the GIP is presented in detail. Section 4 and 5 are dedicated to the description of other important subsystems of Dandelion.

Section 6 shows a brief use case example. Finally some conclusions are extracted.

OVERVIEW OF DANDELION

Dandelion is being implemented as the UI development framework of the HI^3 general purpose AmI platform [9]. It follows a model-driven approach for distributed UIs inspired by Cameleon-RT [5], and it is integrated vertically in the different layers of the HI^3 architecture, which are described in detail in [9].

Dandelion can be divided into two main blocks. On one side, a model-driven UI management system allows developers to define the UI at an abstract level using the UsiXML abstract UI model [6]. On the other, the GIP provides a distributed communication interface that abstracts devices behind a generic set of interaction actions, allowing a decoupled connection of the abstract UI definition to a set of distributed elements that perform the real interaction with the user. Figure 1 shows a block diagram with the different components implementing these two blocks. The model-driven UI management system is implemented by the abstract UI model and the Application Controller (AC), which manages the connection between the abstract UI definition and the devices. The abstraction of devices is conceptually implemented by the GIP and physically realized by the Final Interaction Objects (FIOs), which provide software abstractions of devices by implementing the GIP interface.

Following the Figure 1, from top to bottom; the application logic and the UI description models are the only elements provided by the developer. She describes the UI using the UsiXML Abstract UI model, and then connects the application data and actions to that description. The AC manages this connection by storing a set of mappings between the data objects, the elements of the model and the real devices that will be used to interact with the user. Those real devices are represented in the system by the Final Interaction Objects (FIOs). They encapsulate the specific logic, characteristics and particularities of each device behind the GIP interface.

The physical connection between the AC and the FIOs is decoupled by using the GIP. The AC monitors the application, and when it detects a change in the data model associated to the UI model, it sends a GIP event to the FIOs mapped to the corresponding UI element. Conversely, when the AC receives a GIP event from a FIO, it uses the mapping to know which application data object or action needs to be updated. This process is transparent to the application and the developer.

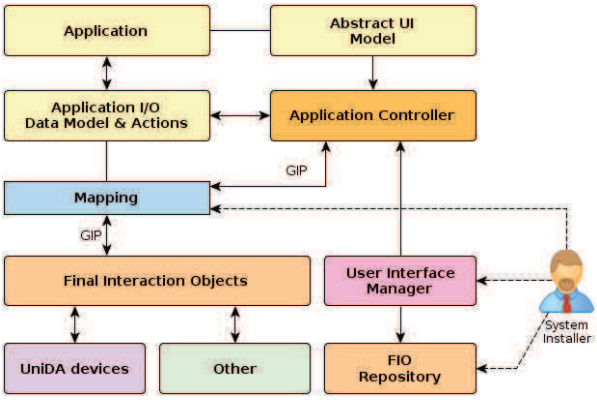


Figure 1. Dandelion system component diagram

The mappings between elements described in the abstract UI model and the FIOs are stored by the AC, but they are managed by the User Interface Manager (UIM). While Dandelion is designed to support the autonomous selection of FIOs at runtime, in the current version this mapping is performed manually by the system installer; who at deployment time selects out of the available FIOs those that better suit the requirements of the application, the user and the environment.

In the next sections the AC, the GIP, the FIOs and their relationships are explored in detail.

GENERIC INTERACTION PROTOCOL

The Generic Interaction Protocol, GIP, is one of the main contributions of this work. It provides a new level of abstraction for distributed hardware devices, encapsulating their specific behaviors behind a generic interface of user interaction operations.

The GIP defines a reduced set of interaction actions which provide a generic remote interface to any kind of interaction device. By implementing this interface, any device can be accessed using the same set of concepts and operations, thus decoupling the application from the underlying interaction technologies and the location of the devices.

The GIP has been designed as an event based distributed protocol following a publisher/subscriber model. A series of publishers, the FIOs, publish events notifying actions that the user has performed (input/selection of data, activation of an action) and a series of subscribers receive those events and react accordingly, the ACs of the different applications. The GIP has been implemented as a distributed agent protocol using the publisher/subscriber capabilities of the HI^3 platform [9]. Figure 2 provides a good overview of the GIP and the relationship to the AC and the FIOs.

The generic interface provided by the GIP defines five events inspired by the set of abstract interaction units (AIUs) used in the UsiXML abstract UI model to describe the UI interaction requirements at an abstract level:

- *output*: an AC requests FIOs to perform data cutput
- *focus*: an AC requests a given FIO to gain focus over the user attention
- *selection*: an AC requests FIOs to show a selection of data; or a FIO informs an AC about the selection of a user
- *input*: a FIO informs every subscribed AC that the user has performed a data input action
- *action*: a FIO informs every subscribed AC that the user has activated an action

Every GIP event has associated a set of data properties which are the data a FIO is able to either output to the user, or gather from her as input. These data is represented as a string and has an associated basic type (integer, double, byte or string) to it so that Dandelion can know what kind of information a FIO is able to represent.

In order to allow some level of customization of the user interface, GIP events are enhanced with a set of properties called Interaction Hints (IH). They are a set of fixed properties that developers can use to provide indications to the FIOs about an interaction action (ex. priority, size, color). The support for IHs is not mandatory, and each FIO can perform them as it wants.

USER INTERFACE DEFINITION AND MANAGEMENT

Dandelion allows developers to describe the UI of their applications using the abstract UI model of UsiXML [6], a user interface description language (UIDL) for the description of different aspects of an UI.

The abstract UI model introduces the concept of Abstract Interaction Unit (AIU) as a representation of the typical widgets found in graphical user interface toolkits. In Dandelion this concept represents any kind of interaction element, such as physical devices like appliances or sensors, or even gestures or GUIs. Using the abstract UI model, a developer describes the application UI as a collection of interrelated AIUs representing the different high-level user interactions (input, output, action, and selection) required by an application and how they relate to each other (containers). The GIP interface is designed to match this set of generic interactions so that it is easier to connect an AIU to a FIO.

Apart from specifying the UI at an abstract level, the developer must provide a set of data and action objects that will be used by Dandelion as the I/O interface between the application logic and the UI. The developer associates every one of these objects to an AIU and, as shown in Figure 2, the AC uses the observer pattern to monitor those objects, detect changes, and send the data to the FIOs, and vice versa. When a GIP event is received by the AC, it looks for its associated AIU, and uses a callback to notify the change to the application logic. This last association is

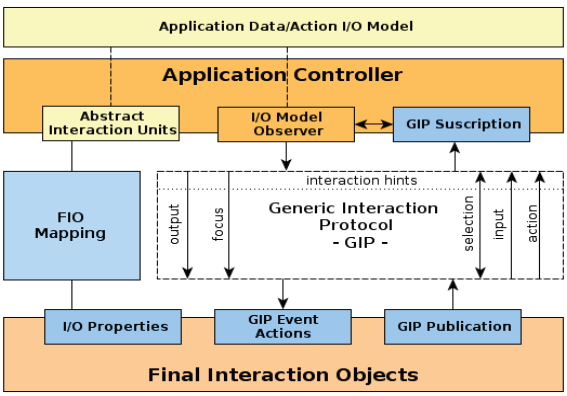


Figure 2. Relationships between the AC, GIP and FIOs

managed by the system installer, who manually connects the AIUs to the GIP properties of the concrete FIOs.

The combination of the abstract UI model with the usage of the observer pattern to drive the connection between the application and the UI, allows a great level of decoupling between the application business logic and the user interface. Nevertheless, this generic description of the UI and its behavior can impact the capacity of UI customization that can be achieved, which is alleviated to some extend by using the GIP interaction hints.

FINAL INTERACTION OBJECTS

The Final Interaction Objects are the end elements in charge of physically interacting with the user. They are software abstractions of heterogeneous interaction resources that could be either hardware (appliances, sensors, etc.) or software (GUIs, voice recognition, etc.).

FIOs are implemented as agents realizing the GIP interface in the sensing/actuation layer of HI^3. They assume the role of GIP publishers, abstracting the behavior of a device as a set of events notifying the different actions the user is performing. Each FIO implementation is only required to support a subset of GIP events because there can be devices that only support input or output. The supported GIP events and the data properties a FIO can input/out are specified in a FIO description, which can be consulted by the system installer using the FIO repository. See Figure 1.

For every different kind of device or interaction software used by an application, a FIO must exist that abstracts them using the GIP. Obviously, a key point is that developers should not need to develop their FIOs, or at least, not many of them. They should be provided by Dandelion itself or by the manufacturers of the interaction resources.

In order to alleviate the problem of developing FIOs for the wide number of different devices and technologies available, Dandelion makes use of the hardware abstraction layer of HI^3, UniDA [4], which provides a generic interface to remotely access and use any kind of hardware device. In UniDA every device is accessed using the same generic operations and concepts, and each type of device is reduced to a set of common operations. It is consequently, possible to use similar devices from different manufacturers or technologies using the same exact API. This way, one FIO can support a wide number of physical devices.

EXAMPLE USE CASE

To briefly illustrate how Dandelion is used to build AmI UIs, a small real world example has been implemented. It is a notification system for a home assistant application, which would be in charge of notifying events, alarms or messages to the users.

The UI of the system is a fairly simple one. It only requires an output element to show the message or notification, and an input one to discard the notification. In a classical PC system it could be a simple notification dialog with a label

and a button. Figure 4 shows how this simple interface could be described using UsiXML. It only requires an output AIU and a trigger AIU that are related inside a container using a compound AIU.

The association between this abstract UI and the AC is achieved by associating a string object to the AIU with id '2' and a callback action to the AIU with id '3'. This is done programmatically in the application.

Our system needs now some FIOs that implement the real interaction with the user. In order to illustrate how this UI could be realized in different environments, a small set of different FIOs has been implemented:

- *output*: colored lights using UniDA
- *output*: An overlay GUI display in a TV set
- *output*: voice synthesizing using Festival
- *input*: a light switch connected to a KNX home automation network using UniDA
- *input*: a hand gesture in a Kinnect like device

The three output FIOs provide one I/O string property indicating that they are able to present a string to the user. Obviously, they will transmit the message in different ways. The colored light device will only notify the presence of a message by powering on a colored light, while the TV display will show the complete message. The input FIOs only provide Action GIP events to notify when the user activates the switch or makes a specified gesture with a hand.

As shown in Figure 3 two different scenarios have been contemplated and, depending on the scenario, the system installer would associate the AIUs to each of the different available FIOs. For example, in the home of a deaf user, the output AIU could be connected simultaneously to colored light FIOs and to a TV FIO. When the application output a notification, the lights will power on, so that the user knows that there is a notification, and can go to the living room to read it on the TV and use the light switch to discard it.

CONCLUSIONS

This paper has shown how the combination of model-driven engineering techniques with distributed device abstraction technologies like the GIP and UniDA establishes a promising approach to allow the development of AmI systems decoupled from the physical location and particular complexities of interaction devices.

```
<AbstractUIModel>
  <AbstractCompoundIU id="1" shortLabel="Notification Dialog">
    <AbstractDataIU id="2" shortLabel="Notification Label">
      <AbstractDataItem\>
      <AbstactOutputIU\>
    <\AbstractDataIU>
    <AbstractTriggerIU id="3" shortLabel="Discard action">
      <AbstractOperationIU\>
    </AbstractTriggerIU>
  </AbstractCompoundIU>
</AbstractUIModel>
```

Figure 4. Description of the example UI with UsiXML

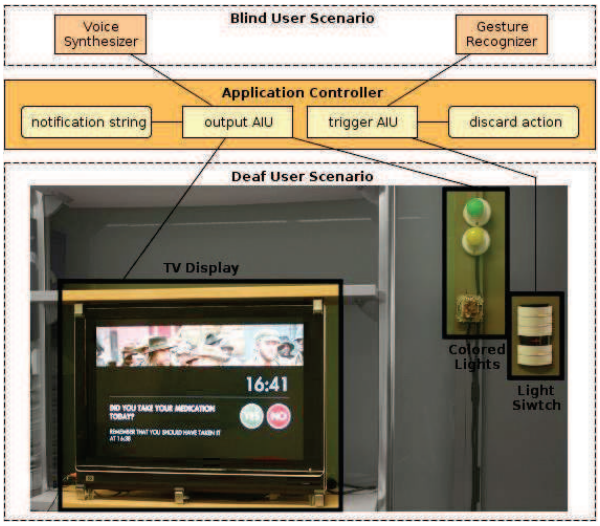


Figure 3. Use case example block diagram and photo

Using Dandelion, developers are able to build AmI UIs declaratively and isolated from the final implementation, thus allowing more portable AmI systems. Furthermore, the UIs are described using machine readable models which in the future could be used to autonomously manage the adaptation of the UI to specific scenarios. Nevertheless, the generic character of the GIP and the abstract UI model restricts the customization capability of the UI, which should be one prominent area of future work.

REFERENCES

1. Augusto, J. C., & McCullagh, P. Ambient Intelligence: Concepts and Applications. Int'l J. Computer Science and Information Systems, vol. 4, number 1, 1–28.
2. Dadlani, P, Peregrin Emparanza, J, & Markopoulos, P. Distributed User Interfaces in Ambient Intelligent Environments: A Tale of Three Studies. *Proc. 1st DUI*, University of Castilla-La Mancha (2011), 101-104.
3. Varela, G., et al. UniDA: Uniform Device Access Framework for Human Interaction Environments. Sensors 11 (10), MDPI (2011), 9361–9392.
4. Thevenin, D., & Coutaz, J. Plasticity of user interfaces: Framework and research agenda. *Proc. INTERACT'99*, IOS Press (1999), 110-117.
5. Balme, L., et al. Cameleon-rt: A software architecture reference model for distributed, migratable, and plastic user interfaces. *Proc. EUSAI 2004*, Springer-Verlang (2004), 291-302.
6. UsiXML. http://www.usixml.org, http://www.usixml.eu
7. Blumendorf, M., Lehmann, G., & Albayrak, S. Bridging models and systems at runtime to build adaptive user interfaces. *Proc. 2nd EICS 2010*, ACM (2010), *9-18.*
8. Abascal, J., & Castro, I. F. de. Adaptive interfaces for supportive ambient intelligence environments. *Proc. 11th ICCHP, Springer (2009),* 30–37.
9. Paz-Lopez, A., et al. Some Issues and Extensions of JADE to Cope with Multi-agent Operation in the Context of Ambient Intelligence. *Proc. PAAMS*, Springer (2010), 607-614

Interconnecting Current Technology in Healthcare Environments

Juan Enrique Garrido
Computer Science Research Institute
University of Castilla-La Mancha
Albacete, Spain
juanenrique.garrido@uclm.es

Víctor M. R. Penichet, María D. Lozano
Computer Systems Department
University of Castilla-La Mancha
Albacete, Spain
{victor.penichet, maria.lozano}@uclm.es

ABSTRACT
Medical staff conditions can be improved in many cases avoiding problems and accelerating procedures that finally affect to patients' health and well-being. Current technology allows the development of systems which automatically offer information and functionality based on users' needs. Additionally, new interaction mechanisms make it possible to detect and analyze the patient's posture, and react accordingly to it. In this way, this paper presents Ubi4health, a system based on new technological opportunities, which automatically manages tasks and emergencies, detects falls and fainting spells and also, incorporates an innovative rehabilitation procedure. The system allows patients to perform their rehabilitation process at home always under the physiotherapists' supervision. Finally, adequate staff can define each rehabilitation plan through an online tool at anytime and anywhere by recording their own exercises through Kinect. The system functionality is deployed by a set of five modules that are interconnected composing an integral solution. Each module offers its interface by distributing it among those devices which require its functionality.

Author Keywords
Ubiquity; Collaboration; Context-Awareness; Awareness; Healthcare; Interaction; HCI.

ACM Classification Keywords
H.5.3: Group and Organization Interfaces.

General Terms
Design, Human Factor, Management.

INTRODUCTION

Healthcare environments may lead to many conditions ([2], [3]) and situations which create a complicated work atmosphere. Employees usually have to attend to more than one task at the same time and also they have to control the end of related processes (e.g. the reception of blood analysis results that are required to apply a specific treatment).

An additional problem appears when healthcare employees have to immediately address unexpected situations which appear suddenly such as falls and fainting spells [6]. These situations can be a significant oversight in healthcare, mainly when they occur out of the employees' line of vision. It is really difficult to ensure that every resident is at least supervised by one employee at any time.

Moreover, healthcare centers may offer rehabilitation treatments in order to improve injuries healing. Generally, medical staff has to attend personally each rehabilitation process while patients are working on them. On the one hand, the rehabilitation process implies that physiotherapists have to control, analyze and evaluate the related process of each patient and also, to adapt it if necessary. On the other hand, patients must go to the medical center. In this sense, they have to leave their home and routine.

In this paper we present Ubi4health, a solution to address each of the described problems in healthcare centers: task and emergency management, fall and fainting detection, and the improvement of rehabilitation processes. The system relies on three fundamental features to complete the objective: collaboration, [4], ubiquity [5] and context-awareness [1]. Collaboration allows the staff to be in contact and work together. Ubiquity allows the system to offer connectivity to employees every time and in any circumstance. And context-awareness is the capability to adapt the functionality based on staff and patients needs.

Ubi4health mainly consist of five modules where functionality is distributed: server, tasks module, falls and fainting spell detection module, exercise set generator module and rehabilitation module. All the components are integrated in the same system. Each module offers its interface that is distributed among the devices as needed. This distributed user interfaces appears in order to satisfy the needs of the medical staff and patients or residents to complete their daily tasks.

The paper is organized as follows: Section 2 describes the developed system focusing on its features. Additionally,

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

this section has a subsection that describes the modules which compose the system and their needed deployment. Finally, Section 3 presents some conclusions and future works.

UBI4HEALTH SYSTEM

The main objective of Ubi4health is to improve the conditions of healthcare environments. Concretely, the following facilities are offered as the main improvements:

1. Each employee is interconnected with his workmates in order to work collaboratively.
2. The tasks to be performed are reminded and if necessary, these are automatically assigned and reorganized.
3. Medical staff has always adequate information based on their context, that is, based on his location, current and following task, experience, etc.
4. Falls and fainting are automatically detected.
5. The rehabilitation staff can extend their action range to patients' home.
6. And finally, the rehabilitation exercises can be online defined at anytime and anywhere.

Functionality through the Distributed User Interfaces

The system functionality is organized based on a set of five modules as it is shown in Figure 1. Each of them is focused on a specific aspect: server, tasks, fall and fainting spell detection, exercise set generator, and rehabilitation process. The modularization allows dividing the deployment to be used. The proposal composes an integral solution for healthcare centers without binding to deploy all the system modules. Therefore, each medical center can use the system as a whole integral solution or use only the specific needed parts. Each functionality of the system offers an interface which is distributed among to those devices that will be used by users' who need the related functionality. Therefore, the users obtain in their device the interfaces that correspond with the functionality they need based on their role and the task they may perform. For example, an auxiliar can be responsible for controlling that a patients' set take their medicines and also, to attend any possible fall or fainting. The device may show the interface of tasks functionality to indicate information about their work to perform and also, the interface of falls and fainting module when an emergency is detected.

The *Server module* offers the services to obtain the needed information for any of the other functionalities. Therefore, the server is responsible of connecting the other ones. An existing local network or Internet can provide this connection. The communication comes from the use of a common database. This essential element contains the information that each module needs about medical staff, patients, tasks, rehabilitation processes, etc. In this way, all the modules works with updated information since any data change is always performed in a common data source.

The *Task module* (see example in Figure 2) helps staff in their task management. The module provides a way to know what each employee has to do at any time. This information is used to remind to the medical staff what are their assigned tasks avoiding oversights. Additionally, the system is able to assign automatically unassigned tasks to the most adequate employee, based on medical staff context (location and the urgency of their current and next tasks).

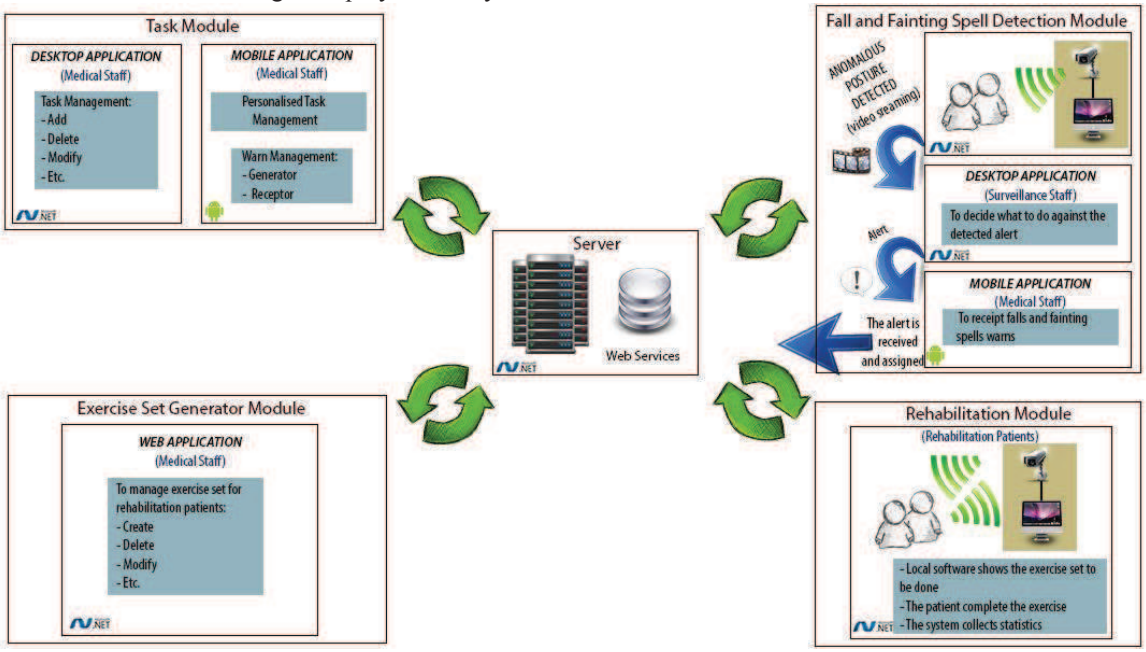


Figure 1: Ubi4health deployment in which is possible to appreciate its different modules

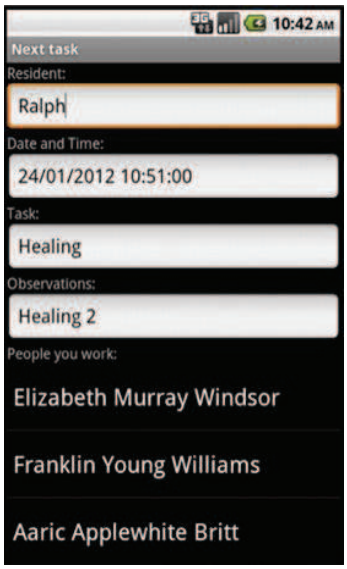


Figure 2: Example of next task reminder from Tasks module

This module contains a communication mechanism that allows the collaboration between the staff through a note synchronous system. To theses objectives, the deployment needed is composed by an Android mobile device with which medical staff can access to needed information at anytime and anywhere. Additionally, those users who do not need mobility can use a personal computer.

The *Fall and Fainting Detection module* provides a way to detect falls and fainting spells where no employees attend the residents. The module is based on the distribution of multiple Kinect devices in appropriate locations around the healthcare center. Accordingly, the module uses the Kinect's movement interaction to identify different postures in order to detect anomalous ones. When a problem is detected, the system notifies the most adequate employees in order to attend the incidence as fast as possible. Firstly, the security staff receives the incidence via video, what allows looking for any incidence, that is, for any fall or fainting. If a dangerous situation is shown, the vigilant sends an emergency signal to the most adequate employee in the medical center based on his current context. To that end, the security staff may use a personal computer in which be able to see on a screen the incidences and the list of adequate workmates to attend them. The medical staff is continuously in movement; therefore they may use an android mobile device through which receives advices notes and confirm that they will be attended immediately.

The *Rehabilitation module* offers a complementary rehabilitation procedure based on the traditional one which provides patients with autonomy. The module allows patients with neurodegenerative diseases or some brain damage (e.g. traumatic brain injury, dementia, cerebral palsy, paraplegia, etc.) to complete their rehabilitation at home using a Kinect device and a computer connected to a screen. Up to the moment the system is focused on two of the most common rehabilitation exercises: (1) walking straight for patients with balance disorder (see Figure 3), and (2) getting up from a chair (see Figure 4). In this way, patients will do the exercises that qualified medical staff has previously assigned and defined. Basically, this part of the proposal helps patients in their rehabilitation process showing the steps to be followed always under the physiotherapist supervision. Additionally, the module corrects the patients if necessary. The benefits can be seen from two points of view. Firstly, from the medical staff perspective, who finds the system a perfect tool with which to gain time. The system offers physiotherapists the possibility of not to attend the rehabilitation process continuously. Such exercises imply many repetitions of the same exercise. On the other hand, patients can be at home in a well-known and more comfortable environment; something that would certainly contribute their physiological benefit. Additionally, the patient can perform the rehabilitation process in an independent and less invasive way without the need to depend on third parts more than the strictly necessary.

Finally, the *Exercise Set Manager module* provides an additional tool for physiotherapist to compose exercises plans for rehabilitation procedures. This module gives physiotherapist freedom to create any exercise set based on Kinect captures. Concretely, the system gives the possibility to use standard or previously recorded postures and also, new postures that the user has to record through Kinect. An important feature of the module is the possibility to be used offline or online. Therefore, users can create new rehabilitation plans at the medical center, or they are able to use it wherever they want through Internet instead.

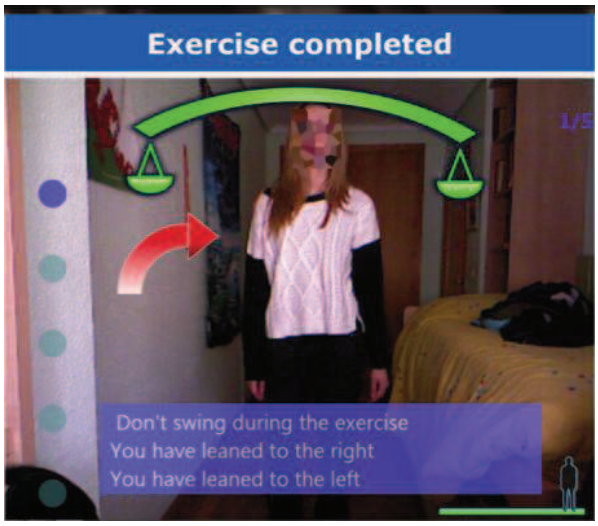


Figure 3: Example of training to walk straight from Rehabilitation module

Figure 4: Example of learning to get up of a chair from Rehabilitation module

As previously mentioned, Ubi4health's modules are interconnected but each of them are focused on a specific aspect. This modularity creates an integral solution which is extensible. This foundation offers the possibility to create new modules in order to extend the reach of the system in order to solve new or existing problems in medical centers. To that end, being centered on a specific technology is not necessary; the unique requirement for the new development is to work with the data of the server through the offered web services.

CONCLUSIONS AND FUTURE WORK

In this paper, authors have presented Ubi4health, a health care system that provides a way to organize the agenda of the medical staff due to an automatically assignment of tasks. The system manages complex situations such as the detection and warning creation of falls and fainting spells through movement interaction using Kinect devices. Additionally, the proposal generates an innovative rehabilitation environment. This environment offers the patient the possibility of performing their rehabilitation plan at home through movement interaction always under the physiotherapist's supervision. Meanwhile, the medical staff can define each exercise set for each rehabilitation plan through the system using existing postures and movements or new ones created by a deployed Kinect recording system.

In essence, the proposal offers an integral solution for medical staff (1) to avoid oversight when performing tasks; (2) to locate and attend emergencies as fast as possible; (3) to perform rehabilitation process in a more confortable place for patients; and (4) to facilitate the creation of new rehabilitation exercises independently of the staff location.

The proposal works with distribution user interfaces since each module that composes the whole system offers its own interface. In this way, the system shows at any time the interface that users need based on their role, task to perform and current context.

There are interesting future works to be applied in Ubi4health. Authors think about the use of sensors to improve the needed tracking process when detecting falls and fainting spells. Regarding the rehabilitation module, which is deployed at the patients' home, two improvements are considered. Firstly, the interface is being improved in order to get a more friendly and intuitive system for the patients. The idea is to offer an interaction as natural as possible with the system and then, to obtain a greater level of acceptance. And secondly, additional exercises can be added to the module in order to offer a more complete rehabilitation range.

ACKNOWLEDGMENTS

We would like to acknowledge the project CICYT TIN2011-27767-C02-01 from the Spanish Ministerio de Ciencia e Innovación and the Regional Goverment: Junta de Comunidades de Castilla-La Mancha PPII10-0300-4174 and PII2C09-0185-1030 projects for partially funding this work.

REFERENCES

1. Anagtopoulos, C.B., Tsounis, A., Hadjiefthmiades, S. Context-Awareness in Mobile Computing Environments. Wireless *Personal Communications,* 42(2006), 445-464.
2. Arnrich, B., Mayora, O., Bardram, J., Tröster, G. Pervasive Healthcare, Paving the Way for a Pervasive, User-Centered and Preventive Healthcare Model. *Methods of Information in Medicine,* 1 (2010), 67-73.
3. Maderia, R.N., Postolache, O., Correia, N., Silva, O. Designing a Pervasive Healthcare Assistive Environment for the Elderly. In *Proc. Ubicomp 2010*.
4. Poltrock, S., Grudin, J. Computer Supported Cooperative Work and Groupware (CSCW). In *Proc. Interact 2005.*
5. Weiser, M. The Computer for the 21st Century. *Scientific America 265,* 3 (1991), 94-100.
6. Xinguo, Y. Approaches and principles of fall detection for elderly and patient. In *Proc. HealthCom 2008*, 42-47.

Emergency Alert Management by means of Distributed User Interfaces

Habib M. Fardoun, Abdulfattah S. Mashat, Lorenzo C. Gonzaléz
King Abdulaziz University, Faculty of Computing and Information Technology
Jeddah, Saudi Arabia
{hfardoun, asmashat, lgonzalez}@kau.edu.sa

ABSTRACT
States of emergency are always an element that must be addressed quickly and concretely, but it cannot always be done in a proper way. This is because when a call of this type occurs in the system, it is possible that the person who performs it does not know exactly where he is, not be understood due to the excitement, or because the call quality is not adequate or there is too much noise around him. Therefore in this paper a new way of managing these alerts, achieving that through new technologies, communication between the user and the system become much easier, thus it improves the ability to act. This is done, by applying Global Positioning System (GPS), which is able to know the location of the emergency, with Web services we will have access to Cloud by using any platform, and with distributed user interfaces application model synchronized in the control center, this will remain the system robustness. But not only that, it is also about managing this type of alert messages to make to work with less error margin and not to have fraudulent requests in the system, for that we will use an expert system to evaluate all requests and classified according importance.

Author Keywords
Emergency Alerts, Information and Communication Technologies, Global Positioning Systems, Cloud Services, Mobile Devices, Distributed User Interfaces

ACM Classification Keywords
H.5.3: Group and Organization Interfaces.

General Terms
Design, Human Factor, Management

INTRODUCTION
Everything relates to emergencies should be treated with special care and to be resolved in a quick and effective way [1]. This paper addresses a specific part within the whole system and possibly the start of any proceedings arising as a result of an emergency; we are talking about the management of emergencies and management requests through distributed user interfaces [2]. For some time, this type of request was made by telephone, where the user must explain clear enough what happened to the one who he is calling [3]. The whole process works in the right way, but we must take into account several factors that make this method a method that is becoming obsolete and in need of improvements. One of these factors is the mood of the user who holds the phone during the call, which is not always in a calm state, and has a sociable and communicative power. Another factor is the possibility that there is no phone coverage in the scene, whether he is in a leisure center, tunnel, etc. Last but not least, there is fraudulent calls or free emergency whose main consequence is the saturation of the network with useless requests using telephone and personal resources that could be used in a better way.
What we propose in this paper is to attempts a new management system where emergency requests will use a control system based on a synchronized distributed user interface. So that instead of using the telephone network for this purpose we use the provided Internet resources. The platform consists of: mobile applications for users with different versions of operating systems, servers that can be accessed via Web services, an expert system that processes and evaluates requests, database which is responsible for storing all the information, and main application responsible for displaying all information to emergency operators and sync it to avoid duplication in the process. Web services, servers, the expert system and the database are maintained following a Cloud platform. Always considering that scheduled backups are done to protect the system against data loss and is possessed of a protocol to act if failing situation happens if any component stops working.

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.

GLOBAL VIEW OF THE SYSTEM
The system is operated by the time that anyone needs some kind of emergency assistance. For this it is essential that the application is installed in all mobile devices, just as the number “112” [4], which is always present in any of these devices, Figure 1 shows a sequence diagram associated with the system operation when an emergency occurs, for the shown case it notify and act without further information.

Once the user has experienced or witnessed a state of emergency, he use his mobile device to run the application

so he can notify the emergency team, being an ambulance, police, fire or other. The application provides a text field where the user type the information related to the incident and once started, it starts the GPS device to locate its position in the process. The application of the text field also has some keywords that the user can click to streamline the process, such as an accident, very grave, unharmed, fire, wounded, dead, unconscious, car, motorcycle, home, etc. After he finish writing the message, the user send the content of the message near to his GPS position from which it was written, so that it is not only the information associated with the event but also the device information, the informant and location. We can say that the application has an option to check for the user to allow his information to be stored in the system following the user rights and the rules of data protection laws [5].

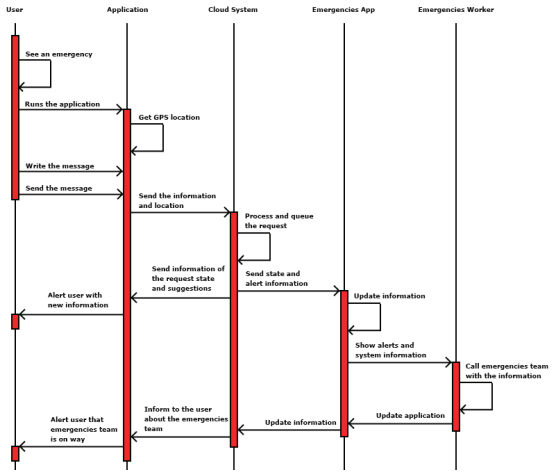


Figure 1: Ideal sequence diagram in the process

The message and all associated information will reach the server where they are processed and reviewed by the expert system responsible for this effect. This system evaluates the severity of the request and the function of the gluing itself [6]. It is also responsible for eliminating fraudulent claims, in which case it could be some kind of penalty to the owner of the device, which the systems known his position. At the time that the request is in the system, an alert is sent to the user to inform the processing of his application along with suggestions to reassure or to act until the help arrives.

The new information in the system is updated directly in the control application available to emergency workers at its headquarters. Thus in every moment they know what emergencies are happening, its severity and what assistance is needed for it. If necessary, these operators would contact through instant messaging with users through the same application or even telephone. Once emergency workers have all the required information, then they prepare to launch the associated protocol by sending the appropriate emergency equipment. Just then the entire application is updated to avoid duplication, and alert the user notifying him that assists are underway. It must be said that the view of the control application will vary depending on the actions taken thereon. Thus, there may be a worker who is viewing information about an alert relating to a fire, and another with an alert associated with a traffic accident.

The whole process and information within it is stored in databases to monitor all the activities on the system, so as to reduce as much as possible the inappropriate actions and in turn keep a record of everything that happened. Besides the above along with maintaining backups of this information, there is a performance-focused hardware in case of failure of a similar component [7].

ARCHITECTURE OF THE SYSTEM

For this project are mainly used technology components shown in Figure 2, where the elements are not related to backup or replication hardware. Next we present the related parts with the applications (mobile or control), Web services and the expert system, are the most important to explain. This information is supplemented with details from the previous section related to the overall operation of the system.

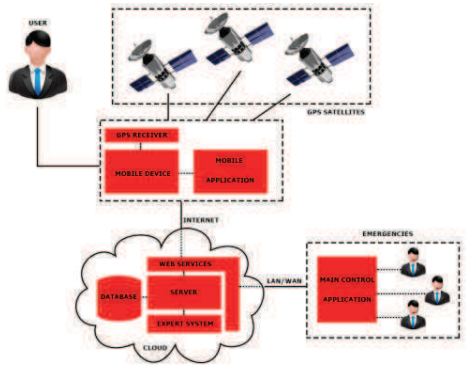


Figure 2: System architecture.

Mobile application

It is the application that must be installed on the user's device, so that anytime he can make use of it in an emergency situation. It provides a direct communication between the user and emergency to avoid communication delays [8]. This communication is done via instant messaging, where the user is the emergency center. It also has a feature that allows the user to describe the situation in the easiest and fastest way possible.

Figure 3 shows the discussed functionality, which provides the user with the keywords that may be useful in the description of the event. These words are updated based on the content of the message text to be adjusting to what the user needs to write. Besides the above, the system warns the user of the status of his request (as shown in Figure 1) and messages by vibration, which provides suggestions for behaviour and performance as a function of incident occurring.

Figure 3: Emergency Alert Application for Mobile Devices

Applied Web Services

The user requests are made through the mobile application and through the Web services for that purpose [9]. Such services are always available because requests always have to be treated at any time and any day of the year. In short they have contingency plans in the event that the services provided by a server is not available, redirecting requests to others until the problem is corrected, similarly to the case of passing communication between the application server and the main control.

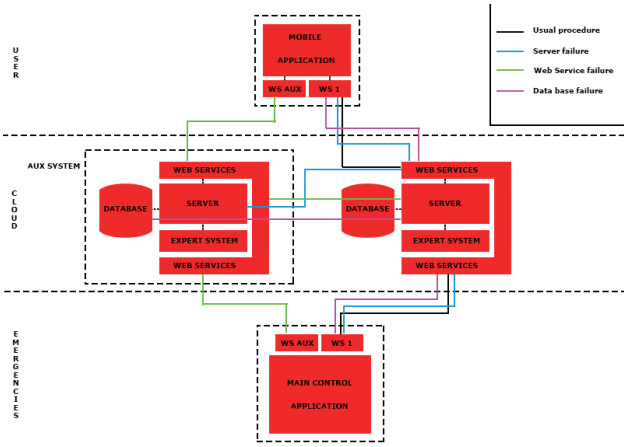


Figure 4: Diagram of the contingency plan

Expert System

This expert system is based on fuzzy rules [10], so that the system outbound requests, after processing by the inference engine, are ranked according to their importance: uneventful, very mild, mild, serious, very serious and immediate attention. This makes it much easier for emergency operators to evaluate and act accordingly to the situation. It also controls that the system inputs are not fraudulent by evaluating the message content of the alerting users, in which in this case the user will be discarded and informed of the possible consequences if he continues this behavior.

Main Control Application based on Distributed User Interfaces

This application is located in the headquarters of emergency equipment control, where operators are responsible for evaluating and implementing the results shown for the same from the previous processing expert system. This application must be well synchronized with other agencies to avoid duplication. So that when an operator runs a petition, he should automatically disappear from the view of other operators who are envisioning the application. Thus, when an operator selects one of the requests to be with him, it will be omitted from other instances.

Also keep in mind that the application groups the requests that are located at identical places in a similar time points to further expedite the processing thereof.

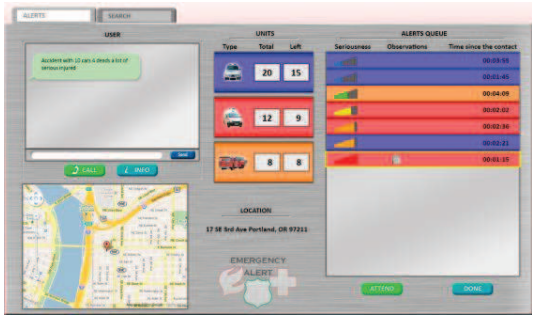


Figure 5: Main application in emergency headquarters

As seen in Figure 5, the application sorts the emergencies according to their severity [11]. In addition to each one it is assigned a color depending on whether the message content is related to the need for ambulance, police or fire. Alongside this is provided in the alert field named "Comments" which aims to inform the staff that monitors the implementation of the event characteristics. These characteristics are taken from the body of the message, so in case there is a fire, the fire icon appears, and so on. This achieves that the person responsible for managing alerts from the first look he will obtain a large amount of information related to a specific alert. Besides the above, Figure 7 corresponds to the terminal of an emergency operator, who has selected the most serious alert [11]. However in the other terminal of another operator, shall follow the same previous alerts minus the one that the first operator has selected.

DIFFUSE EXPERT SYSTEM FOR MAKING DECISIONS

This section describes an example of using the fuzzy expert system and inference for a given input [12]. This shall be based on the following request: "There has been an accident between my car and a bike. The person on the bike is on the ground unconscious and his pulse is very low. It was at the intersection of the street 1 with street 2. The system evaluates the request and takes into account the facts "accident", "car", "bike", "low pulse" and "unconscious", and subjects them to the knowledge base and the inference engine [13]. Depending on the different states involved in a

particular accident, they are classified into care or triage priorities [14]. So it associated those with extreme urgency with the priority 1, the priority 2 is about those who are seriously ill, the priority 3 for those who their condition is not urgent, and priority 4 or 0 for those that do not need assistance or have been killed. In the technical standard 2004 related to hospital emergency services in the health sector, we could see the symptoms that are associated with each priority status [15]. Figure 6 represents the fuzzy set.

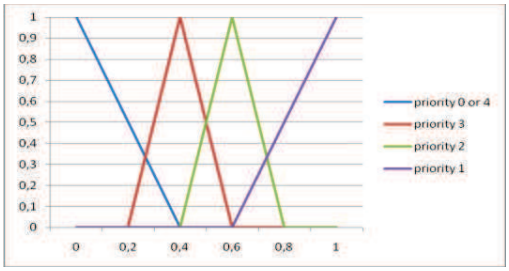


Figure 6: Fuzzy set for triage

Together with the previous set and the following rules that exists en the base of knowledge:

If state is “unconscious” then

priority is “priority 1”

If state is “priority 1” then

finalstate is “critical”

We proceeds the process of inference, where inputs are treated to obtain a conclusion output range which is determined by the fuzzy set, presented in Figure 9. The inference is carried out by applying fuzzy logic [6] [16].

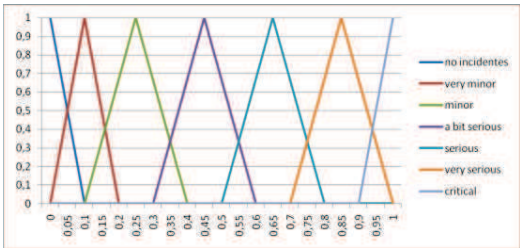


Figure 7: Fuzzy set for the resulting state to be displayed in the main application control

The final result is shown in each alert to thereby illustrate the user's control center what is the level of importance of it. With all this, it makes the ordinary the user will not have to spend time learning to use the system with a certain very specific discrete inputs, which would entail a very high learning curve that would take too much time and effort.

APPLYING DISTRIBUTED USER INTERFACES THROW THE SYSTEM

As can be seen, the system makes use of different types of interfaces [17], depending on whether the user is making the request to the system using the mobile device or emergency operator responsible for management of requests.

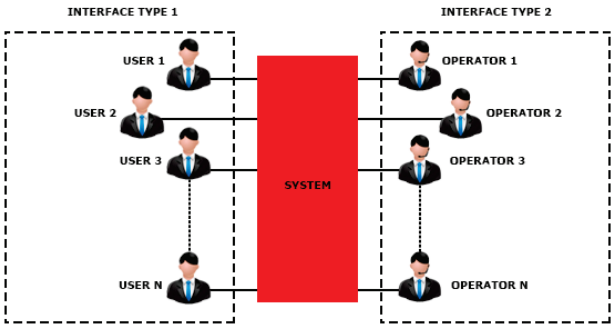


Figure 8: Fuzzy set for the resulting state to be displayed in the main application control

Figure 8 shows this fact, however, we must take into account that for example, each operator cannot see exactly the same image that his mate. This is because according to the selected alert the system displays different information and knowledge so that an operator cannot select or see the alert that is being treated by another, it can be concluded that the information shown on the display is different.

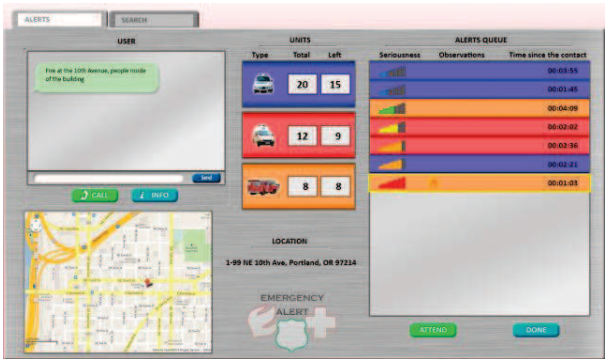


Figure 9: Screenshot of the application of a particular control emergency operator

Figure 9 shows the display screen of the application of an operator acting at the same instant that his mate does with the screen Figure 5. However, we can see how the shown data differ slightly to accommodate every request and for its proper synchronization, to achieve consistency and to avoid confusion or errors in its management.

Although, we may observe better the use of DUI within this application, when users share their application screen with the alerts operations room, to perform an action they do not understand or to clarify some actions that they should do to carry out a specific task. For example, the user of the application can share the camera of his device with the operator, how will points him over the image they both are viewing what he shall do with the situation he is facing. By this the user can feel more secure to touch an injured man and to be supervised by the operator (in this case a doctor), till the emergency team arrive the place.

CONCLUSION

As has been noted throughout the article, the proposed system is an improvement and updating of existing system for emergencies. It is necessary for the remodeling of it to suit the new generations that come with new technologies and models use different devices. The proposed system here provides both: access to the emergency system, and the exchange of information between the platform and the user, helping himself to act in the best possible way. It also decreases the response time due to poor communication between the two parties or the possible state of agitation of the issuer of the request.

Regarding to the future work, it has been raised a number of areas and functions that would add an improvement in communication between the emergencies and also between the user and the system, as follow:

- Ability to send photos or videos of the event on the part of users. This will increase the amount of information that the emergency operator can operate and act as best as possible based on the same.
- Improved interface to include elements that reduce the time spent on writing. Such elements are indicating options to choose various aspects related to the incident. Thus instead of writing what happened, they will choose the options you describe. For example, the system would provide the option of "event type" in which they could select fire, accident, theft, murder, another. It would also have other options such as "number of injuries" or "event severity", which may vary depending on the type of event.
- Availability of an application or program to the event description and details of the location. This would be similar to an element of a browser screen car show information in a visual deployed emergency teams. Thus both operators and emergency teams have the information, thus being able to act more quickly and efficiently.
- Possibility of feedback within the system of particular event information by emergency teams. It may be that once the emergency services arrive at the scene contemplates something other than what is detailed in the information. In that case it would be appropriate to update the information and thus the operator had the option of calling more teams or act according to new data. Also once there materialize emergency teams the exact area of the event with what geographic location would be updated with the right place.
- Creating a dedicated database to store the locations of existing cameras, so that when a certain incident occurs in a specific place, the cameras are arranged in such locations may take images of the event and a consideration of the scope.
- Inclusion of an improved parser for data entry. That way when the user enters quantities associated with objects, the system can quantify it and add to the information already obtained by the expert system. In addition, better manage data input to the system such as the example of "not breathing", "wheeze" or other. In this way be realized more information obtained and this enhances the evaluation of the situation by the platform.

REFERENCES

1. Walley, P. (2003). Designing the accident and emergency system: lessons from manufacturing. Emergency Medicine Journal, 20(2), 126-130.
2. Gallud, J. A., Tesoriero, R., Vanderdonckt, J., Lozano, M., Penichet, V., & Botella, F. (2011, May). Distributed user interfaces. In PART 2-Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems (pp. 2429-2432). ACM.
3. Grimes, Gary J. Cellular telecommunication switching system for providing public emergency call location information. U.S. Patent No 5,388,147, 7 Feb. 1995
4. Spanish emergency alert number. Link: http://www.112.es/.
5. ISO Pearson, S., & Benameur, A. (2010, November). Privacy, security and trust issues arising from cloud computing. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on (pp. 693-702). IEEE
6. Hagiwara, S., Oshima, K., Aoki, M., Murata, M., Ishihara, K., Kaneko, M., ... & Tamura, J. I. (2013). Usefulness of fibrin degradation products and d-dimer levels as biomarkers that reflect the severity of trauma. The journal of trauma and acute care surgery, 74 (5), 1275-1278.
7. Becker, G. A., Condon, R., Knight, D. A., Medders, D. E., & Rowan, M. (2010). U.S. Patent No. 7,844,577. Washington, DC: U.S. Patent and Trademark Office.
8. Liu, M., Ho, D. W., & Niu, Y. (2009). Stabilization of Markovian jump linear system over networks with random communication delay. Automatica, 45(2), 416-421.
9. Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2010). Web Services: Concepts, Architectures and Applications. Springer Publishing Company, Incorporated.
10. Fuzzy Expert System to Determine Criminologist Profiles Based on Lüscher's Test and Socio-Criminologist Variables. Christian Cardona, Est., Diana Restrepo, Est., Demetrio Ovalle, PhD. GIDIA. Revista Avances en Sistemas e Informática, Vol.4 No. 2, Septiembre de 2007, Medellín, ISSN 1657-7663 Edición Especial: II Congreso Colombiano de Computación – CCC2007
11. Brucker, A. D., & Petritsch, H. (2009, June). Extending access control models with break-glass. In Proceedings

of the 14th ACM symposium on Access control models and technologies (pp. 197-206). ACM.

12. Meharrar, A., Tioursi, M., Hatti, M., & Boudghène Stambouli, A. (2011). A variable speed wind generator maximum power tracking based on adaptative neuro-fuzzy inference system. Expert Systems with Applications, 38(6), 7659-7664.
13. Piltan, F., HAGHIGHI, S. T., Sulaiman, N., Nazari, I., & Siamak, S. (2011). Artificial control of PUMA robot manipulator: A-review of fuzzy inference engine and application to classical controller. International Journal of Robotics and Automation, 2(5), 401-425.
14. Monares, Á., Ochoa, S. F., Pino, J. A., Herskovic, V., Rodriguez-Covili, J., & Neyem, A. (2011). Mobile computing in urban emergency situations: Improving the support to firefighters in the field. Expert systems with applications, 38(2), 1255-1267.
15. Cooper, S., Endacott, R., & Cant, R. (2010). Measuring non-technical skills in medical emergency care: a review of assessment measures. Open Access Emerg Med, 2, 7-16.
16. Straccia, U. (2012). Description logics with fuzzy concrete domains. arXiv preprint arXiv:1207.1410.
17. Bar-Ilan, J., Zhitomirsky-Geffet, M., Miller, Y., & Shoham, S. (2012). Tag-based retrieval of images through different interfaces: a user study. Online Information Review, 36(5), 739-757.

Distributed User Interfaces to Enrich Collaborative Teaching Methods

Habib M. Fardoun, Daniyal M. Alghazzwi
King Abdulaziz University, Faculty of Computing and Information Technology
Jeddah, Saudi Arabia
{hfardoun, dghazzawi} @kau.edu.sa

Antonio Paules Cipres
University of Castilla-La Mancha,
Information Systems Department
Albacete, Spain
apcipres@gmail.com

ABSTRACT
Usually the teaching process includes collaborative working methods where teachers and students interact. This teaching process requires devices and applications that are in the market, but many of these applications for collaborative work do not allow the continuation of such methodologies in a simple manner. Our experience in the educative field leads us to view the importance to include within the traditional methodology distributed user interfaces for better collaborative education work in the classroom and as part of the second cycle of primary education. In this paper we present an improved tool to solve these problems, and with it, a scientific study from surveys and from real activity in a school, which give us feedback for our future research.

Author Keywords
Educational tools, Education Teaching Methods, Collaborative Teaching Methods, Information and Communication Technologies, Educative Cloud Computing, Educative Attributes

General Terms
H.5.3: Group and Organization Interfaces.

General Terms
Design, Human Factor, Education

INTRODUCTION

The development of educational applications for classroom must take into account the needs of collaborative work and projects, and enables the realisation of group dynamics and agility, without idle the student's time [1]. In a series of surveys conducted in schools, teachers raised the following questions [2]:

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.

- Do you apply collaborative work in your classroom sessions and divide it into projects? 75% of teachers said they did this type of session.
- Do ICTs streamline classroom work and the preparation of the applied sessions? 75% of teachers say yes to this statement.
- Do ICTs reduce students' time in the accomplishment of collaborative work sessions? 80% of teachers say no.
- Does the sharing of space and time appear right when you prepare your ICTs activities? 50% of teachers said they faced problems allocating time, and the other 50% responded that workspace sharing among students was inadequate.

Therefore, there is a real need to create an application that meets these needs in collaborative project work, and in addition improves efficiencies in both workspace and time. Time is important because the sessions are limited, so the teacher must estimate the approximate duration of each activity. We believe in the need for a precise definition of a classroom and collaborative workspace that allows the implementation of these activities, and in addition to more specifically define the basic patterns of educational collaborative work. With the increase in equipment that has occurred in recent years, and technological developments in IT devices, we found tablets, small-projectors, and transparent screens.

Schools with ICT equipment try to apply new methodologies for studying teaching methods [3]. During the development of this application we took the principles of distribution and the basic rules of educational collaborative work into the classroom. By taking these principles we developed an application, which is an extension of the traditional teaching method. To achieve this we employ concepts using collaborative work and distributed user interfaces, as well as including new equipment for the classroom to make learning interactive and real. The proposed application tries to link these concepts to the issues that schools face when using devices for collaborative educational work.

STATE OF ART

There are many commercial collaborative platforms where students and teachers can perform the teaching learning process together [4] [5]. Collaborative work is done by transferring files to the application, besides using user accounts for the work teams. The transfer of files allows students to search and find the file, but does not allow a dynamic classroom session, so the loss of time is considerable. An important point is the structure of the classroom. It is a traditional distribution, and we can guess that the methodology used will grab the traditional lecture method. Where the teacher put his contents as we can see in Figure 1 with ICT classes [6], they made an extension of it. We have proven this fact by visiting most of the centres in the province of Huesca and found that 90% of this distribution was followed in classroom placements:

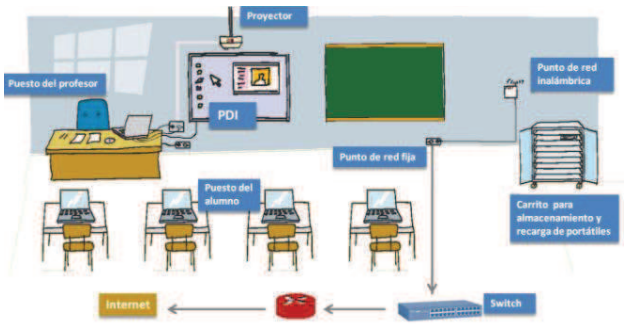


Figure 1. Traditional Classrooms with ICTs

This distribution does not facilitate collaborative individual work and teamwork. Note that students should place the tables in working groups and the teacher predetermines the choice of peers. A primary teacher of the CEIP Grañen Santiago Apostle, Huesca, points to the need for clustering, not in terms of pupils levels, but according to the duration of the Tablets batteries and the Laptops location (near to the outlets).

Workspace and equipment

The incorporation of ICT in primary education has led to the development of a new system of organization in the classroom [7]. The endowment of this project, through the research institute, has been for a class of 16 students and two teachers, one of them a tutor and a specialist in English language.

We have included this material to improve and update the allocation of resources in the classroom, in order to establish a fully ICT workspace where students can carry out their activities with updated materials that will eventually be easily accessible for schools. In Figure 2 we show the equipment we have, and the definition of the task and the distribution within the classroom that the teachers described. In the tables of the teams we found three areas:

- Zone 1: In the red box, there is a television touch area for brainstorming and collaborative work of team members.
- Zone 2: In the green box, there is only a tablet for the personal workspace of each student.
- Zone 3: The orange area is the meeting point for the collaborative work of the teams. It is the central panel where students share experiences and the teacher can explain activities by the appropriate method.

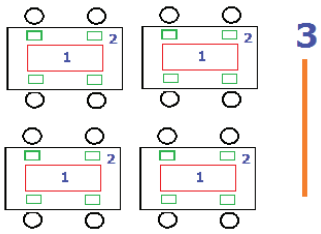


Figure 2. Classroom Distribution

Once the classroom was structured and after spending some time in it, while students' work and share experiences with their team members and classmates, we define which behaviour and actions they perform, and obtained new educative patterns related to the use of ICT and Distributed User Interfaces (DUI) [8].

Identified Patterns

The design model and the definition of the patterns in this system are important, as we found on the one hand the educative patterns responsible of the teaching learning process and, on the other hand, the patterns of definition of the roles and actions that team members have and can perform "exchange of information" these patterns can be done through the drag and drop motion between different devices.

- Share mate: the student can send information to one of his team members, share it with him and perform the educative task together.
- Request team member: The user requests contribution, through DUI of one or various members of his team members. He will share his device with all at once.
- Send doubt: This pattern is defined for the team members to submit their questions to the central panel, explain above in "zone 3".
- Sharing: In this pattern the user with the appropriate role in the team coordinates all the work for the team.
- Division of labour: The teacher starts the system with a division of labour between the teams. This pattern is used because of the level of the students and also in order to undertake a division of labour among peers.
- Send activity: Sending the teacher activities involves a series of steps. After the students send the first job done the system is responsible for joining the work of the group members together.
- Control: The teacher has full control of the system and is able to model students work. From his mobile device he can access any device or display and modify its contents,

as well as to allow him to determine tasks, leading to the development of the students.

We also found that other patterns that structure the educative content are implicated in the development process of the activity [9], each educative content will has:

- File: contains the educative content
- Title: This is the initial tab of the student work;
- Initial evaluative tab: This tab will also include the teacher's initial statement of practice in terms of all the materials the students need, in addition to structuring the working groups and team members beyond the distribution of tasks and roles in the team.
- Final evaluative tab: This tab of the system generates a shortcut to the students tabs so the teacher can conduct an evaluation and determine the qualification to be given as a result of achieving the objectives and evaluation criteria established for the job.
- Conclusion: The end tab, where members of the group make their findings, is capable of binding to the Start-up tab of another group.

The internal structure of the overview pattern is free; the student will structure his information, as appropriate, for each activity.

SYSTEM ARCHITECTURE AND FUNCTION

The system is developed using Cloud Computing [10].

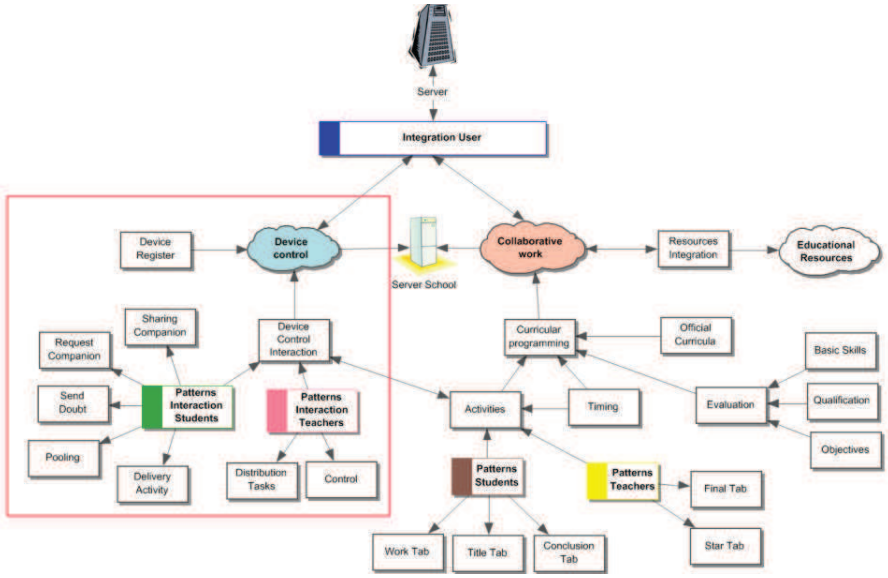


Figure 3. Proposed Architecture

This architecture divides the system into three clouds that serve different needs, as shown in Figure 3. First, the user integration layer in blue from which we have developed the Web Services that allow access to users, groups and school schedules. This is a layer in which the data are only consulted, because it is supposed, according to Spanish laws [11], the Educational Department, of each community, manages the user maintenance.

The Device Control cloud, in green, is responsible for managing the devices, it registers them in the system once the application starts, and performs the interaction patterns between different devices, i.e. the user activates the pattern that will work to exchange information with his colleagues. This is a process by which the object is passed to the partner as a copy in shared memory. It means that the two users can interact with the same object and take action on it, add, delete and modify. However, the user can launch the object as read-only, as this is a property of the pattern and is taken into account in the model. These patterns have classified the models of interaction for students, to match the patterns described above.

The Collaborative Work cloud, in brown, manages the system's curriculum for teachers to develop their collaborative work sessions. As we can see, compared to the elements found in the definition of collaborative work in the Device Control cloud, this cloud is in charge of information processing in collaborative business logic, where teachers can create and prepare activities for their students like chips, and students can create tabs within a workgroup.

In this last cloud, we find the schedules and organised curriculum of sessions that follow official curricula. As we can see, the worksheets (tabs) are integrated into the activities and the teacher gives content to these activities through the creation of tabs for students from the beginning. During the development of the activity, the student may need to add more worksheets, but the teacher always gives an initial structure for a collaborative activity script that serves students. In this cloud we also found a section for the process evaluation and qualification of students through basic skills. Here, teachers perform an assessment of students' work. We also find that the teacher's work is starts

and ends with a tab following the previously defined patterns. It provides access to clouds of educational resources available to public administration. We see it done by integrating these resources and performing, through web services, a parameterized search. This cloud could eventually be extended to teachers, to launch and share educational resources and activities that have worked with students, and to provide feedback to the system.

In short we can say that the system is a centralised model based on the operation of the interface for the "displacement" of objects within the system. These models facilitate the implementation, development and expansion of new collaborative patterns where users have access to multiple devices.

SCENARIOS

In this scenario we have tried to carry out a task for environmental knowledge in programming. First, the teacher performs the task by creating a desktop application. Figure 4 shows a screenshot of this activity, it shows the first screen where the teacher adds the title and general description of the activity.

Figure 4. Activity principle interface

Figure 5 shows a screen where teachers create working groups and works assigned to each of the students. As we can see in the worksheets, the teacher adds the parts each student needs to perform. Once these tabs have been added, the teacher, using the left keypad, inserts these objects and may even introduce students to the necessary materials or create a script for that tab.

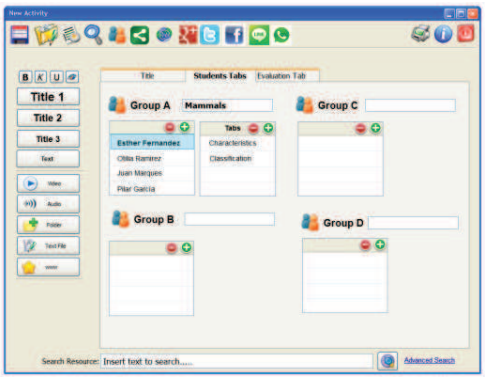


Figure 5. Groups and tabs interface

The taskbar on the left shows the objects or patterns that are added to the worksheets. As we can see, patterns correspond to text files, where teachers add content to student records. Once the teacher has prepared the statement, he starts the work session, which is loaded into the students' Tablet, who begins working with the worksheets, Figure 6.

Figure 6. Students Tablet

In this activity the student completed and populated the characteristics of mammals, which he composed from the predefined objects available in the application: video, folder, file, link, text and colour. As seen in the central part, the object content can be maximized to full screen for a larger working area. At the bottom we find the various options available from the objects in the white work area. In this way we get a system that allows students to interact with any content associated with objects in the editing area. Below the central screen sharing, where students send objects for editing and sharing, this copying of objects is done by request of the student, and modifications take place within the object or activity being carried out, as we can see in Figure 7.

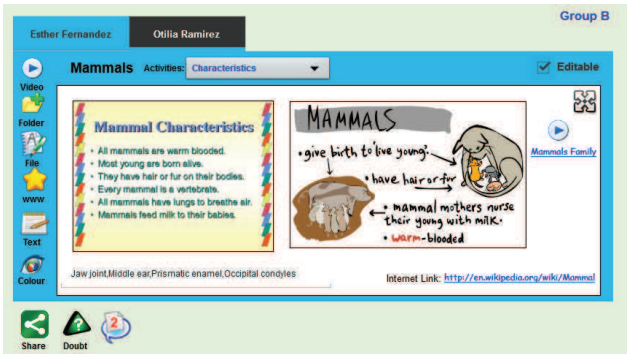


Figure 7. Common central area

Figure 7 shows how the behaviour and performance of the application is the same as the student's Tablet, but in this case it shares the central area (1), as in Figure 1 above. The

working group can share with classmates and members of other groups, send possible questions to the teacher, and also receive notifications from other collaborative groups. We can see the structure of the workspace in this area is the same, and the operations are the same.

CONCLUSION

Today, collaborative work in primary school education is a difficult task with ICT. With the development of this prototype, teachers indicate that idle members are registered, and the group dynamic is agile and closer to the traditional way of collaborative work in the classroom. During this development we considered that teachers have existing needs and determine what needs they may have in the future. Teachers tell us that students know how to use Word type applications. Therefore, the editor should be as similar as possible to avoid having to train students. We also need to add a writing tool that these requirements have obtained from the use of the application be added in the next version.

We have made a version that lacks connectivity between centres, and does not allow different members of the educational community to expand collaborative work in other centres. However, from a technical point of view this is possible if we make some changes and integrate audio and video. This application can be used for all types of group dynamics in the business world. Since the organisers of meetings have to share materials and ideas, we can then propose working meetings to exchange ideas.

ACKNOWLEDGMENTS

We appreciate the provision of materials to the research institute, and the collaboration of teachers in primary schools (John XXII, Huesca and St. James, Grañen), for allowing us to prepare the prototype within their working sessions at school.

REFERENCES

1. Sánchez, J., & Salinas, A. (2008). ICT & learning in Chilean schools: Lessons learned. Computers & Education, 51(4), 1621-1633.
2. Jamieson-Proctor, R., Watson, G., Finger, G., Grimbeek, P., & Burnett, P. C. (2007). Measuring the use of information and communication technologies (ICTs) in the classroom. Computers in the Schools, 24(1-2), 167-184.
3. Fardoun, H., Montero, F., & Jaquero, V. (2010). Designing e-Learning Systems to Support new Teaching Techniques. Journal of Computer Science and Engineering, 2(2).
4. Cipres, A.P.; Fardoun, H.M.; Mashat, A., "Cataloging teaching units: Resources, evaluation and collaboration," Computer Science and Information Systems (FedCSIS), pp.825, 830, 9-12 Sept. 2012
5. Tesoriero, R., Fardoun, H., et al., Human-Computer Interaction. Interacting in Various Application Domains. ISBN: 978-3-642-02582-2. 236-245. Pages 236-245. 2009.
6. Escuela 2.0 Project. Education Ministry of Spin. Last access: 17/01/2013. Link: http://blog.educastur.es/escuela20/escuela-20-2/
7. Slavin, R. E. (Ed.). (2013). School and classroom organization. Routledge.
8. Gallud, J. A., Tesoriero, R., Vanderdonckt, J., Lozano, M., Penichet, V., & Botella, F. (2011, May). Distributed user interfaces. (pp. 2429-2432). ACM.
9. Fardoun, H. M. (2011). Elearnixml: towards a model-based approach for the development of e-learning systems (Doctoral dissertation, Universidad de Castilla-La Mancha).
10. Fardoun, H. M., Lopez, S. R., Alghazzawi, D. M., & Castillo, J. R. (2012). Education System in the Cloud to Improve Student Communication in the Institutes of: C-LearniXML++. Procedia-Social and Behavioral Sciences, 47, 1762-1769.
11. Juan J. Martínez Sánchez. La Legislación Escolar En La Ii República Española.

Supportive User Interfaces and Task Migratability in Smart Environments

Peter Forbrig, Michael Zaki
University of Rostock, Dep. of Comp. Science
Albert-Einstein-Str. 21
D-18051 Rostock, Germany
+49 381 498 7620
[peter.forbrig|michael.zaki]@uni-rostock.de

Philippe Palanque, Marco Winckler
ICS-IRIT, University Paul Sabatier,
118 route de Narbonne
31062 Toulouse CEDEX 9, France
+33 (0)5.61.55.63.59
[palanque|winckler]@irit.fr

ABSTRACT

This paper discusses supportive user interfaces as a special kind of distributed user interfaces. It introduces tangible objects as elements of a supportive user interface and discusses the role of metaphors. Furthermore, the role of supportive user interfaces for implementing the usability criteria of task migratability in smart environments is discussed. Some challenges are identified and the combination of tangible user interfaces and graphical user interfaces is suggested.

Keywords

Distributed user interfaces, supportive user interface, task migratability, smart environment

INTRODUCTION

The general term of "supportive user interfaces" fits to nearly all interactive applications as in some way every user interface has to be supportive. As a result of the SUI 2011 workshop participants agreed on the following more specific and precise definition:

"A supportive user interface (SUI) exchanges information about an interactive system with the user, and/or enables its modification, with the goal of improving the effectiveness and quality of the user's interaction with that system." [7].

According to this definition a user interface should be distributed and adaptable in order to give the user the opportunity to interact with the system in a more appropriate way according to the specific encountered context of use. The idea of such interfaces is very much related to the "Meta-User Interface" approach [1], which has been introduced to control interactive ambient spaces.

A general definition of distributed user interfaces is given by Elmquist [5]. Additionally, five dimensions of such interfaces were identified. His definition states: "A distributed user interface is a user interface whose components are distributed across one or more of the dimensions input, output, platform, space, and time."

Our following discussion will be focused on distributed user interfaces in the context of smart environments. Some of these ideas were already presented in [10].

Existing approaches of supportive user interfaces will be discussed first. The usability concept of task migratability is presented afterwards and some aspects of tangible user interfaces are provided. Advantages and disadvantages of tangible user interfaces compared to graphical user interfaces are discussed and at the end we conclude our ideas of using tangible user interfaces as a special kind of SUI and to use SUIs for dynamic task allocation.

DISTRIBUTED USER INTERFACES IN SMART ENVIRONMENTS

Our experimental basis is a lab with installations of a smart meeting room that is equipped with a lot of sensors, projectors and cinema screens (see Figure 1). The room itself can be considered as a composition of distributed user interfaces. Bayesian algorithms try to infer next possible actions of the users and based on that information convenient assistance is to be provided.

Figure 1: Smart Meeting Room

The authors in [12] refer to the ambiguity of those systems by stating "This creates complex and unpredictable interactive computing environments that are hard to understand. Users thus have difficulties to build up their mental model of such interactive systems. To address this issue users need the opportunity to evaluate the state of these systems and to adapt them according to their needs."

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.

In this paper, Meta-UIs are suggested as a possible solution for the described problem.

A functional model and system architecture for Meta-User Interfaces for smart environments is provided by Roscher et al. [17]. The discussion focuses on the development of user interfaces that are distributed on different devices. "The Migration menu provides possibilities to redistribute a UUI (ubiquitous user interface) from one interaction resource to another, e.g. transfer the graphical UI to a screen better viewable from the users' current position. Through the Distribution menu the user can control the distribution on more fine grained levels by distributing selected parts of the UI among the available IRs."

In [18] the five features shapeability, distribution, multimodality, shareabilty and mergeability were specified. These results come originally from [1].

"1. **Shapeability**: Identifies the capability of a UI to provide multiple representations suitable for different contexts of use on a single interaction resource.

2. **Distribution**: Identifies the capability of a UI to present information simultaneously on multiple interaction resources, connected to different interaction devices.

3. **Multimodality**: Identifies the capability of the UI to support more than one modality.

4. **Shareability**: Denotes the capability of a UI to be used by more than one user (simultaneously or sequential) while sharing (partial) application data and (partial) interaction state.

5. **Mergeability**: Denotes the capability of a UI to be combined either partly or completely with another UI to create combined views and input possibilities."

Distribution is sometimes also called migratory or migratablity. Grolaux et al. discuss in [12] the migration of parts of distributed interfaces to different platforms and devices.

All these mentioned features characterize the technical properties of user interfaces in given ubiquitous computing environments. However, these features do not cover the aspect of usability that is especially important for intelligent environments. Our discussion will especially focus on the dynamic allocation of tasks (task migratability) to devices, platforms, systems and users. This aspect is a little bit more abstract than the distribution of user interfaces. It will be discussed how task migrateability can be influenced by distributed user interfaces that we call supportive user interfaces.

We will especially discuss the role of tangible Meta-UIs for this purpose. Objects become part of a distributed interface. Such a user interface can even consist of several physical objects in a 3D space.

TASK MIGRATABILITY

Task migratability is known to be one of the usability criteria of interactive systems. It specifies the possibility of transferring of control for tasks execution between user and system. "It should be possible for the user or system to pass control of a task over to the other or promote the task from a completely internalized one to a shared and cooperative venture" [9].

A lot of applications provide only a static solution that does not allow migrating tasks from devices to users or the other way round. Software developers often decide during design time which task is to be allocated to which actor.

The Importance of Task Migratability

The work of the ten authors cited the most in HCI is studied in [13]. We counted the number of times where a particular HCI principle was proposed. Task migratability is ranked the fifth among all usability principles after multiplying the counted number by a weighting factor derived from the author citation frequency.

Usually, computer scientists tend to delegate the supportive tasks to softwares running on computers. However, it is not mandatory to employ a computerized technique for the sake of providing an optimal support to resident actors. In some cases, tasks have to be performed by humans. It is obvious then that "designers need to acquire a deeper understanding of what the tasks of the users might be in certain situations and how to support their achievement." [8] Software solutions have to offer a way enabling users themselves to influence the distribution of tasks. Supportive user interfaces can help reconfiguring the applications.

Allocation of Function and Authority Sharing

When designing a (partly-) autonomous system, 10 levels of automation can be considered according to [16]. Work on function allocation such as the ones described in [11] or [3] aim at supporting the design of automation and more precisely at identifying and assessing candidate functions to be automated. Beyond that, if the use of the system is highly dynamic i.e. evolves regularly (for instance in order to handle unexpected adverse events such as malfunctions, faults, malicious attacks …), there is a need for dedicated support to anticipating evolutions and for providing adequate solutions. Papers such as [1] propose a model-based tool-supported approach for the design and development of distributed user interfaces in the context of highly dynamic complex systems requiring repetitive and systematic activities to be allocated to the system in order to allow operators to be focusing on more analysis and decision related tasks. Such approach embeds automatic generation of distributed user interfaces allowing operators to monitor the execution of semi-autonomous procedures. Addressing function allocation correctly might have a huge impact on the usability especially in smart environment when quite often automation is discovered through usage. Bad designs might lead to so-called automation surprises having potentially disastrous impact on usability and possibly safety [15].

Task Migratabilty in Smart Environments

From our point of view task migratabilty is especially important within the domain of smart environments.

However, at the moment this has not been sufficiently discussed in the literature. The main tenet of smart environments systems is to support users as much as possible. Sometimes such systems are detailed designed and implemented for specific applications. However, most of the reported approaches work with Bayesian techniques that need training data. These data are provided by sensors reflecting the physical changes within an environment.

There are approaches like [17] that allow users to configure the environment. However, the concept of Meta-UIs is not directly related to task migratablity between users and systems. Tasks are often only migrated between different devices. In this way the allocation of tasks between users and system remains the same. However, this technology can of course be used to influence task allocations to users as well. Supportive user interfaces for smart environments could be a solution. We strongly believe that this approach can improve the usability of smart environments. Preliminary experiments within smart meeting rooms support this hypothesis. However further experiments have to be performed to thoroughly evaluate the approach.

TANGIBLE USER INTERFACES

The idea of tangible user interfaces is not new. There is an ongoing research to use such interfaces in conjunction with table top systems. There is the intention to couple the different worlds of our reality as 3D space and the world of the computer. In [14] it is called coupling of bits and atoms.

In a smart ambience, the whole environment can be considered as a tangible user interface. Moving objects around like putting a notebook on a table can change the state of the environment's software. In the described example, a projector could connect to the notebook and present the slides on the computer. Even moving around as a human can be considered as an interaction. In our example with the notebook, the slides of that person standing in the room's presentation zone can be presented. In case this person leaves and another one enters the zone, the room acts then accordingly.

Our smart meeting room lab has an explicit tangible user interface. The tangible object is a pencil that is used in conjunction with a white board and a cinema screen. The white board is not visible when the screen is down. If the pencil is in a box, no white board is needed and the cinema screen can be down and be used by a projector. When the pencil is taken out of the box the screen has to go up to make the white board accessible. In case a projector was presenting on the screen, this projector has to be shut down.

The behavior is triggered by sensors in the box. However, from the point of view of the user, the pencil seems to be responsible for the behavior. The pencil however represents the tangible user interface and allows interacting with the environment. If the whole room is considered as a user interface for the smart environment, the pencil can then be seen as a very simple supportive user interface as it allows configuring the smart meeting room.

From our point of view, it may be very beneficial to consider real objects as parts of a distributed user interface, so that they can play the role of supportive user interfaces. Real physical objects can be used to express the kind of support that is appreciated by the users. In that way, explicit support is given by the users to the system to identify the current situation in detail. It might be difficult for the meeting room to identify whether a brainstorming meeting, a conference or a business meeting is taking place in the room. Even some sub-states such as meeting where full support, medium support or no support is required can be determined.

For this purpose a tea pot can be used. When it is placed on the meeting table, a brainstorming meeting is performed with a minor support. If there are two tea pots on the table some more is appreciated. Three tea pots specify that the maximum available support should be provided.

A tea pot placed on the side board signals that a conference is taking place in the room. If a tea pot is placed on a small table, a business meeting then is in progress.

In this case, the tea pots play the role of the supportive user interface. Their location configures the provided kind and level of support.

The consideration of real objects as distributed user interfaces and the concept of task migratability raise a lot of interesting questions.

1. Is it possible to decide in which situations it is better to use existing objects in the environment or to use new ones?
   - Pro existing: More convenient
   - Con existing: Objects might be used to drink tea or in order to reconfigure the system
   - Pro new: Less probability of unintended usage
2. Is it better to use one object at different locations or different objects at one location?
   - Pro one: More convenient
   - Con one: Difficult to remember
   - Pro several: Better to remember
   - Con several: Too many objects
3. Should existing metaphors be favored or new ones introduced?
   - Pro existing: Better learnability
   - Con existing: Confusion with traditional usage
   - Pro new: Clear interaction
   - Con new: Difficult to learn

There will be no general answers to those questions. However it would be interesting to discuss pros and cons with participants of the workshop and to identify some criteria that help to analyze application domains

TANGIBLE UIs VERSUS GUIs

It is of course not possible to express all necessary information that should be provided to an environment by tangible user interfaces. They are helpful only if a limited state space exists and one of these states has to be specified. The level of support or the limited number of presentation styles can be expressed easily.

However, it is not obvious when this is really convenient.

We identified different patterns for presentations like presentation with one projector, presentation in a sliding window mode with all projectors, presentation of the outline with one projector and presentation of the current slide with another one, presentation of the outline with one projector and a skidding window with three other ones, etc.

This can be expressed by a tangible user interface for users often working with the room. However, somebody new to the room would like to have a GUI to express the pattern for the presentation. Sometimes a certain file has to be selected for presentation. For this purpose tangible user interfaces are not helpful. However, if certain profiles are predefined the selection can be done in a tangible way.

A really broad variety of support is possible. Everything is configured with the help of a GUI like in [17] and [18], predefined configurations are selected by tangible objects or there is a solution using both ideas of a GUI and a tangible user interface. It seems to be really important to study the different approaches in detail and to consider also providing different kinds of support in parallel. In this way users can decide which kind of interaction they prefer. However, the unintended use of objects remains problematic.

Experiments will provide more insights for this interesting topic. However, we already truly believe that task migratability is really an important success factor for smart environments.

CONCLUSIONS

In this paper we argued to consider task migratability as an important aspect for smart environments. At the moment this aspect plays only a minor role in the scientific discussions concerning the implementation of smart environments. We believe that the acceptance of the concept of smart environments may increase if the user is able to influence the dynamic task allocation in a convenient way.

The paper introduced tangible user interfaces for smart environments as a special distributed user interface and a subset of supportive user interfaces. Tangible user interfaces can help to explicitly inform the environment about the current intentions of the user. Thus, implicit interactions in smart environments have an explicit influence. Also, future experiments need to show that the combination of tangible and graphical user interfaces can improve the usability of smart environments. In both cases task migratability has to be taken more into account.

During the workshop we would like to provide some examples for dynamic tasks allocation in smart meeting rooms and discuss their usefulness with the participants. We hope to get feedback from the participants concerning the design of tangible and graphical user interfaces.

REFERENCES

1. Barboni, E, Martinie C., Navarre, D, Palanque, P, Winckler, M. Bridging the Gap between a Behavioural Formal Description Technique and User Interface Description Language: Enhancing ICO with a Graphical User Interface Markup Language. Journal of Science of Computer Programming Vol. 78, 2013. (to appear).
2. Blumendorf, M. Multimodal Interaction in Smart Environments A Model-based Runtime System for Ubiquitous User Interfaces. Dissertation, Technische Universität Berlin, 2009.
3. Boy G. Cognitive Function Analysis for Human-Centered Automation of Safety-Critical Systems. Proc. of ACM SIGCHI conference on Human Factors for Computing Systems 1998: 265-272
4. Coutaz, J. Meta-User Interfaces for Ambient Spaces. In Proc. of the 5th Int. Ws. on Task Models and Diagrams for Users Interface Design: TAMODIA 2006, pp 1-15, Springer LNCS 4385.
5. Elmqvist, N.: Distributed User Interfaces: State of the Art, Proc. DUI 2011 CHI workshop, p. 7 -12.
6. Demeure, A. Lehmann, G., Petit, M. and Calvary, G. (Eds): Proceedings of the 1st International Workshop on Supportive User Interfaces: SUI 2011 Pisa, Italy, June 13, 2011, http://ceur-ws.org/Vol-828/.
7. Demeure, A. Lehmann, G., Petit, M. and Calvary, G. SUI 2011 Workshop Summary Poster, in [5]
8. Dittmar, A. and Forbrig, P., Selective modeling to support task migratability of interactive artifacts, Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction – Interact 2011, Lisbon, Portugal, Springer-Verlag, Volume Part III, ISBN 978-3-642-23764-5, p. 571-588.
9. Dix, A., Finlay, J.E., Abowd, G.D. and Beale, B.: Human-Computer Interaction, 3rd edn, Prentice-Hall, Englewood Cliffs (2003)
10. Forbrig, P.: Interactions in Smart Environments and the Importance of Modelling, Romanian Journal of Human - Computer Interaction Vol. 5 (2012) p. 1-12.
11. Fröberg, A., Eriksson, H., Berglund, E. Developing a DUI Based Operator Control Station: A Case Study of the Marve Framework. . 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.
12. Grolaux, D., P. V. Roy, P. V. and Vanderdonckt, J. Migratable user interfaces: Beyond migratory interfaces. In IEEE/ACM Conf. on Mobile and Ubiquitous Systems, 422–430, 2004.
13. Hinze-Hoare, V. Review and Analysis of Human Computer Interaction (HCI)Principles, July 2007, http://arxiv.org/ftp/arxiv/papers/0707/0707.3638.pdf
14. Ishii, H. and Ulmer, B., Tangible bits: towards seamless interfaces between people, bits, and atoms. In: Proceedings of the CHI'97 conference on human factors in computing systems, Atlanta, Georgia, March 1997, pp 234 – 241
15. Palmer, E. "Oops, it didn't arm." - A Case Study of Two Automation Surprises . 8th Int. Symp. on Aviation Psychology, 1995.
16. Parasuraman, R.; Sheridan, T.B.; Wickens, C.D. "A model for types and levels of human interaction with automation" Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Trans. on, vol.30, no.3, pp.286-297, May 2000
17. Roscher, G., Blumendorf, M. and Albayrak, S. Using Meta User Interfaces to Control Multimodal Interaction in Smart Environments, Proceedings of the IUI'09 Workshop on Model Driven Development of Advanced User Interfaces, http://ceur-ws.org/Vol-439/paper4.pd
18. Roscher, D., Lehmann, G., Blumendorf, M. and Albayrak, S. Design and Implementation of Meta User Interfaces for Interaction in Smart Environments, in [5]
19. Zaki, M., Wurdel, M., and Forbrig, P., Pattern Driven Task Model Refinement, International Symposium on Distributed Computing and Artificial Intelligence, DCAI 2011, Advances in Soft Computing, Vol. 91, ISBN = 978-3-642-19933-2 , p. 249-256.

Engineering transitions to bind distributed interaction

João Paulo Delgado Preti
Instituto Federal de Ciência e Tecnologia de Mato Grosso (IFMT)
CP 78.005-200 MT BR
joao.preti@cba.edu.ifmt.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
The allocation of interactive narrative across several devices characterizes distributed interaction, a phenomenon that is occurring naturally as the number and availability of devices to support interaction grows, even though there is still only *ad hoc* technological support for designing such systems. This paper focuses on a relevant aspect of distributed user interface (DUI) – the bridges that allow user interaction from one device to another. We have called them "*transitions*". In this paper, we characterize this concept and present some transition supports services to support transitions in DUIs.

Author Keywords
Transition; crossmedia; distributed user interfaces; SOA.

ACM Classification Keywords
H.5.2. User Interfaces: Interaction styles

General Terms
Human Factors; Design.

INTRODUCTION
Albeit digital natives naturally incorporate a myriad of communication and computer-supported alternatives into daily personal and professional life, it is noticed that the designed, purposeful distributed interaction, in order to explore the distribution synergy only recently has started to be explored.

The research field within HCI that encompasses aspects of this distribution has been named Distributed User Interfaces, DUI, which has been defined synthetically by Elmqvist [2]. The integrated use of multiple device communication has also received attention from the design of communication professionals. In this community, distributed interaction has been explored under the concept of crossmedia storytelling. A narrative distributed across different media is defined as crossmedia [4].

In this paper, we address one aspect that is common to both fields: the **bridges** that allow interaction to happen in more than one device, either sequentially or simultaneously. In this paper, we set foundations on crossmedia interaction design and on DUI applications to characterize these bridges, which we name "transitions" and present experiments that supports the development of applications using transitions by distributed input and output.

Our results come from a three-year research project funded by Fapesp and Microsoft Research to understand and develop distributed interaction. We believe that the current state-of-the-art does not allow for standardization of transitions. There is still no "winning design" nor toolkits that can be used for distributed interaction. There are very few experiences of development frameworks and middlewares. We also believe that the lack of technological support for applications to conveniently explore the potential of DUIs synergy has limited the potential of commercial crossmedia applications and marketing campaigns. We expect that our discussion about transitions can help in the development of better tools.

TRANSITION MECHANISMS PROPERTIES
In a previous work of our research group we have identified transition characteristics by analyzing twenty five market and academic crossmedia applications samples, which included movies, TV series, games, advertisement campaigns, magazines, newspapers, iTV, Web and mobile applications. From these samples, we have extracted the transitions and worked on their description. A C# framework entitled X-Gov was also created to allow the development of government crossmedia applications [4].

We have identified seven transition mechanisms properties: cross-device movement, temporal coordination, call-to-action, data transportation support, engagement, feedback and privacy.

We also advocate that a technological framework that integrated transition technologies and offered designers a simple interface for using the transition mechanisms in their applications could leverage the concept of DUI. The literature points to the existence of some crossmedia and DUI frameworks: [1], [3], [5], [6], [7]. We notice that there are solutions for the composition, adaptation, management and distribution of content, but none of those known to us has focused in transitions.

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

DISS: A MIDDLEWARE FOR SUPPORTING TRANSITIONS IN DISTRIBUTED USER INTERFACES

The creation of the crossmedia framework was a starting point to investigate and define some requirements of a distributed interaction and to theorize about the transition concept. This research helped us realize the need of an infrastructure that supported DUI application development, like our own framework, or applications that need to realize a distributed interaction without the framework, applications that don't want to extend a .Net platform or that don't want to inherit government requirements, for instance. The designed infrastructure was a middleware extension of a Service Oriented Architecture (SOA).

Distributed Interaction Support Service (DISS) is a service that intermediates the communication between devices. DISS specifies a set of necessary services to allow the coordination between multiple devices sequentially or simultaneously and implements a subset of the framework requirements stated in a previous work.

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. DISS is being built on top of DPWS (Device Profile for Web Service) specification as architecture base for discovery and communication between devices. DISS is composed of the following services: Interaction Modality Mapping Service (IMMS), Device Management Service (DMS), Context Support Service (CSS), User Management Service (UMS), Content Conversion Service (CCS) and Transition Support Service (TSS), this last one the focus of this paper.

Transition Support Service (TSS)

In our middleware, TSS is responsible for allowing the users to continue their tasks into another device or transfer the task to another user to continue the task that was being held. Three types of transitions need to be supported in our middleware.

The first one is the input transition, when a user wants that other users interact with UI components, while the first one keeps the task management. For example a user requesting another one (in another device) to complete an input text, or to select an item of a list or even provide an image or an audio stream.

The second one is transition between devices, when users are directed to another device in order to achieve the required steps in pursuit of their goals. For example, a scene in which the user wants to continue the interaction on the TV to the phone could occur via QR Code, even though this strategy is not sufficient to transmit the entire interaction state that the user had performed on TV.

Last one is transition between users, when it is necessary that other user carry out a task already begun by another user. This service is responsible for migrating a user session to another, keeping the original state of the interactions previously performed. This service also needs to know the history of actions performed and place the new user in the last step performed by his predecessor.

From the technological point of view, several platforms are currently required to support transitions. Despite the technological convergence has simplified the task of engineering, different protocols and services are necessary to integrate different types of transition into an application. TSS defines a set of web services to allow transitions in an open way. The paragraphs that follow show what has been implemented and tested about the first of the three types of transitions and what still needs to be done.

The used tools were JMEDS and DPWS Explorer of WS4D. JMEDS implements DPWS Stack and DPWS Explorer works as a client to navigate and interact with devices in the network. Java 1.6, Android SDK, Eclipse Indigo, Primefaces 2.2 and Tomcat 7 were used to develop and deploy the application. SOAPUI 4.5.1 and Wireshark 1.6.0 helped us debugging the application to understand messages exchanged between devices. These tools were executed on Mac OS X Lion and Linux Ubuntu 12.04. The browser used was Firefox version 15 and the devices were Samsung Galaxy S3, Samsung Galaxy N7000 and Nexus 7.

A scene in which Alice Desktop asks remotely for Bob Smartphone to interact with an UI component that Alice is in doubt about what to do, type or choose was defined. Figure 1 describes in detail this scene for a web browser client.

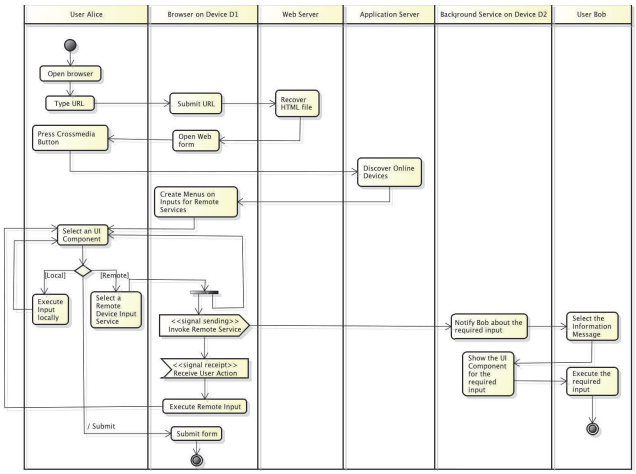


Figure 1. Desktop browser requesting remote input to a smartphone.

One concrete scene for Figure 1 is a user that requests from a desktop browser a keyboard remote input to another user with a smartphone device. The browser used was Firefox with a custom Firefox extension created to invoke a SOAP service in an application server with DPWS stack implemented. The server service has the responsibility to maintain a list of connected devices in the network, the device that wants to participate in the distributed interaction

needs to be in the same local network or use a proxy. The referred SOAP service gets all available devices on the network and returns a list of devices in a XML structure that is parsed by the browser extension to build the popup menu for each input in the HTML web page.

As can be seen in Figure 2, a popup menu with a list of devices and operations is shown when *onmouseover* event is triggered. Before that, the user needs to click in the image button below, which will execute the Firefox extension mentioned previously.

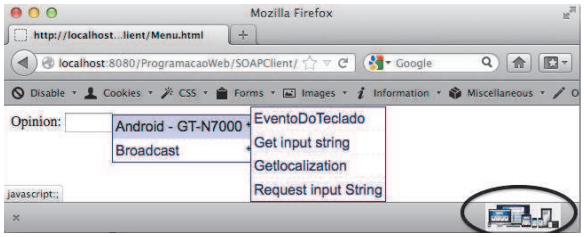


Figure 2. Textfield options for remote input.

When the user requests a remote input, a notification is launched in the smartphone device as can be seen in Figure 3. An application, named XDevice, was created to implement the required services on Android platform. If the user selects icon to show notifications, a "Keyboard Request" notification presents the request message.

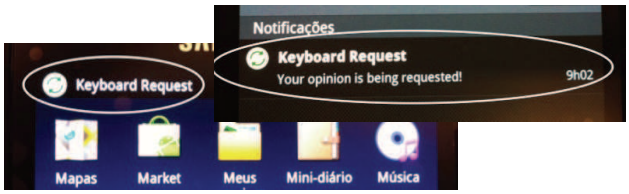


Figure 3. Cross-device transition notification.

Selecting the "Keyboard Request" notification displays an input text field with a *qwerty* keyboard expecting the user remote interaction (Figure 4). Meanwhile, the browser is expecting the input text to be sent back.

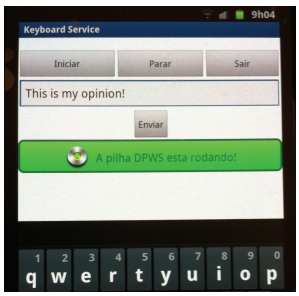


Figure 4. Response to cross-device request.

Browser implementation faced many problems due to browser restrictions. The Access-Control-Allow-Origin HTTP header must be set in all web server responses. Invoking web services with javascript demands hard code and is prone to errors. Javascript also does not support threads and DPWS stack, which require server side implementation or a plugin browser that supports it. Ahead in this paper a better solution is presented by implementation of a server side web component (Figure 5).

The same scene without the browser, with just Android applications, was easier to handle. The web browser and application server cease to exist because DPWS Stack is implemented in both devices and avoid browser security problems and Javascript restrictions.

These experiments helped modeling a state diagram for DUI components that supports this kind of transition (Figure 5). The component is ready after getting all information about devices in the network and only does this operation again if a parse error occurs in the moment of creating the popup menu (considering an error in the returned devices list) or in case of a network error in invoking the selected operation (considering the device may not be online anymore).

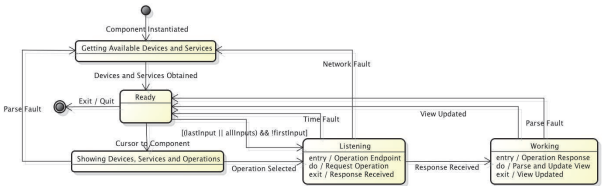


Figure 5. Generic State Diagram for DUI Components.

The guard condition [(lastInput || allInputs) && !firstInput] encompasses the diversity of scenes, as in Table 1.

S	R	Scene
1	0	Just push a message, not a request.
1	1	Typical case.
1	N_l	Remote user can submit continuously, but only the last input is valid.
1	N_a	Remote user can submit continuously and all inputs are valid.
N	1_f	Multicast request, only first response of all devices is valid.
N	1_l	Multicast request, only last response of all devices is valid.
N	M_{fd}	Multicast request, only first response of each device is valid.
N	M_{ld}	Multicast request, only last response of each device is valid.
N	M_a	Multicast request, all devices can submit continuously and all submissions are valid.

Table 1: Solicitation (S) / Response (R) Scenes

We tested the subset {1:0 , 1:1 , 1:N_l , 1:N_a} successfully. The subset {1:N_l , 1:N_a} were achieved by using WS-Eventing present in the DPWS Stack with two Android devices. For the broadcast request set {N:1_f, N:1_l , N:M_{fd} , N:M_{ld} , N:M_a} a pool thread implementation is needed. We are investigating the possibility of using Constrained

Application Protocol (CoAP), an emerging protocol defined by IETF CoRE WG that can create an alternative to HTTP RESTful APIs on highly resource-constrained devices, using HTTP over UDP instead of TCP with rudimentary reliability features. It can save a lot of protocol overhead that inherently comes with TCP. jCoAP is the CoAP Java implementation being used to realize these broadcast tests.

Extending UI components of MVC frameworks to support the state diagram of Figure 6, can help in the rapid build of cross platform applications with distributed interaction, addition to eliminating the need of a browser extension or plugin. We have extended two JSF UI components (inputText and GMap), one for remote input and another to point the geographical location of all DPWS compliant devices in the network. All devices must be configured to use the same proxy so that they don't need to be in the "same" network. The web server listens to every entry of a DPWS device compliant in the network and registers its location so that web components can obtain a list of available devices and respective services and operations. This implementation differs from Figure 1 because there is no need of implementing a discovery service for the JSF framework extension that acts as a DPWS client, using the middleware native discovery service.

Another experiment was a scene with the use of data streaming in which user Alice Desktop wants to capture the audio of the user Bob Smartphone. Figure 6 represents Bob's phone informing him that someone has requested his audio stream. XDevice application starts automatically to capture the audio, needing an intervention of Bob in case he doesn't want to. No audio compression was done (PCM audio format) and tests occurred in a 10/100 wifi network with an audio delay of about 6 seconds.

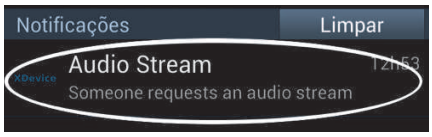


Figure 6. Notification that the audio is being captured.

The experiments presented helped us in the definition of 17 services and 49 operations that can be seen in Figure 7. Many of the designed services has an *registerAsObserver()* operation, which allow a client to be notified about a service event, for instance, a client can be notified from every keyboard triggered event of a remote device.

CONCLUSION

We believe that the middleware that we built can help programmers to develop better DUI applications. Because of the many different mechanisms that work as transitions, we understand that the most important research action is towards standardization of transitions. From the interaction design point of view, a common language for DUI transitions is still missing. We advocate that this language should be based on a deeper user research to develop an intuitive set of seamless transitions.

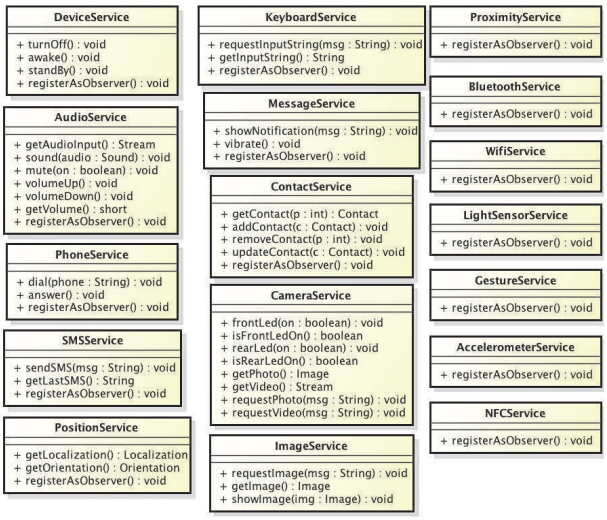


Figure 7. Device Services.

ACKNOWLEDGMENTS

This work has been partially funded by grants CAPES and Fapemat.

REFERENCES

1. Bardram, J., Doryab, A. and Gueddana, S. 2011. Activity-Based Computing – Metaphors and Technologies for Distributed User Interfaces. *DUI 2011* (Vancouver, BC, May. 2011), 61–64.

2. Elmqvist, N. 2011. Distributed User Interfaces: State of the Art. *DUI 2011* (Vancouver, BC, May. 2011), 7–12.

3. Elmqvist, N. 2011. Munin: A Peer-to-Peer Middleware for Ubiquitous Visualization Spaces. *DUI 2011* (Vancouver, BC, May. 2011), 17–20. DOI=http://dx.doi.org/10.1007/978-1-4471-2271-5_8.

4. Filgueiras, L.V.L., Correa, D.O., Neto, J.S.O. and Facis, R.P. 2008. X-Gov Planning: How to Apply Cross Media to Government Services. *Digital Society, 2008 Second International Conference on the* (2008), 140–145. DOI=http://dx.doi.org/10.1109/ICDS.2008.29.

5. Manca, M. and Paternò, F. 2011. Distributing User Interfaces with MARIA. *DUI 2011* (Vancouver, BC, May. 2011), 93–96. DOI=http://dx.doi.org/10.1007/978-1-4471-2271-5_8.

6. Melchior, J., Vanderdonckt, J. and Van Roy, P. 2011. Distribution Primitives for Distributed User Interfaces. *DUI 2011* (Vancouver, BC, May. 2011), 29–32. DOI=http://dx.doi.org/10.1007/978-1-4471-2271-5_8.

7. Ng, K., Badii, A., Sailor, M., Ong, O., Neagle, R. and Quested, G. 2006. AXMEDIS Programme and Publication Tools Integration with Workflow-Enabled Communication and Process Control. *Automated Production of Cross Media Content for Multi-Channel Distribution, 2006. AXMEDIS '06. Second International Conference on* (2006), 110–120. DOI=http://dx.doi.org/10.1109/AXMEDIS.2006.50.

Challenges on Distributing a Collaborative Sketching System Across Multiple Devices

Ugo Braga Sangiorgi, Mathieu Zen, Vivian Genaro Motti, Jean Vanderdonckt
Louvain Interaction Laboratory, Université catholique de Louvain
Place des Doyens, 1 - B-1348 Louvain-la-Neuve (Belgium)
{ugo.sangiorgi, mathieu.zen, vivian.genaromotti, jean.vanderdonckt}@uclouvain.be

ABSTRACT
The recent popularization of touch screen devices have brought to users the opportunity to use different devices to interact and to share content, while current advances in the mobile context brought new capabilities for systems to run on many devices while maintaining the system's consistency. Those two factors combined pose new opportunities for researchers to explore how users can collaborate using an heterogeneous set of devices, that can include large tabletops, smartphones or e-readers. This paper starts the discussion on four challenges related to this context.

Author Keywords
Electronic sketching; Distributed systems; Collaborative design; Prototyping.

INTRODUCTION
The current popularization of touch screen devices and the multi-platform capabilities made possible by HTML5 might pose new opportunities for developers to build distributed interactive systems with minimum effort on adapting systems for each platform. Systems to support design activities are also included in this set of new opportunities, also giving room for researchers to investigate how designers use sketching to prototype interfaces on the current multi-platform scenario.

Multi-platform sketching is the activity of drawing with an electronic stylus at different devices while having the same system running on those different devices [13] and developed the GAMBIT system, a multi-platform collaborative tool originally conceived for User Interface design that allows users to share pictures and sketches on different devices using a virtual, spatially infinite wall.

The tool is an essential part of a research on sketching, whose goal is to investigate electronic sketching usage in current UI design practices taking into account the multi-platform context for producing and validating interactive prototypes. The system was created to be:

1. Sketch-based - electronic sketching is supported as the main mode of interaction, it is used to quickly put ideas on an external medium, where they can be discussed, improved and stored for further reference;
2. Distributed - for it allows users to sketch using the device of their preference, and also allows the prototyping and testing of systems on the very device it is intented to run. The system was built with HTML5 and Javascript in order to run on any device with browsing capabilities, through a browser or embedded into a native application;
3. Collaborative - for it focus on group sessions, allowing not only designers to sketch and discuss together, but also to include end users in the process.

This paper is organized as follows: in the next section we motivate the research on distributed user interface together with the related works. Then we present the Distributed sketching system constructed to investigate sketching activities on a multi-platform context. And finally we present a set of challenges related to distributing a collaborative system into multiple screens.

MOTIVATION AND RELATED WORK
Although the motivation of this research work is UI development, the observations presented here can scale up to sketching systems and distributed systems in general.

To support sketching into UI design, we needed to analyze the process in which UI design is included. The tools available for UI development are usually not focused on UI **design**, in which designers usually explore different alternatives but in UI **modeling** as a final product, where designers must attend to formal standards and notations. There are many tools available for both modeling and design, however practitioners are currently forced to choose formal and flexible tools. Whichever they choose, they lose the advantages of the other, with attendant loss of productivity and sometimes of traceability and quality.

As categorized by [3] and depicted in Table 1, a Distributed User Interface (DUI) is a user interface whose components are distributed across one or more of the dimensions input, output, platform, space, and time, and can be described as follows:

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

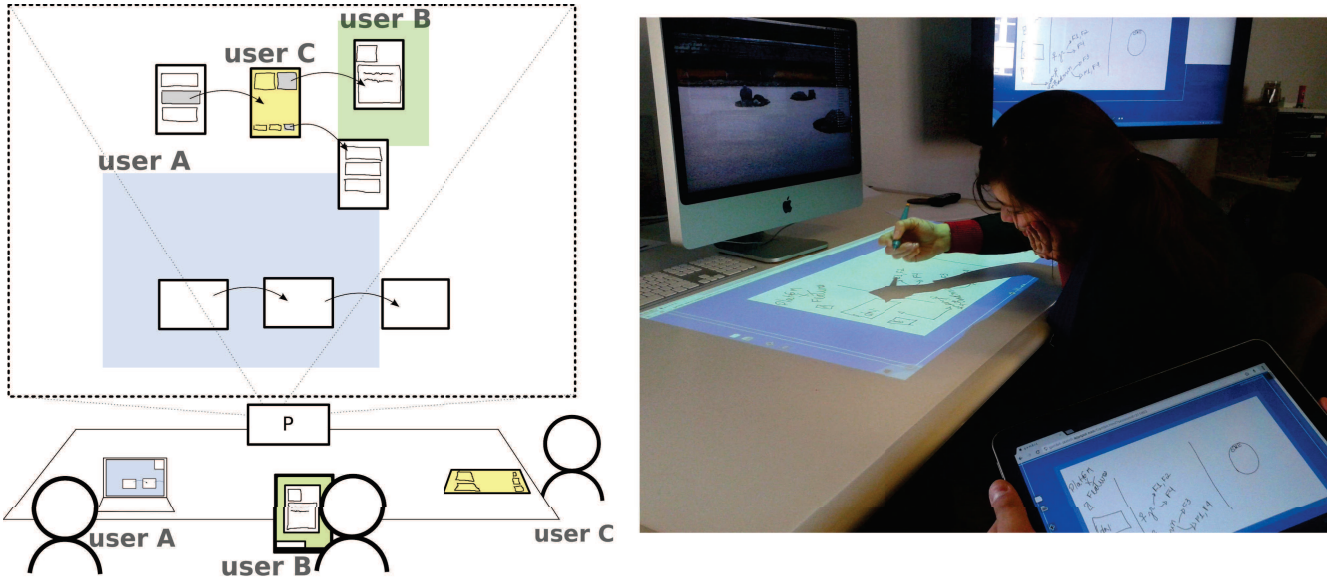


Figure 1. GAMBIT, a multi-screen distributed sketching system showing, in the right part, a session shared with tablet (front), an interactive tabletop (middle) and a TV (back).

Input: Managing input on a single computational device, or distributed across several different devices (so-called input redirection);

Output: Graphical output tied to a single device (display), or distributed across several devices (so-called display or content redirection);

Platform: The interface executes on a single computing platform, or distributed across different platforms (i.e., architectures, operating systems, networks, etc);

Space: The interface is restricted to the same physical (and geographic) space, or can be distributed geographically (i.e., co-located or remote interactive spaces);

Time: Interface elements execute simultaneously (synchronously), or distributed in time (asynchronously).

Tool	Input	Output	Platform	Space	Time
Calico	One	One	One	Single	Sync
DENIM	One	Many	One	Single	Sync
DAMASK	One	Many	One	Single	Sync
I-LAND	Many	Many	One	Multi	Async
TEAM STORM	Many	Many	One	One	Sync
WallShare	Many	Many	One	Multi	Async
Dazzle	Many	Many	One	Multi	Async
CrossWeaver	One	Many	One	Single	Async
InkKIt	One	One	One	Single	Sync
SketchiXML	One	One	One	Single	Sync
GAMBIT	**Many**	**Many**	**Many**	**Multi**	**Async**

Table 1. Tools for UI design by sketching according to DUI systems classification.

CALICO [8], INKKIT [11] and SKETCHXML [2] are categorized as single input, single output, single platform systems despite being created for sketching in general, cannot be effectively used for designing multi-platform systems. DAMASK [6], DENIM [9], I-LAND [16], TEAM STORM [5], CROSSWEAVER [15], WALLSHARE [18] and DAZZLE [10], despite having the possibility of distributing the output across many devices, are single-platform, which means that all the devices need to run the same computational environment.

GAMBIT is positioned as a distributed system that not only can have input and output from many devices, but also from many platforms, ranging from desktops and interactive televisions to smartphones and e-readers. In this sense, devices that are better suited for sketching input can be used for pen interaction while large displays can be used for visualization. Also, a session can have parts of the shared "wall" interface distributed in space among many projections, composing a virtual meeting room, for instance.

DISTRIBUTED SKETCHING SYSTEM

Sketching is considered to be a powerful tool for doing design. As the findings of [4] point out, the presence of ambiguity in early stages of design broads the spectrum of solutions that are considered and tends to deliver a design of higher quality. Furthermore, the very process of sketching and discussing is largely considered to have the same importance as the final product of a meeting or design session [1].

Fostering creativity is considered to be important since design is essentially a problem of *wicked nature*, i.e. the process of solving it is identical with the process of understanding it [12]. In wicked problems, the designer does not have a clear understanding of what to produce and has only a vague goal in mind in the beginning.

Design sessions were observed in two companies related to user interface development in Belgium. The people involved on those sessions were designers, project managers, programmers and frequently stakeholders. In overall, in these companies the design sessions are usually done around a central topic, about which people discuss in order to produce some artifact, usually a report with a list of requirements, wire-

frames and some session log of the decisions made around the interaction. It is important to note that this report is not produced at the site but after the meeting, for what people usually take pictures for remembering and registering what was discussed. Nevertheless, the design sessions most often proceeded with three distinct phases:

1. Sketch production - One or more participants produce sketches to express ideas.
2. Sharing: the participants normally share the drawings using a big sheet of paper and use post-its. The sheets are arranged as a storyboard on a wall for discussion.
3. Discussing: the participants refined the sketches based on what was discussed and learned on the discussion.

By creating a distributed sketching system we are willing to offer designers a flexible solution in which they can choose their device of preference to sketch. GAMBIT's basic requirements list was presented in [13] and it is developed as depicted on Figure 1: the many input devices can be tablets, mobile phones, large graphical tablets, etc. and they be used to sketch and submit drawings to a virtual wall where all the drawings are organized spatially. The roles of the devices are interchangeable – a user might request the wall's control at any time, organizing and grouping the sketches, or even creating the possibility of drawing "out of sight" (i.e. out of the public projection) and then putting the drawing for public discussion, like in the sessions observed at the IT companies.

Since GAMBIT is a web-based system, the wall might be a full-screen browser window opened on a desktop computer, a projection (P in Figure 1) or a large interactive display. Figure 1 (left) also shows *user A* visualizing a big part of the screen on its own laptop, while *user B* is focused on sketching a document on the upper part of the wall and *user C* is navigating through a prototyped interaction path.

CHALLENGES AND FUTURE WORK

GAMBIT was originally conceived as a tool for aiding UI designers to produce interactive prototypes, however it poses as an example of a general-purpose system for sharing pictures and drawings in a virtual multi-screen environment. We introduce a set of four general challenges for this distributed sketching system:

C1 *Make users aware of each other's activities* if, at one hand we have the "out of sight" sketching activity, in which users want to sketch outside of the group for later discussing in public, on the other we would like to make users aware of who is sketching in public. 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.

C2 *How to "point" at something remotely?* We have made some initial observations of people sketching in groups, and we have noticed the "point to" gesture were ubiquitous in all of them: the physical communication have a greater power to deliver messages than the electronic mediated one. We have observed that remotely located users (aided with video conference applications) had difficulties when trying to point to some part of the screen, for what they used to draw upon.

C3 *Concurrency and Conflicts*: This is a classic problem of collaborative systems which we need to address and yet there is no implementation on the system to deal with the concurrent modifications and conflicts.

C4 *The right tool for the job*: Not all the devices have the same resolution or performance. We need to identify which devices are suitable for the basic activities (sketching, sharing and discussing) but also for specific activities, such as handwriting [14] [7]. Also, experienced sketchers such as architects may have different requirements than UI designers and software engineers.

The challenges posed by the novel approach presented on this paper are not completely new for Distributed User Interfaces or Collaborative Work domains, except perhaps for C4 which is specific to Electronic Sketching. C1 is discussed in [1] (in the chapter "I Know What I See, But What Do You See?"). However, classic collaborative systems are largely considering just one device (usually big tabletops) and not many devices at the same time, which might increase the problem complexity.

C2 is well covered in [17], which proposes to use a camera to capture users' arms and project a sort of shadow on the other devices. However, that would require a camera pointing at each device, and for sake of simplicity a mechanism such as a virtual laser pointer could be implemented.

Due to the popularization of touch screen devices users now have an unprecedented number of options to interact and share content, and yet there are no open standards available for making applications distributable on those devices. The list of challenges presented on this paper is intended to bring the discussion of "old" problems of collaborative systems to the contemporary context.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of the QualIHM project and its support by Région Wallonne, Direction générale opérationnelle de l'E-conomie, de l'Emploi et de la Recherche (DGO6).

REFERENCES

1. Churchill, E., Snowdon, D., and Munro, A. *Collaborative virtual environments: digital places and spaces for interaction*, 1 ed. Computer Supported Cooperative Work. Springer, 2001.
2. Coyette, A., and Kieffer, S. Multi-fidelity Prototyping of User Interfaces. *Ifip International Federation For Information Processing* (2007), 150–164.
3. Gallud, J. A., Tesoriero, R., and Penichet, V. M. *Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem Series*, xvi ed. Human-Computer Interaction Series. Springer London, London, 2011.
4. Goel, V. "Ill-Structured Representations" for Ill-Structured Problems. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, vol. 14, Lawrence Erlbaum (1992), 130–135.

5. Hailpern, J., Hinterbichler, E., Leppert, C., Cook, D., and Bailey, B. TEAM STORM: demonstrating an interaction model for working with multiple ideas during creative group work. In *Proceedings of the 6th ACM SIGCHI conference on Creativity Cognition*, ACM (2007), 193–202.

6. Lin, J., Landay, J. A. J., Berkeley, U. C., and L, J. A. Damask: A tool for early-stage design and prototyping of multi-device user interfaces. In *In Proceedings of The 8th International Conference on Distributed Multimedia Systems (2002 International Workshop on Visual Computing)* (2002), 573–580.

7. MacLean, S., Tausky, D., Labahn, G., Lank, E., and Marzouk, M. Is the iPad useful for sketch input? A comparison with the Tablet PC. *EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling* (2011), 7–14.

8. Mangano, N., Baker, A., and van Der Hoek, A. Calico: a prototype sketching tool for modeling in early design. In *MiSE 08: Proceedings of the 2008 international workshop on Models in software engineering*, ACM (New York, NY, USA, 2008), 63–68.

9. Newman, M., Lin, J., Hong, J., and Landay, J. DENIM: An informal web site design tool inspired by observations of practice. *Human-Computer Interaction 18*, 3 (2003), 259–324.

10. Oehlberg, L., Simm, K., Jones, J., Agogino, A., and Hartmann, B. Showing is Sharing : Building Shared Understanding in Human-Centered Design Teams with Dazzle. In *Proceedings of the Designing Interactive Systems Conference on - DIS '12* (2012), 669–678.

11. Plimmer, B., and Freeman, I. A toolkit approach to sketched diagram recognition. In *Proceedings of the 21st British CHI Group Annual Conference on HCI 2007: People and Computers XXI: HCI...but not as we know it - Volume 1*, BCS-HCI '07, British Computer Society (Swinton, UK, UK, 2007), 205–213.

12. Rittel, H. Dilemmas in a general theory of planning. *Policy sciences 4*, 2 (1973), 155–169.

13. Sangiorgi, U. B., Beuvens, F., and Vanderdonckt, J. User Interface Design by Collaborative Sketching. In *Proceedings of the Designing Interactive Systems Conference - DIS '12*, ACM Press (Newcastle Upon Tyne, United Kingdom, 2012), 378.

14. Sangiorgi, U. B., and Motti, V. Assessing lag perception in electronic sketching. *Proceedings of the 7th Nordic Conference on Human-Computer Interaction - NordiCHI'12* (2012).

15. Sinha, A., and Landay, J. Capturing user tests in a multimodal, multidevice informal prototyping tool. In *In Proceedings of ICMI-PUI: ACM International Conference on Multimodal Interfaces*, ACM Press (Vancouver, BC, 2003), 117–124.

16. Streitz, N., Geiß ler, J., and Holmer, T. i-LAND: an interactive landscape for creativity and innovation. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '99*, no. May, ACM New York, NY, USA (1999), 120–127.

17. Tuddenham, P., and Robinson, P. Coordination and Awareness in Remote Tabletop Collaboration. *Human-Computer Interaction* (2010), 407–434.

18. Villanueva, P. G., Gallud, J. A., and Tesoriero, R. WallShare: a multi-pointer system for portable devices. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, ACM (New York, NY, USA, 2010), 416.

Combination between Multi-Agent System and Tangigets for DUI Design on several Tabletops

Yoann Lebrun
UVHC,
LAMIH-UMR CNRS 8201,
F-59313 Valenciennes, France.
CCI Grand Hainaut - PRL
360 rue Marc Lefrancq,
F-59300 Valenciennes, France
lebrun.yoann@gmail.com

Sophie Lepreux
UVHC,
LAMIH-UMR CNRS 8201,
F-59313 Valenciennes, France,
sophie.lepreux@univ-valenciennes.fr

Christophe Kolski
UVHC,
LAMIH-UMR CNRS 8201,
F-59313 Valenciennes, France,
christophe.kolski@univ-valenciennes.fr

René Mandiau
UVHC,
LAMIH-UMR CNRS 8201, F-59313 Valenciennes, France,
rene.mandiau@univ-valenciennes.fr

ABSTRACT
The paper concerns the design of DUI composed with interactive tabletops allowing users to manipulate virtual and tangible objects around these several surfaces. We propose a model dedicated to the management of distributed interactive surfaces with Multi-Agent platforms. A case study illustrates the approach used: this case study implies a traffic management simulator distributed on two *TangiSense* tabletops equipped with RFID technology.

Author Keywords
Distributed User Interfaces, Multi-Agent System (MAS), Tabletop, Tangible Object, Tangigets, RFID.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Design; Human Factors.

INTRODUCTION
Through the use of connected tabletops, it is possible to interact with several people located in remote places [11]. Indeed, the emergence of computer networks (increase data transfer, wireless network, etc.) enables the design of new distributed applications on these interactive supports.

In consequence adapted design methods and models have to be proposed. This paper describes a generic agent-based model of a Multi-Agent System (MAS) for DUI design dedicated to connected tabletops. In this model, we suppose that generic tangible objects, called Tangigets [2, 12] are the interaction supports, manipulated by the users. This model aims at taking into consideration the located and remote interactions. The located interactions do not have incidence on connected supports whereas the remote interactions allow modify the display on the remote connected platforms. The criteria used in the model to adapt the HCI are the number of connected platforms. Furthermore the MAS model allows connecting the remote platforms.

A case study is presented; it illustrates an innovative simulator allowing the distributed management of road traffic. This simulator is distributed on two *TangiSense* interactive tabletops equipped with RFID technology. These tabletops have the particularly to be tangible and not multitouch. The illustrations show these connected tabletops and the advantages of the proposed model in order to facilitate the interactions realized by two types of *Tangigets*.

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

STATE OF THE ART

Design methods for DUI domain
In the HCI field, the idea of UI plasticity was adapted to distributed interfaces [1, 4, 6, 9, 14]. One suppose that the users want to be able to move from one platform to another one without loss of coherence in the use of the application, without loss of data (e.g. they want to continue to deal with their e-mails or surf on Internet while being mobile). The Cameleon model became a framework for the modeling /transformations of the HCI [3]. Nevertheless, this model concerns only the user interfaces.

Otherwise, the technological progress implies the multiplication of interacting devices such as smartphones (Ubiquitous Computing [17]), tablets, tabletops, *etc.*

Among researches, the aim is to allow the users interact more naturally, which includes the tangible interaction.

In [5], and also in a recent Special Issue on "Distributed User Interfaces" of IJHCI (vol. 28, issue 11, 2012) a set of approaches for DUI design are envisaged.

Design methods for TUI domain

TUI (Tangible User Interface) has been proven effectively to allow users to control and comprehend digital information as it supports direct manipulation and the utilization of spatial knowledge [7]. Since, numerous researches are focused on this type of interaction (see for instance the recent editions of the TEI and ITS conferences).

For example, Ullmer [16] introduced the concept of token+constraint systems, which considers tokens as physical objects representing digital information or operations, and constraints confining regions in which tokens are placed and manipulated. The TAC paradigm uses Ullmer's token+constraint approach as its basis. It then extends the concept of constraint, stressing that a TUI may be described as a set of relationships between a token, a set of constraints, and a variable [15].

The tangible objects have to be clearly defined in order to have a sense to the users. The user has to understand the functionality associated to each tangible object (in the GUI, the problematic was the same which led to the solution of widget concept). With the same logic, we proposed the Tangiget concept [2, 12].

Zoil [18] aims at proposing design principles dedicated to post-wimp distributed interfaces. But this article shows also that it is difficult to maintain the usability in these systems.

Provide intelligence in new post-WIMP and distributed systems in order to allow designing applications more complex than office software but also to ensure a use comfort, lead us to combine MAS with design principles of DUI.

PROPOSITION

This section presents our model for managing entities on several tabletops. These entities may be virtual and / or tangible (depending on the interactive surfaces concerned) and are evolving either locally (*e.g.* on only one interactive surface) or in a distributed way (*e.g.* on a network of interactive surfaces).

The proposed interaction model is based on Multi-Agent System (MAS) concepts. These concepts are used to design and implement distributed applications for interactive tabletops. They take into consideration the heterogeneousness of the entities, and also the distribution of the information between entities. In addition, it considers the interaction between several interactive surfaces.

In Figure 1, we present a model for the design of applications distributed on several interactive surfaces. For ease of reading, we represent only two interactive surfaces but the number of surfaces is not limited. To define the relationships between agents evolving on a MAS platform, we associate one MAS platform by interactive surface.

The MAS platform is FIPA-compliant (cf. http://www.fipa.org/) to follow criteria of compatibility. The MAS platform is composed of a set of functionalities like the agents management (e.g. to create, to delete, to research, etc., an agent on the platform) and messages protocols (MSG) to exchange information between local agents and between agents in a remote platform (e.g. to inform, to request, to confirm, etc.).

To represent the entities evolving on interactive surface, we distinguish and associate three types of agents: virtual, Tangiget and clone agent. **The Virtual Agent (VA)** is associated to virtual objects with a graphical representation on the interactive support. **The Tangiget Agent (TA)** is connected to a Tangiget object. The agent association to Tangiget object depends on the object proprieties. We distinguish local and distributed properties. A Tangiget object with local property can be manipulated by users and have local actions in only one interactive surface. A Tangiget object with distributed property can be manipulated by users to interact remotely with other tabletops. **The clone agent (CA)** is linked to a Tangiget agent with distributed property. The number of clone agents for one Tangiget agent depends mainly on the number of connected interactive tables and the concerned distributed application.

In Figure 1, we show that the number of agents is not limited and that an agent is necessarily associated to a platform (we consider also in this figure two platforms; but the number of platforms is not restricted). For example, $\mathbf{VA}_1^2$ is associated to the virtual agent number 1 evolving on the interactive surface number 2. We note **i** and **k**, the integer variables representing the Tangiget agents (with local properties) matched to interactive surface 1 and 2. We note **j** and **l,** the integer variables representing the Tangiget agents (with distributed properties) and the clone agents associated with these agents. For example on the interactive surface 1, when one of these agents is created ($\boldsymbol{TA}_j^1$), it is automatically cloned to all other connected interactive surfaces. In the case of two connected platforms, we have only a single clone created on the interactive surface 2 ($\boldsymbol{CA}_j^2$). If an interactive surface is not initially connected, the MAS platform analyzes dependencies to create all the useful clone agents during the initialization.

CASE STUDY

In this section, we show a case study related to the distributed simulation of road traffic management to implement the model presented in the previous section.

Initially, the tool [8] has been designed on one interactive *TangiSense* tabletop. This tabletop exploits the RFID technology for the detection of objects equipped with tags.

This tabletop is designed and produced by the RFIdées Company. We have extended our approach to a distributed approach for two interactive *TangiSense* tabletops (these tabletops are connected on a network). The simulator aims to test different hypotheses concerning the traffic [10]: waiting time reduction in crossroads, crisis situations management, infrastructure changes, and so on.

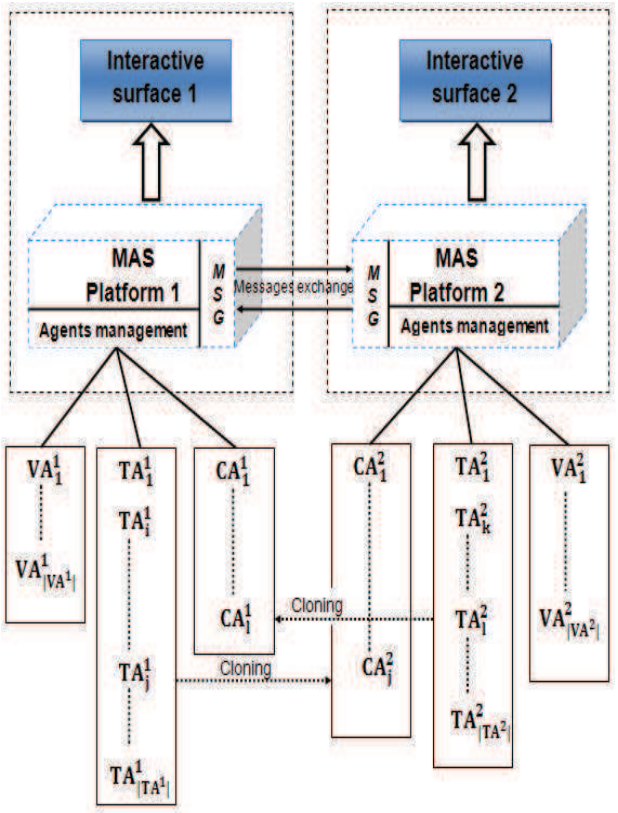


Figure 1. Model dedicated to the management of distributed interactive surfaces with MAS Platforms (legend: TA = Tangiget Agent; VA = Virtual Agent; CA = Clone Agent, MSG = messages protocols)

The road traffic simulator is implemented with the JADE multi-agent platform that respects the specifications provided by the FIPA standard.

The Figure 2 illustrates two examples of implementation of Tangigets in the road traffic simulation. Virtual agents are represented on each tabletop by vehicles moving randomly or according to a set of goals on the entire road network. These agents have behaviors that allow or not for example to respect the Highway Code. Tangigets (with local properties) are used for manipulating the map. These objects are manipulated by the users and they interact with other virtual objects. These objects are equipped with RFID tags to be able to modify the network structure.
For example, to move the map, to view information about the name or the speed of the road, to zoom in or out and change the scale of the map, they may use different Tangigets. We note that these Tangigets do not affect the other tables. This kind of Tangiget allows the users to have an independent view of the map. For example, as showed in Figure 2(a), a user at the tabletop 2 zooms by turning a Tangiget and changes locally the scale of the map.

Figure 2(b) shows another example: a Tangiget with distributed properties is used on the tabletop 1. These properties allow it to be cloned using agent located in another interactive surface. The advantage of this Tangiget is to coordinate display tabletops to work together on the same area. When this object is put down on the tabletop 1, the tabletop 2 will generate a clone agent (these two agents exchange messages in order to keep the data coherence). Messages contain the position and the scale of the map. This message is used by the tabletop 2 to obtain the same vision as the tabletop 1.

Figure 2. Road traffic management on two *TangiSense* with Tangigets (a) without effect on the other tabletop, (b) with effect on it

Accordingly such representative examples, we validated the principles of MAS to Tangiget, virtual and clone entities operating at the surface of interactive tables. We showed the interactions between two interactive surfaces enable the collaboration and the information exchange between different users during the simulation.

CONCLUSION

Distributed applications on several tabletops lead to new design problematic, particularly when the objective is to manage different types of virtual and tangible objects in these applications. Each tabletop is in fact considered as an interactive surface usable by several users. We have proposed a model for the management of distributed interactive surfaces with MAS platforms. In this case a tangible object used on a surface may be cloned and represented virtually in another surface, by the use of so-called Tangiget objects proposed in previous works [2, 12].

The application concerns road traffic management simulation on several tabletops. This innovative distributed application allows us to validate the model proposed and to implement (a) different types of functions, but also (b) tangible objects able to manage them locally or remotely.

Our research perspectives are the following: (a) to implement different types of Tangigets usable with several connected tabletops, (b) to study conflict management between the different Tangigets. More the works of [13] which propose so-called *GaussBits* to interact with objects on different types of surfaces offer perspectives to test our model with other surfaces and technologies.

ACKNOWLEDGMENTS

This research was partially financed by the French Ministry of Education, Research & Technology, the Nord/Pas-de-Calais Region, the CNRS, the FEDER program, the International Campus on Safety and Intermodality in Transportation (Plaiimob project), and the French National Research Agency (ANR TTT and IMAGIT projects, financial IMAGIT support: ANR-10-CORD-017). The authors would like to thank the partners with whom we collaborated on the TTT and IMAGIT projects: LIG, RFIdees and the CEA.

REFERENCES

1. Bandelloni, R. and Paternò, F. Migratory user interfaces able to adapt to various interaction platforms. *Int. J. Human-Computer Studies* 60, 5-6 (2004), 621-639.
2. Caelen, J. and Perrot, C. Bibliothèques d'objets. Rapport de projet ANR IMAGIT [Object libraries. Report of the ANR IMAGIT project] (2011), Grenoble, France: LIG.
3. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. *Interacting with Computer 15*, 3 (2003), 289–308.
4. Demeure, A., Sottet, J-S., Calvary, G., Coutaz, J., Ganneau, V. and Vanderdonckt J. The 4C Reference Model for Distributed User Interfaces. In *Proc. ICAS 2008*, D. Greenwood, M. Grottke, H. Lutfiyya, M. Popescu (eds.), IEEE Computer Society Press (2008), Los Alamitos, Gosier, 61-69.
5. Gallud, J.A., Tesoriero, R. and Penichet, V.M.R. (Eds.), *Distributed User Interfaces, Designing Interfaces for the Distributed Ecosystem*, Springer (2011).
6. Grolaux, D., Van Roy, P. and Vanderdonckt, J. Migratable User Interfaces: Beyond Migratory User Interfaces. In *Proc. of MOBIQUITOUS'04*, IEEE Computer Society Press (2004), Los Alamitos, 422-430.
7. Ishii, H. and Ullmer, B. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proc. ACM CHI'97* (1997), 234–241.
8. Kubicki, S., Lebrun Y., Lepreux, S., Adam, E., Kolski, C. and Mandiau, R. Simulation in Contexts Involving an Interactive Table and Tangible Objects. *Simulation Modelling Practice and Theory 31* (2013), 116–131.
9. Larsson, A., Ingmarsson, M. and Sun, B. A Development Platform for Distributed User Interfaces. In *Proc. SEKE07* (2007), Boston, U.S.A.
10. Lebrun, Y., Adam, E., Kubicki, S., Mandiau, R. A Multi-Agent System Approach for Interactive Table Using RFID. In *Proc. PAAMS* (2010), 125-134
11. Lepreux, S., Kubicki, S., Kolski, C. and Caelen, J. Distributed Interactive Surfaces: A step towards the distribution of tangible and virtual objects. In J.A. Gallud, R. Tesoriero, V.M.R. Penichet (Eds.), *Distributed User Interfaces, Designing Interfaces for the Distributed Ecosystem*, Springer (2011), 133-143.
12. 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 (2012), 709-721.
13. Liang, R.H., Cheng, K.Y., Chan, L.W., Peng, C.X., Chen, M.Y., Liang, R.H., Yang, D.N., Chen, B.Y. GaussBits: magnetic tangible bits for portable and occlusion-free near-surface interactions. In *Proc. CHI* (2013), ACM, 1391-1400
14. Luyten, K., Van den Bergh, J., Vandervelpen, C. and Coninx K. Designing distributed user interfaces for ambient intelligent environments using models and simulations, *Computers & Graphics 30*, 5 (2006), 702-713.
15. Shaer, O., Leland, N., Calvillo-Gamez, E. H. and Jacob, R. J. K. The TAC paradigm: Specifying tangible user interfaces. *Personal and Ubiquitous Computing 8* (2004), 359–369.
16. Ullmer, B. Tangible interfaces for manipulating aggregates of digital information. *PhD thesis*, Massachusetts Institute of Technology (2002).
17. Weiser, M. Some computer science issues in ubiquitous computing. *Communications ACM* 36, 7 (1993), 75-84.
18. Zöllner, M., Jetter, H.C. and Reiterer, H. ZOIL: A Design Paradigm and Software Framework for Post-WIMP Distributed User Interfaces. In *Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem*, Springer-Verlag (2011), 87-94

Proxywork: Distributing User Interface Components of Web Applications

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

ABSTRACT
In this paper, we present the *Proxywork* application 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 operations (i.e. show, hide, copy, move, etc.) attached to Web page components. As these operations are automatically attached to Web page components on runtime by the *Proxywork* Web proxy, Web pages do not require any extra information to be distributed among different displays. To illustrate how to distribute Web application user interface components, this paper presents the University of Castilla-La Mancha Web site as a study case showing the results of performing user interface distribution operations among displays running on different platforms.

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

General Terms
Design, Human Factors, Languages.

Keywords
Distributed User Interfaces, Web, Proxywork

1. INTRODUCTION
Cloud computing allows users to centralize information in a secure and scalable way at reasonable low prices. In fact, many companies and individuals migrate their applications to the cloud to reduce maintenance costs.

According to Mimecast[1], 70 % of companies worldwide use cloud services to provide end users with information which is accessible from anywhere. In addition, the advances in Web development tools have reduced the time to market of Web applications employing rich user interfaces (UIs).

Once we have exposed the importance of Web Applications, and how these applications can be used ubiquitously, we are ready to expose how to exploit the advantages the Distributed User Interface (DUI) paradigm in ubiquitous computing environments.

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

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

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

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

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

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

This article is organized as follows. This section reveals a gentle introduction to the motivation of this work. The Section 2 presents the most relevant related work. A detailed description of the *Proxywork* proxy is presented on Section 3. Section 4 presents a case of study that show how *Proxywork* supports the scenario presented. Finally, Section 5 presents article conclusions and future work.

2. RELATED WORK
DUIs are actually evolving; therefore, there are few frameworks that support DUIs. Moreover, most frameworks do not support full-fledged DUIs.

From native programing language perspective, UI development toolkits such as Java Swing or Windows Presentation Foundation (WPF) do not support DUI concepts. They just allow developers

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

to assign UI components to a container/context where they can communicate each other. However, once these components are assigned to their container they cannot be reassigned or redistributed to another container or context residing on different runtime platforms. An example of fixed distribution of repartition of UI elements between smartphones and TVs is presented in [8]. Therefore, UI component distribution is predefined and opportunistic because no reconfiguration is allowed at runtime.

Some approaches allow the distribution of UI components at runtime with limitations. For instance, some Web browsers allow users to open a new window to display images that are embedded into a Web page. However, the distribution is limited to the same runtime platform. Besides, the distribution is limited to images and a couple of HTML elements, such as links. If you want to distribute other HTML UI components, such as a DIV tag that defines a panel on a form, it is not possible.

Therefore, the granularity of UI components to be distributed is fixed and often coarse-grained (dialogs and windows); it is not possible to distribute at the widget level. The same limitations are applied to replication support.

In [3], a Web page is split into several partial pages which are distributed to all the users. Out approach supports multi-device and multi-user Web browsing where clients are connected to the server that delivers the page.

A proxy split pages according to device and user constraints. Therefore, each Web page is represented by a XML file containing specific tags to configure how the Web page is split among users and devices. Although, this work is similar to our proposal, the advantage of our approach lays on the lack of the definition of configuration tags at design time to distribute the UI. These tags are replaced by the enrichment of the Web page with distribution operations at runtime.

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

In [5], authors attempt to model the UI component distribution. However, the granularity is limited to tasks that are defined at design time.

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

Another solution regarding partial Web migration is presented in [2]. It allows users to select parts of existing interfaces interactively and migrate to a target device. To achieve this goal, this approach uses a native application that allows users to select the parts of the web application interface to migrate. The main difference between the work presented in [2] and ours is the lack of a native application to distribute UI components, the interface to distribute UI components is embedded into UI components, instead. As result of this way of performing distribution operations, the UI is inherently easier to use.

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

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

3. PROXYWORK: DISTRIBUTING WEB APPLICATIONS

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

The Figure 1 shows the overview of the *Proxywork* architecture.

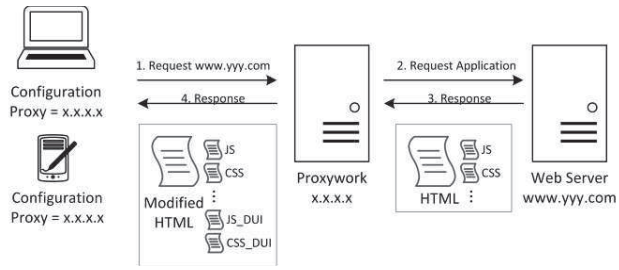


Figure 1. Overview of the Proxywork architecture

For instance, if the *Proxywork* proxy is hosted on the x.x.x.x IP address listening on port 80. Devices that are part of the same display ecosystem should set Web browser proxy IP address to x.x.x.x and the port to 80.

Once the device Web browser is registered, the request follows these steps to display the Web page enriched with distribution operations: The request for the page http://www.yyy.com departs from device browser and arrives to the *Proxywork*. *Proxywork* requests the Web resource to the web server where the application is hosted (http://www.yyy.com). The Web server returns the Web resource to the *Proxywork*. *Proxywork* modifies the resource by inserting HTML, CSS and Javascript extra code in the page in order to add distribution operations, and provide a list of devices where users are able to distribute UI components. *Proxywork* returns the Web page transformed into a distributable Web page to the Web browser device that sent the request. Finally, the Web browser shows the distributable Web page (http://www.yyy.com).

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

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

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

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

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

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

3.1 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 the HTML tags. Therefore, *Proxywork* sets the distribution granularity to the <DIV> HTML tag level because this tag represents groups of graphically related tags. However, users are able to set other HTML tags to make the granularity more flexible.

3.2 The Proxywork distribution process

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

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

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

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

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

Besides, the Code Manager delegates to the Link Manager module the modification of the navigation links of the Web page. Finally, the Code Manager delegates to the Granularity Manager the selection of the HTML tags that are enriched with distribution capabilities. Once, the Code Manager has processed the Web page, it is sent back to the browser that made the request.

3.3 Distribution operations

This section exposes the set of distribution operations supported by *Proxywork*.

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

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

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

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

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

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

Let A and B be two different device Web browsers. If users perform a Copy on a UI component named X of A that affects B, all subsequent operations performed on X do not affect B in any way.

However, if users perform a Distribute operation on a UI component named Y of A that affects B, the Y UI component disappears from A to appear in B. Besides, any operation performed on Y in B is targeted to A.

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

4. CASE OF STUDY

This section presents how *Proxywork* is employed to support DUI for Web applications in a real scenario. However, the reader can find a video demo which summarizes most important *Proxywork* features in http://youtu.be/MEC2Y5rVGXQ.

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

The distribution is carried out by 2 devices: Dell XPS M1530 laptop computer running the Microsoft Windows 8 operating system. And Nokia Lumia 900 smartphone running the Microsoft Windows Phone 8 operating system.

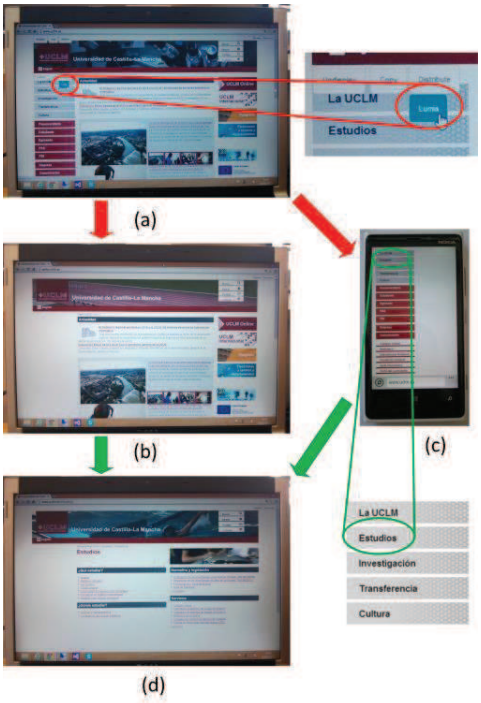


Figure 2. Distributing the navigation menu

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

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

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

5. CONCLUSION AND FUTURE WORK

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

The distribution granularity can be configured to manage the complexity of Web pages. By default, the application granularity is set to the <DIV> tag level.

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

To show the capabilities of *Proxywork*, we exposed a case of study. We exposed how to deal with the distribution of Web application navigation bars where users distribute the navigation bar from a laptop or desktop computer to a smartphone in order to use the smartphone as a "remote control" of the computer display.

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

Besides, we are thinking on allowing the automatic layout of UI component according to the target device when they are distributed.

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

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

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

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

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

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

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

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

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

CopyFlyPaste: Distributing information on Distributed User Interfaces

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

ABSTRACT
Collaborative systems, ubiquitous computing environments and collaborative work, are strengthening some techniques that help users to work with information in these environments. Currently there are many ways to transfer information among a set of devices; however, they do not provide a direct mechanism to exchange information in the same way we do it in the same device. This paper presents the CopyFlyPaste direct manipulation technique to exchange information among different devices. It allows users to copy any resource from a source device to paste it directly into different one without worrying about underling issues related to the communication infrastructure supporting the information transport. This paper also describes the AirClipboard application which supports the CopyFlyPaste interaction technique. This application is evaluated on real users to demonstrate the efficiency, effectiveness and satisfaction of use compared to a set of traditional techniques used to exchange information among different devices.

AUTHOR KEYWORDS
CopyPaste, CopyFlyPaste, AirClipboard, Distributed User Interfaces.

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

GENERAL TERMS
Human Factors; Design; Languages.

1. INTRODUCTION

In a ubiquitous computing environment, we use a set of devices to perform tasks. The combination of devices is usually dynamic and heterogeneous. For instance, to copy a file from one disk, and paste it into another disk, in the same computer, is an easy task to perform. However, to copy one, or several files, from one computer to another one may not be as easy as in the previous case.

The situation becomes more difficult when inexperienced users have to exchange a paragraph of text, or a simple URL, from one compute to another.

Currently, users employ different mechanisms to transfer files, or any another type of information among a set of computers. If it is a short text, users usually type the text in the target device (i.e. email addresses or URLs). If the information to transfer is a set of files, or long texts, there are other alternatives (i.e. sharing files using a distributed file system, an FTP server, e-mail, removable storage units, synchronized folders, instant messengers, etc.).

This paper presents the CopyFlyPaste technique based on the traditional "copy and paste" (Copy&Paste) provided on most operating systems to easy the transference of information in a distributed environment employing a mechanism that abstracts the device distribution from users.

This paper is structured as follows. This section presents an overview of the problems motivated this work. In Section 2, it shows the work related regarding information transference techniques on a distributed environment. The Section 3 describes the CopyFlyPaste technique. Subsequently, the Section 4 presents the AirClipboard prototype that implements the CopyFlyPaste. Finally, the Section 5 exposes the set of conclusions and future works.

2. RELATED WORK

This section presents an overview of the tools that users typically employ when transferring information from one device to another, or when the users exchange information to among group of users.

The Pick-and-Drop technique is one of the oldest [6]. This technique allows users to simulate the action of picking a resource from one computer and drop it in another one by the means of a special pen. Note that the use of an extra device (special pencil) is mandatory and users have to move themselves from one device to another.

Similar techniques that employ the special pen to transfer information are compared in [5]; however these techniques also introduce the use of an extra device.

Besides, a set of manipulation techniques extending the Drag-and-Drop technique are compared in [3]. One of these

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

techniques is the hyperdragging [7] where are able to users to smoothly exchange digital information among portable computers, table and wall displays, and other physical objects. Hyperdragging requires the knowledge of the physical relationships among devices and a camera to recognize objects.

SPARSH [4] allows users to grab the data item they wish to copy from a device ("recording" it in the user's body) in order to leave it in device users want to paste the grabbed data. Note that this work does not allow users to work with devices that do not support touch capabilities.

Touch & Connect and Touch & Select [10] use the NFC technology to copy a file from a computer to a mobile phone. This system requires a short distance between the devices in order to transfer information. The Touch Projector system proposed in [1] allows users to select files from a projected screen through a phone, and drop it into another screen.

PhonePick&Drop and PhoneCopy&Paste techniques allow users to the transfer data from a computer to a phone and vice versa [8]. The Touch Projector system and PhoneCopy&Paste and PhonePick&Drop techniques require the use of the user's phone to perform tasks.

Finally, the *Network Clipboard* utility software allows users to copy texts, files and images among computers; tough, it does not support multiple user.

Besides, There are some approaches that do not employ the Copy&Paste technique to transfer information among computers.

Table 1 Comparative analysis of the amount of actions required to transfer information between 2 devices

System	Actions	#
IM	Copy(source) → Paste(source) →Press send button(source) →Copy(destination) → Paste(destination)	5
E-mail	New email(source) → Press attach resource(source) → Select attached(source) → Press send mail (source) → Read mail (destination) → Press download attached(destination) → Select destination(destination) → Press save resource (destination)	8
FTP Client	Open FTP session(source) → Copy resources from PC(source) → Paste resources to repository(source) → Open FTP session(destination) → Copy resources from repository(destination) → Paste resources to PC(destination)	6
Folder Synch.	Copy resources(source) → Paste resources to synchronized folder(source) → Copy resource from synchronized folder (destination) → Paste resources(destination)	4
CopyFlyPaste	Copy resource (source) → Paste resource (source)	2

Instant messaging applications (i.e. *Skype, Micrtosoft Messenger, GTalk, Facebook Messenger, Whatsapp, Viber, Tango,* etc.) have some disadvantage, if the same user wishes to transfer a resource between two of his/her devices, he/she has to log in twice. And some applications do not allow user to log in more than once.

E-mail clients (i.e. *Outlook, Gmail Web, Mozilla Thunderbird, Windows Live Mail*, etc.) have some drawbacks, for instance: (a) the resource transference from one device to another is not transparent to the users, and (b) mail providers usually limit the size of the resources that can be sent as attachments to emails.

FTP clients (i.e. *Filezilla*, *FireFTP*, *Windows FTP*, etc.) share similar problems to the ones detected on e-mail clients, FTP clients do not provides users with a "transparent" technique. Therefore, this mechanism requires users to manage to concept of repository in order to perform the transference, and this knowledge may not be familiar to users with a short experience on computers.

The folder synchronization applications (i.e. Dropbox, ZumoDrive, SkyDrive, Google Drive, etc.) require users to define a group in order to share a folder, requiring the setup of group users which is not usually clear beforehand.

The Table 1 shows the amount of actions required to transfer information between different devices using different information transference systems.

3. COPYFLYPASTE

This section presents our approach to solve the self of problems stated on Section 2.

2.1. Technique overview

The CopyFlyPaste is defined as an extrapolation of the traditional technique of copy and paste (Copy&Paste) to a distributed environment.

When users employ the Copy&Paste technique, they: select a resource, or set of them; then they select an operation to perform, for instance the "copy" (using *Ctrl + C*) operation; finally they execute the operation on the destination, for instance, the "paste" (using *Ctrl + V*) operation.

However, when users employ the CopyFlyPaste technique, they select the resource, or the set of resources, to transfer by using the same *Ctrl + C* key combination used to perform the copy action on traditional scenarios. Consequently, users perform the "copy on the fly" operation which is the first action of the CopyFlyPaste technique. Immediately after, resources are available to all users that are in the same working group. Finally, from any computer belonging to the group, the user that is interested on the information performs the "paste on the fly" action which is the second action of the CopyFlyPaste technique by using the same *Ctrl + V* key combination used to perform the paste action on traditional scenarios.

Therefore, the CopyFlyPaste technique offers the following advantages regarding the approaches analyzed on Section 2: (a) perform a distributed Copy&Paste information transference action in a transparent way from the user perspective, (b) avoids the use of without of any extra

object to perform the information transference, (c) achieves the location independence due to the Internet infrastructure, (d) encourages the independence between the copy and paste actions (the user performing the copy action is not usually the same that performs paste action).

2.2. CopyFlyPaste for DUI

In a 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 [11]. Besides, in DUI scenarios to transfer resources between interfaces hosted on different devices is very common, but this task is not trivial.

The CopyFlyPaste technique offers a solution to resolve the commented problem because users can copy any resource from a source interface and they can paste it directly into different one without worrying about underling issues related to the communication infrastructure.

2.3. Scenarios

This section describes a set of scenarios where users and groups work with CopyFlyPaste.

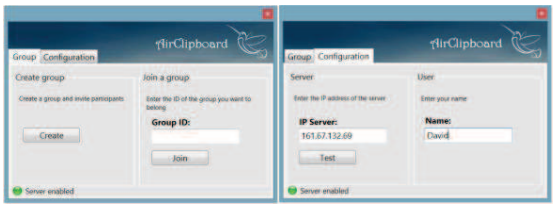
1. **Single user / Same machine:** This is the basic, and the most common, scenario. A single user works with a single computer, and needs to copy and paste resources. In this case, the CopyFlyPaste technique behaves exactly like the tradition Copy&Paste.
2. **Single user / Multiple machines:** This scenario occurs when the same user has to transfer information between different devices.
3. **Same group of users:** This scenario is very common in collaborative situations where the information is shared among a group of users.
4. **Single user / Multiple groups of users:** This scenario is more complex than the previous one because it occurs when there are multiple groups of users who work together using the CopyFlyPaste technique. Therefore, users are able to join to a single group at a given time.
5. **Multiple users / Different groups / Multiple groups per user:** In this scenario, the same user is able to belong to different workgroups at the same time. To copy or paste information from/to a specific group, the user select the group where to perform the operation.

4. AIRCLIPBOARD. CASE OF STUDY

To illustrate the CopyFlyPaste technique, the AirClipboard application was developed. And along with this section, we explain the application functionality.

The use of this tool is very simple. First, the user has to launch the application and join to an existing workgroup, or create new one. The Figure 1.a shows how to perform this task.

The use of this tool is very simple. First, the user has to launch the application and join to an existing workgroup, or create new one. The Figure 1.a shows how to perform this task.



(a) (b)

Figure 1 (a) Allow to create a new workgroup or to join an existing workgroup, (b) Allow to configure the server.

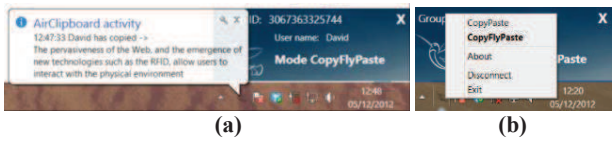
The use of this tool is very simple. First, the user has to launch the application and join to an existing workgroup, or create new one. The Figure 1.a shows how to perform this task.

The tab configuration allows users to change their name and the IP address where the server is located. Figure 1.b shows this step.

Once users are registered in a workgroup, the application is visible in a small area of the screen at the bottom right corner (see Figure 2.a). This area shows up the group information, the username and the information transference mode.

The application defines 2 operation modes: the CopyPaste mode and the CopyFlyPaste mode. When users are in CopyPaste mode the system performs the traditional CopyPaste action. However, when the mode is changed to the CopyFlyPaste (see Figure 2.a) mode, the resource to be transferred is available to all users in the system. Therefore, they are able to perform the paste operation, to get the copied information.

Users are able to change the mode by accessing the context menu that pops-up from the system tray icon that appears in the taskbar (see Figure 2.b).



(a) (b)

Figure 1 (a) CopyPaste/CopyFlyPaste mode and notifications, (b) Contextual menu.

When users perform the copy action, a notification pops-up from the system tray icon to notify that a resource is available to be pasted. Notifications also show the user who has copied the resource, and part of the copied content. An example of this situation is shown in Figure 2.a.

Regarding to the scenarios discussed in Section 2.3, AirClipboard covers the set of scenarios from 1 to up to 4.

5. USABILITY EVALUATION

The validation of the approach was performed using a usability evaluation based on the 9 steps for conducting a qualitative test according to Sauro [9].

The reason is to evaluate the AirClipboard is to confirm, or not, the following hypothesis: "CopyFlyPaste is more efficient to copy and paste text between devices than the system that is currently being employed by users".

We selected a user population of 10 users. All of them have an advanced technological knowledge, and have previously transferred information among computer using alternative methods. All participants were asked to perform the same 6 tasks in the same conditions. These tasks were:

Task 1: (T_1): The user need search an image of a red car from Internet, on the Computer 1, and display the image found on the Computer 2.

Task 2: (T_2): In this task, the user works with another user. The first user has to find an image of a blue car from the Internet, using the Computer 1, and then display it on the Computer 2. Then the second user should search, on the Computer 2, an image of a red car, and allow the first user to display it on the Computer 1.

The users were asked to choose the system they wanted to use to transfer the information. Users performed the two tasks with the system they have chosen, and later, the same two tasks were performed using the AirClipboard system. After each testing session, users fulfilled a satisfaction the System Usability Scale (SUS) [2] questionnaire.

Table 2 Completion time for each task (in sec.)

ID	*System chosen*	*System chosen*		*AirClipboard*	
		T_1	*T_2*	*T_1*	*T_2*
1	Gmail	70	86	38	37
2	Pen Drive	83	129	16	40
3	Gmail	36	74	24	37
4	Gmail	37	80	16	32
5	Gmail	51	95	14	35
6	Sync folder	13	37	11	23
7	Gmail	83	92	33	52
8	Gmail	33	61	12	32
9	Gmail	55	76	19	41
10	Gmail	35	72	14	30
	Average	*49,6*	*80,2*	*19,7*	*35,9*
	Confidence interval	[32,99-66,21]	[63,12-97,28]	[13,13-26,27]	[30,37-41,43]

As result of the test session we collected the information shown in Table 2. The Table 2 shows the users' time to complete each task and the confidence intervals around the average time. Note that all users completed all tasks with both systems.

The Figure 4 shows that users are more efficient when they use the CopyFlyPaste instead of the system they have chosen. The collected data reveals that using the CopyFlyPaste technique, both, task 1 and task 2, were performed on an average time that is less than half the time employed to perform the same task with the technique users have chosen. These results validate the hypothesis raised.

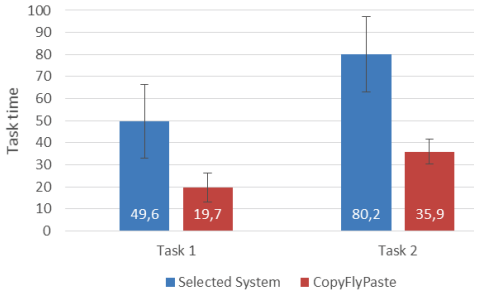


Figure 4 The confidence intervals around the average time for each task in sec.

On the other hand, the SUS score is 86.8 regarding a maximum of 100. If this score is compared with the benchmark presented by Sauro, we can affirm that AirClipboard is better than the 75% of the applications evaluated in his benchmark.

6. CONCLUSIONS AND FUTURE WORKS

In this work we propose a technique to facilitate the resource transference among devices that is "transparent" to the user (no information overhead derived from the use of the tool). Thus, the CopyFlyPaste technique allows users to copy a resource from one device, and paste it directly to another device without worrying about infrastructure or distribution issues. Furthermore, this technique does not employ any extra device to perform the information transference; besides, it can be used on any device connected to the Internet. Finally, it decouples the copy and paste actions from the user that performs the action.

We conducted a comparison of the proposed technique to existing systems to transfer information among devices. This comparison shows that the proposed technique involves fewer actions than traditional approaches and therefore, it is simpler to use and learn.

To illustrate the potential of CopyFlyPaste technique we developed the AirClipboard tool. Regarding to the efficiency, the hypothesis "CopyFlyPaste is more efficient to copy and paste text between devices than the system currently used by users" has been confirmed. And the satisfaction evaluation has revealed positive results.

As future works, we are developing an improved version of the prototype to support any type of multimedia resource, and we are also implementing the rest of the scenarios described in Section 2.3.

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 Scientific Research, Technological Development and Innovation Regional Plan 2011-2015(PRINCET).

REFERENCES

1. Boring, S., Baur, D., Butz, A., Gustafson, S. and Baudisch, P. Touch projector: mobile interaction through video. In CHI pp. 2287–2296. 2010.
2. Brooke, J. SUS: A quick and dirty usability scale, in: P. W. Jordan, B. Weerdmeester, A. Thomas, I. L. Mclelland (Eds.), Usability evaluation in industry, Taylor and Francis, London, 1996.
3. Collomb, M., and Hascoet, M. Extending drag-and-drop to new interactive environments: A multidisplay, multi-instrument and multi-user approach. Interacting with Computers, 20:562-573, 2008.
4. Mistry, P., Nanayakkara, S., Maes, P. SPARSH: Passing Data using the Body as a Medium. In the proc. of CSCW2011. Interactive Paper. Hangzhou, China. 2011.
5. Nacenta, M. A., Aliakseyeu, D., Subramanian, S. and Gutwin, C. A comparison of techniques for multi-display reaching. In Procs of the SIGCHI conference on Human factors in computing systems CHI '05 pp. 371-380, ACM, New York, USA. 2005.
6. Rekimoto, J.: Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In: Proc. of UIST'97. ACM Press, New York (1997) 31–39.
7. Rekimoto, J., Saitoh, M., 1999. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In: ACM CHI'99 Proceedings, ACM Press 378–385.
8. Schmidt, D., Seifert, J., Rukzio, E. and Gellersen, H. A cross-device interaction style for mobiles and surfaces. In Procs of DIS '12 pp. 318–327, ACM, New York, NY, USA. 2012.
9. Sauro, J. A Practical Guide to Measuring Usability, CreateSpace, 2010.
10. Seewoonauth, K., Rukzio, E., Hardy, R. and Holleis, P. Touch &; connect and touch & select: interacting with a computer by touching it with a mobile phone. In Procs of the 11th International Conference on HCI with Mobile Devices and Services MobileHCI'09 pp. 36:1–36:9, ACM, New York, USA. 2009.
11. Vanderdonckt, J. Distributed User Interfaces: How to Distribute User Interface Elements across Users, Platforms, and Environments. Proc. of XI Interacción, 20-32, 2010.

Distributed Data and Displays via SVG and HTML5

Jeff Wilson
Carleton University
Ottawa Canada
jeff.wilson3@carleton.ca

Judith Brown
Carleton University
Ottawa Canada
mmjbrown@connect.carleton.ca

Robert Biddle
Carleton University
Ottawa Canada
robert_biddle@carleton.ca

ABSTRACT
The emergence of open source HTML5 JavaScript frameworks that support client-side templating and data binding offers opportunities for low cost interactive visualizations that can share state between document fragments, separate documents, screens, and systems thus paving the way toward distributed user interfaces. We explore work involved in sharing prototype visualizations using SVG and discuss how open source frameworks can improve the design experience in creating novel and accessible interfaces.

Author Keywords
HTML5; SVG; client-side templates; reactive programming; DUI.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

General Terms
Human Factors; Design.

INTRODUCTION

There are numerous potential uses for multi-display environments, particularly in real-time collaboration. Our lab has studied these environments in software development, usable security, intelligence analysis, and network monitoring. We are interested in deployments with large surface displays in wall-mounted and table configurations, and we also wish to validate the benefits that handheld displays might bring to these sorts of systems.

Rapid prototyping and deployment of new interaction methods that employ multiple displays can become constrained by available architectures, particularly when bringing these prototypes to users' workplaces for validation. For this reason we are interested in application architectures that leverage the ubiquity of web browsers as well as standard web-based protocols.

Having had some success in this area we have begun to search for useful building blocks upon which new interaction methods and tools can be quickly developed and validated. We are particularly interested in frameworks that are being designed with real-time multi-display collaboration in mind.

Figure 1. Dashboard Distributed in 42 Minutes

One such framework is Meteor.js [1]. The Meteor team hopes to define "A better way to build apps", an ambitious and laudable goal. Their product is still pre-beta, but on first examination it appeared to be in an advanced enough state for use in prototyping. It also fit well with our previous experience with the network application platform Node.js [6].

This paper describes, from a developer's perspective, the experience of using Meteor to design a distributed interactive graphical dashboard backed by a system capable of forming a model-driven distributed user interface. Figure 1 shows how an application written using Meteor can be quickly partitioned across systems; a dashboard originally designed for a single analyst was split into a distributed system with twinned views in under an hour.

Usage contexts

The contexts of use we wish to support can be broken down with these basic questions : *who*, *what*, *where*, and *when*. The question of *how* is the subject of this paper.

- **Who** – Single users on single screens, multiple users on single screens, single users on multiple screens, and multiple users on multiple screens.
- **What** – Interactive visualizations, streaming updates, and web GUIs manipulating various levels of data: domain (shared), collaboration (shared), and session (private).
- **Where** – At a desk, co-located within a team room, or participating in a group session from a remote location.
- **When** – Concurrently by publish-subscribe semantics and data replication.

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

BACKGROUND AND RELATED WORK

The last decade has witnessed a gradual and perhaps reluctant march of web application logic from the server to the client [4, 11, 10, 2, 12]. The reluctance to migrate application logic entirely to the browser can be explained by a lack of standards and concerns about security. These issues remain relevant today, but there are promising developments in open source communities forming around HTML5 and the ever popular Node.js.

Web developers have historically been saddled with the challenge of forming mental models for their applications out of assemblies of nested HTML tags that bear little resemblance to application structure. This conceptual *impedance mismatch* has often been applied to *object-relational mapping*. In HTML we find another mapping of objects into attachment points like DIVs and SPANs, a conceptual leap alleviated only slightly by CSS class names.

The reason for this may be that prior to HTML5 web browsers complied with stricter rules when rendering HTML into the web pages we enjoy. It was possible to create the illusion of a web "application" with JavaScript libraries such as jQuery, but these applications had to be defined in an *imperative* fashion that waited for rendering to complete before "tricking" the document object model (DOM) into behaving like an application. The problem with this approach is that the development effort seems to scale exponentially with respect to application complexity.

The HTML5 specification loosened some of the constraints on HTML *tags* (keywords that declared segments of HTML like `<HEAD>` or `<BODY>`), inviting a new generation of JavaScript frameworks to modify the DOM earlier in the rendering phase. This allows for more *declarative* coding techniques that define web applications with *data binding* in client-side templates.

The advent of client-side templates allows the developer to treat their application as an assembly of *document fragments*. This might seem like a distinction without a difference, but frameworks like Meteor and AngularJS [5] encourage coherence around *domain* semantics rather than around units of HTML rendering. Document fragments then become much easier to conceptualize as logical *views* with reactive binding for data and events.

Meteor takes an additional step toward improving the conceptual framework of web applications by simplifying access to data. Their reference implementation describes database queries in a variant of MongoDB's syntax and it offers the same query methods in both server and client JavaScript code. Meteor clients keep a subset of the server data using *publish-subscribe* semantics, creating a "MiniMongo" cache that can be easily redefined with a key-value pairs stored in the client's Session data. Templated HTML fragments are bound to these cached document collections that allow both reading and writing. Subject to permissions that are easily defined, cache writes are flushed to the server where they are automatically replicated to other subscribers.

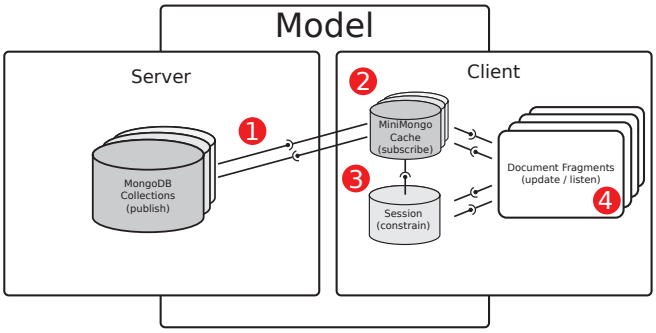


Figure 2. Conceptual Data Model using Meteor.

DATA FLOWS AND DISTRIBUTED UI

With its transparent replication Meteor is able to define a *reactive* application architecture driven primarily by database updates and user events. Database changes–from both the client and the server–are managed with the concept of *reactive computation*. HTML templates are defined with a small vocabulary of substitution, condition, and repetition constructs. Meteor performs an analysis of these features to label fragments of a web page as *reactively* dependent on changes to data. Developers are able to hook into various phases of this reactivity or to define new queries based on custom logic (e.g. reacting to a timeout condition).

Meteor automatically replicates data to all clients subscribing to server-published collections. Figure 2 shows how this allows for the conception of a data model that transparently spans the client and server contexts. The key lies in applying reasonable subscription parameters to collections published on the server (label 1 in Figure 2). Subscribing to massive data collections can be achieved by defining appropriate filters, limits, and database indexes. This ensures that the data replicated to subscribing clients (label 2) appears in reasonably small quantities that can be quickly updated and rendered. Subscription parameters are typically kept in Meteor's client-only Session store (label 3). Document fragments bound by template semantics (label 4) will be automatically re-rendered when either the collections or the session variables are changed. Note that changes to session variables are private to the client, whereas changes to collections can come from the client or the server.

The automatic replication of server and client collections (subject to filters of course) allows for multiple screens to be updated nearly simultaneously, and it is this feature of Meteor that lends itself to distributed user interfaces. If two or more users wish to see precisely the same subsets of shared collections, it is a simple matter of defining their subscription parameters in a common collection, rather than in session variables. By adding an additional field to shared subscription data it is possible to partition these subscriptions into multiple collaborative contexts, for instance defining metaphorical *rooms* or *lobbies* where all users within these spaces share common streams of domain data events.

DATA-DRIVEN VISUALIZATIONS

Thus far we have described a fairly typical scenario for HTML development, but the use of templates and data binding can also apply to rendering SVG. Notice the template placeholders on lines 3 and 4 below.

```
1  <body>
2    <svg id='map' >
3      <g id='countries'>{{> Countries}}</g>
4      <g id='circles'  >{{> Circles}}  </g>
5    </svg>
6  </body>
7
8  <template name="Countries"></template>
9  <template name="Circles"></template>
```

This listing shows a sample template definition for drawing with D3.js within Meteor. The empty templates on lines 8 and 9 are linked to JavaScript objects, such as the one being extended below. The use of two templates within a single SVG drawing (one of which is used in line 12 below) gives us distinct execution contexts that prevent unnecessary refreshes of relatively static data sources.

```
1  EventData = new Meteor.Collection("events");
2
3  _.extend(Template.Circles, {
4    rendered: function () {
5      var self = this;
6      if (!self.handle) {
7        self.handle =
8          Meteor.autorun (function () {
9
10         var svg = d3.select('#map');
11
12         var circles = svg.select('#circles')
13           .selectAll('circle')
14           .data(EventData.find().fetch());
15
16         circles.enter()
17           .insert('circle')
18           .attr('cx',
19             function (d) { return d.x; })
20           .attr('cy',
21             function (d) { return d.y; })
22           .attr('r',
23             function (d) { return d.r; });
24
25         circles.exit().remove();
26
27       });
28     }
29   }
30 });
```

Line 1 of this listing defines a data collection. This collection is queried on line 14, forming a reactive dependency that Meteor automatically updates bidirectionally. Line 3 shows that we are defining methods for the template corresponding to Line 9 in the HTML listing. Lines 16 to 25 define what to do when new objects enter and exit the client-side collection. Configured as shown, the SVG will simply insert and remove circles when records are added and removed according to defined database subscriptions (e.g. line 14 above).

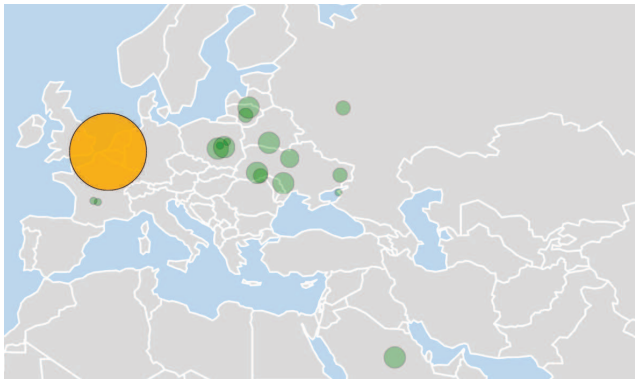


Figure 3. Using geographic projections in D3.js

This simplified example belies the power of D3.js. For instance the functions in lines 19 and 21 could have been defined to derive a circle's coordinates from a function applying a geographic projection, and the radius calculation in line 23 could apply a logarithmic scale for intensity.

Figure 3 shows this type of mapping. This example is taken from a visualization of Apache web server log events we developed. In the actual system additional code is added for tooltips as well as mouse clicks or screen touches. The map was also configured to support zooming and panning.

Web pages can be composed of multiple nested templates. Figure 4 shows an Apache Log Dashboard composed of four parent templates with several children. This dashboard queries a database of approximately a million log entries using Schneiderman's visualization mantra "overview first, zoom and filter, details on demand" [9].

The original design called for a single user on a single screen. As an exercise for this paper, we asked how long it would take to put the world map on its own page in such a way that a remote user could listen for session filters applied by a user on the main dashboard. The original design kept session changes driven by touch or mouse events private to the dashboard user. A new collection was defined called *SharedSession*, and the time series view and the full-page world map template were modified to share subscription parameters through this new collection. There were some additional changes to allow routing by URL using Backbone.js [3], but in less than an hour we had deployed a distributed near real-time user interface.

DISCUSSION AND FUTURE WORK

A design challenge with one test cannot be considered complete validation of our hopes, but it does demonstrate that Meteor was designed with distributed displays as an important principle. The term the Meteor team uses for this feature is "latency compensation", a phrase that sounds like it solves a different problem that we are still interested in exploring, namely that of distant users losing out on races to the same resource. Meteor doesn't currently compensate for unequal latency. For now all updates take place on the quite reasonable assumption of "last writer wins".

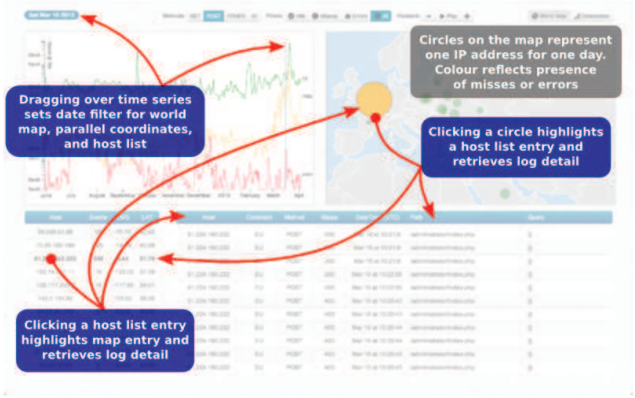


Figure 4. Apache Log Dashboard

In the short term we will be exploring graphical cues to help distant collaborators cope with this assumption when competing for the same on-screen elements. The sharing of real-time data should at the very least help with advisory information, and pushing what is in many cases a human coordination problem into the users' hands is an important first step. In the longer term we might explore more automated approaches to the management of unequal latency.

Many of our designs focus on large screens with multitouch interfaces, and the combination of Meteor and D3 promises to help with the server infrastructure in these plans. Some challenges remain with respect to gestures, in particular gestures that require time to resolve or that can be interpreted in more than one way. Meteor itself will not solve these issues for us, but their flexible package management will provide support for any new libraries we might write or import. The functional programming style made possible by JavaScript is a good fit for gesture handling, and a transparent SVG overlay offers a useful surface for capturing, interpreting, and propagating gesture events. There are numerous JavaScript libraries for smaller screens, but we hope to explore forms of gestural interaction that are more typical of large and/or distributed surfaces where multiple users may be interacting with a single conceptual or actual display. The regular expression approach defined by Kin et al.[7] may offer insights that could be leveraged in client-side JavaScript.

Also outside Meteor's purview but definitely within our own is the issue of how different browser/OS combinations consume gestures before passing them to our applications. This has been an interesting challenge that would not be encountered in a native environment, but we remain hopeful that these issues can be solved.

Part of that research will include more expressive HTML made possible by web component frameworks like Google's AngularJS. The potential for extending HTML is similar to techniques for extending SVG using XML and SMIL as described by King et al. [8]. At the moment we find that we can approach the declarative model they advocate by means of the template-based designs well supported by Meteor and D3, but we are interested to see if AngularJS can be blended with Meteor.

Meteor's foundation in Node.js has served us well in this project and others. The open source Node.js community, and indeed the growing Meteor community offer numerous resources that are easy to integrate on both the server and the client. This richness will allow us to focus on our research efforts in the development of collaborative infrastructure and tools.

REFERENCES

1. Meteor. `http://meteor.com/`.
2. Bolin, M., Webber, M., Rha, P., Wilson, T., and Miller, R. C. Automation and customization of rendered web pages. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, UIST '05, ACM (New York, NY, USA, 2005), 163–172.
3. DocumentCloud. Backbone.js. `http://backbonejs.org/`.
4. Fraternali, P. Tools and approaches for developing data-intensive web applications: a survey. *ACM Comput. Surv. 31*, 3 (Sept. 1999), 227–263.
5. Google. Angularjs. `http://angularjs.org/`.
6. Joyent. Node.js. `http://nodejs.org/`.
7. Kin, K., Hartmann, B., DeRose, T., and Agrawala, M. Proton: multitouch gestures as regular expressions. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ACM (2012), 2885–2894.
8. King, P., Schmitz, P., and Thompson, S. Behavioral reactivity and real time programming in xml: functional programming meets smil animation. In *Proceedings of the 2004 ACM symposium on Document engineering*, DocEng '04, ACM (New York, NY, USA, 2004), 57–66.
9. Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, IEEE Computer Society (Washington, DC, USA, 1996), 336–.
10. Tatsubori, M., and Suzumura, T. Html templates that fly: a template engine approach to automated offloading from server to client. In *Proceedings of the 18th international conference on World wide web*, WWW '09, ACM (New York, NY, USA, 2009), 951–960.
11. Vosloo, I., and Kourie, D. G. Server-centric web frameworks: An overview. *ACM Comput. Surv. 40*, 2 (May 2008), 4:1–4:33.
12. Wong, J., and Hong, J. Marmite: end-user programming for the web. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, ACM (New York, NY, USA, 2006), 1541–1546.

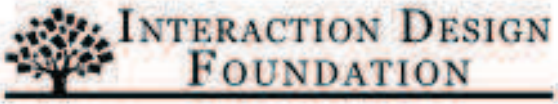
ORGANIZERS AND SPONSORS

Organizers

King Abdulaziz University

Université catholique de Louvain UCL

Sponsors



ISBN 978-84-616-4792-7