

DUI 2011

Distributed User Interfaces



CHI 2011 Workshop
Vancouver, BC
May 7th, 2011

CHI
2011

1st Workshop on Distributed User Interfaces 2011
Vancouver (BC, Canada) - May 7, 2011
ISE Research Group, University of Castilla-La Mancha, Spain
BCHI, Université catholique de Louvain, Belgium
HCI Lab, Miguel Hernández University of Elche, Spain

José A. Gallud, Ricardo Tesoriero, Jean Vanderdonckt, Victor M. R. Penichet,
Maria Lozano, Federico Botella (Eds.)
Edition prepared by Juan E. Garrido. Cover designed by Gabriel Sebastián (Photo: Shutterstock)
© Each author retains the rights to his/her work.

DL: AB-235-2011
ISBN 978-84-693-9829-6
Publisher: University of Castilla-La Mancha, Spain

DUI 2011

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. Unlike existing user interfaces, DUIs enable end users to distribute any user interface element, from the largest one to the smallest one, across different computing platforms and physical environments. This workshop invites authors to submit contributions on concepts discussion, applications, techniques or environments where DUIs play a central role by improving the interaction experience.

Table of Contents

Distributed User Interfaces: State of the Art Niklas Elmqvist	7
Approach for Enriching Distributed User Interfaces with Awareness Montserrat Sendín and Juan Miguel López	13
Munin: A Peer-to-Peer Middleware for Ubiquitous Visualization Spaces Niklas Elmqvist	17
Distributed Dynamic Peephole Interfaces for Mobile Map Navigation on Projector Phones Markus Löchtefeld, Sven Gehring and Antonio Krüger	21
Distributed User Interfaces for Smart Environments: Characteristics and Challenges Marco Blumendorf, Dirk Roscher and Sahin Albayrak	25
Distribution Primitives for Distributed User Interfaces Jérémie Melchior, Jean Vanderdonckt and Peter Van Roy	29
Humane Machine Interaction Design: DUIs Design Thinking for Social Innovation Niki Lambropoulos and Shopi Danis	33
ZOIL: A Design Paradigm and Software Framework for Post-WIMP Distributed User Interfaces Michael Zöllner, Hans-Christian Jetter and Harald Reiterer	37
Improving E-Learning Using Distributed User Interfaces Habib M. Fardoun, Sebastián Romero López and Pedro G. Villanueva	41
Lessons Learned from the Design and Implementation of Distributed Post-WIMP User Interfaces Thomas Seifried, Hans-Christian Jetter, Michael Haller and Harald Reiterer	45
A Model for DUI Development Anders Fröberg, Henrik Eriksson and Erik Berglund	49
Multi-touch Collaborative DUI to create Mobile Services Gabriel Sebastián, Pedro G. Villanueva, Ricardo Tesoriero and José A. Gallud	53
Using Multi-touch Technologies to Perform Collaborative Map Exploration Pedro G. Villanueva, Ricardo Tesoriero, Gabriel Sebastián and José A. Gallud	57
Activity-Based Computing – Metaphors and Technologies for Distributed User Interfaces Jakob Bardram, Afsaneh Doryab and Sofiane Gueddana	61
Distributed interactive surfaces using tangible and virtual objects Sophie Lepreux, Sébastien Kubicki, Christophe Kolski and Jean Caelen	65
Dual Displays: towards an interaction model and associated design guidelines Nima Kaviani, Matthias Finke, Rodger Lea and Sidney Fels	69

Distributed User Interface for a Multi-Tablet Active Reading System Nicholas Chen, Francois Guimbretiere and Abigail Sellen	73
Supporting Distributed Decision Making Using Secure Distributed User Interfaces Thomas Barth, Thomas Fielenbach, Mohamed Bourimi, Dogan Kesdogan and Pedro G. Villanueva	77
Co-Interactive Table: a New Facility Based on Distributed User Interfaces to Improve Collaborative Meetings Elena de la Guía, María Lozano and Víctor M. R. Penichet	81
Improving Ubiquitous Environments through Collaborative Features Juan Enrique Garrido, Victor M. R. Penichet and María D. Lozano	85
Drag&Share: A Shared Workspace for Synchronous Collaboration Félix Albertos Marco, Victor M. R. Penichet and José A. Gallud	89
Distributing User Interfaces with MARIA Marco Manca and Fabio Paternò	93
A Formal View of Distributed User Interfaces J. J. López-Espin, J. A. Gallud, E. Lazcorreta, A. Peñalver and F. Botella	97
Distributed User Interfaces in Ambient Intelligent Environments: A Tale of Three Studies Pavan Dadlani, Jorge Peregrin Empananza and Panos Markopoulos	101
Visually Augmented Interfaces for Improving Awareness in Mobile Collaboration Barret Ens, Rasit Eskicioglu and Pourang Irani	105

Distributed User Interfaces: State of the Art

Niklas Elmqvist
Purdue University
elm@purdue.edu

ABSTRACT

In this paper we summarize the state of the art in the field of distributed user interfaces (DUIs). Some of the topics surveyed include pervasive and ubiquitous computing, migratory and migratable interfaces, plasticity and adaptivity in interaction, and applications to multi-display and multi-surface environments. Our purpose is to provide a solid foundation for future research in distributed user interfaces.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Interaction styles*

Author Keywords

DUIs, multi-display environments, ubiquitous computing.

INTRODUCTION

As we draw nearer to Mark Weiser’s vision for the computer of the new century [73], it is time to marshal the existing research conducted in all of the subareas of human-computer interaction (HCI) that allow computer interfaces to be distributed across multiple devices, multiple users, and multiple platforms; in the same space or in different geographical spaces; and at the same time or at different points in time. In this paper, we first draw on previous work to define this new field of **distributed user interfaces** (DUIs). We then briefly summarize research that fit this definition. We hope that this survey will serve as a platform for any researcher, practitioner, or student interested in working in this new area.

DEFINITION: DISTRIBUTED USER INTERFACES

Although the term “distributed user interface” or “DUI” has been used regularly in several publications since a few years back (notably [4, 19, 37, 68, 67]), to our knowledge the term has yet to be formally defined. Synthesizing across all of the informal definitions in earlier work, we get the following:

*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.*

We define the above five distribution dimensions as follows:

- **Input (I).** Managing input on a single computational device, or distributed across several different devices (so-called *input redirection* [30, 42, 72]).
- **Output (O).** Graphical output tied to a single device (display), or distributed across several devices (so-called *display* or *content redirection* [12, 62, 72]).
- **Platform (P).** The interface executes on a single computing platform, or distributed across different platforms (i.e., architectures, operating systems, networks, etc).
- **Space (S).** The interface is restricted to the same physical (and geographic) space, or can be distributed geographically (i.e., co-located or remote interactive spaces [2]).
- **Time (T).** Interface elements execute simultaneously (synchronously), or distributed in time (asynchronously).

Because the focus of our definition is on the user interface technology itself, and not collaboration, we do not include the users as a distribution dimension. In other words, whether or not a DUI is used by a single user or multiple users is not pertinent to our definition. Furthermore, while some existing DUI definitions (such as [68]) restrict the interface components of a DUI to exist in the same, co-located space, our definition is more general and adopts the space/time dimensions of general CSCW groupware [2].

STATE OF THE ART

Here we survey the state of the art in DUIs: research projects that distribute one or several of these dimensions.

Migratory and Migratable Interfaces

One of the earliest distributed user interface concepts was Krishna’s and Cardelli’s work on *migratory applications* [8], which are applications capable of roaming freely on the network instead of being confined to a particular computer. However, the original idea is limited to migrating whole applications between hosts running the same operating system.

More recent work on *migratable user interfaces* [25] remove these constraints by allowing distribution (through migration) at an interface component level. Similarly, Bandelloni and Paterno [5] use a migration server to replicate the run-time state and adapt the interface a device, whereas Mori et al. [40] present a model-based approach for migrating UIs.

Plastic and Multi-Target Interfaces

Almost the first issue that arises when migrating an application from one device to another with different input and output capabilities is how to adapt the application interface

to the new device [8]. Thevenin and Coutaz [66] named this concept the *plasticity* of a user interfaces and defined it as the capacity of a UI to withstand variations in both the device and its physical environment while preserving usability.

Given that usability requirements are satisfied, the interface should ideally have to be specified only once and then instantiated multiple times depending on physical variations. Calvary presents a reference framework for such multi-target user interfaces [15]. The work by Mori et al. [40] employs a logical model-based approach to describing interfaces without having to deal with low-level implementation details.

Usability is a key constraint, but is difficult to validate. Aquino et al. [1] present an evaluation method for multi-target user interfaces based on model-driven engineering [1]. This approach will test different features of a generated user interface depending on the actual platform used.

Multi-Device Environment Models

At the core of distributed user interfaces is the physical environment—consisting of multiple, heterogeneous, and distributed devices, displays, and surfaces—where the DUI runs. One of the early terms for such spaces was computer-augmented environments [75] or multi-computer environments [50]. Alternative terms include multi-display environment (MDE) [58] as well as multi-surface environment [17]. We use the unified term *multi-device environment* here.

A number of theoretical frameworks and conceptual models for MDEs have been proposed. Grudin discusses the utility of multiple monitors for partitioning digital worlds [26]. Coutaz et al. [17] presents an ontology for multi-surface interaction distributed in time and space. The office of the future [49] project attempts to augment an office-sized room with displays using geometrically-corrected projectors, structured light, and camera-based tracking. Terrenghi et al. [65] define a taxonomy for multi-device ecosystems for multiple participants. Hybrid user interfaces [22] use a see-through head-mounted display to expand the screens of standard desktop computers into the air surrounding it. This idea of **appropriating** surfaces in the physical world has lately resurfaced as interaction devices shrink in size [29].

Multi-Device Environments

The literature on actual implementations of multi-device environments is very rich and spans a wide variety of fields such as CSCW, HCI, and even computer graphics. Perhaps the earliest example of an MDE was the Spatial Data Management System (SDMS) [13], which was a multimodal interface system supporting gesture and voice commands to control spatial information on a multi-display setup (a wall-sized projection display and a small touch display).

The late 80s and early 90s were characterized by significant advances in both theory and technology for MDEs. The Co-Lab meeting room [58] introduced concepts of multi-user interfaces, multi-display environments, and WYSIWIS (what you see is what I see). Wellner's DigitalDesk [74] system was one of the first tabletop displays, and combined an augmented desk, a camera, a tablet, and a projector. Another seminal work was the experimental i-LAND [59] system

that integrated several computational resources in the same physical space into what became known as a roomware system [60, 61]. The i-LAND implementation included a display wall, an interactive table, and chairs with embedded computers for either solitary or collaborative work.

Mobile and handheld devices have long been prime candidates for integrating with stationary computers and large displays. The PDA-ITV dual-device system [53] was an early multi-display environment consisting of a PDA and an interactive TV. The Pebbles framework was intended for combining handhelds with PCs [42, 41]. The WebSplitter system supports multi-device and multi-user web browsing using an XML-based configuration system [28]. WallShare [70] allows multiple participants, each with a mobile device, to collaborate on a shared wall display. Similarly, laptops share the same properties of mobile devices because they are portable. The Augmented Surfaces [51] workspace uses preinstalled projected displays and camera-based tracking to allow moving entities between different laptops in a shared space, such as a conference room. In contrast, the IMPROMPTU [12] framework uses off-the-shelf computers and software to support collaborative software development.

Some projects build entire physical spaces consisting of a wide range of heterogeneous displays and devices. The Stanford Interactive Workspaces project [23, 31, 32] proposes novel methods for collaboration in technology-rich physical spaces; the concrete instantiation of an interactive workspace was called iRoom. However, instead of building a dedicated interactive space from the ground up, another approach is to augment an existing physical space. The office of the future project [49] uses the intrinsic geometry of an office to turn all surfaces into displays. In a similar fashion, the Everywhere Displays projector uses a rotating mirror to direct the light from an LCD/DLP projector onto different surfaces in an environment [47]. Even if not using projected displays, the surfaces of different displays in the same physical environment must be corrected for perspective depending on the location of the viewer; this is the basic idea behind the E-conic [45] multi-display multi-user environment.

Recent projects on multi-device environments tend to be table-centric in that they often incorporate a large, horizontal interaction surface as the centerpiece of the environment. This is generally because research has shown that tabletops support a more participative collaboration style than wall-mounted (vertical) displays [54]. As a case in point, the UbiTable [56] is a horizontal “scrap display” for impromptu information sharing during face-to-face collaboration. MultiSpace [21] uses a table-as-hub design philosophy for managing electronic documents across tables, walls, and mobile devices. Similarly, the WeSpace [76] environment combines a large vertical display with a multitouch tabletop display for collaborative work and information sharing. The system transparently integrates personal laptops with the shared tabletop display in a way that permits casual usage.

Multi-Device Interaction Techniques

A key component of a multi-device environment is how to manage interaction. The general methodology for MDE in-

teraction is to use *input redirection* [30] where the input events from one device are sent to another device in the environment. This notion was first introduced by Rekimoto as multi-computer direct manipulation [50]; the same paper also introduced the pick-and-drop technique for effortless information transfer between different computers.

The most common multi-device interaction technique is multi-display pointers that jump across multiple displays [43]. The PointRight [30] technique utilizes knowledge of the topology of the interactive workspace to redirect pointer and keyboard input to different displays. Mighty Mouse [14] uses VNC to redirect mouse input across different sessions on different computers and different platforms. The mouse ether [6] approach adapts mouse move events between different displays in a multi-device environment. The multi-monitor mouse [7] warps the mouse cursor between adjacent screens in a multi-display environment. For input in heterogeneous MDEs, the perspective cursor [46] corrects the mouse cursor depending on the perspective from the user's viewpoint. Finally, ninja cursors [34] replicate the mouse cursor into several ones arranged in a regular grid to improve pointing performance on large displays.

Not all multi-device interaction is indirect control of a cursor. Guimbréiere et al. [27] present fluid interaction techniques for directly interacting with wall-sized displays. DataTiles [52] are physical tiles made from transparent acrylic that can be used to bridge graphical and virtual interaction on tabletop displays. The XWand [77] is a wireless device for natural interaction with intelligent environments.

As in much HCI research, evaluation plays a pivotal role in determining the optimal interaction method for multi-display interaction. Nacenta et al. [43] evaluates different techniques for multi-display reaching, finding a speed advantage for those with a uniform control-to-display ratio. Other work derives the optimal method—simply stitching displays together—for dealing with the displayless space between physical displays [44]. Finally, Wallace et al. [72] compare input and content redirection for multi-display environments using Swordfish [71] (an MDE framework supporting lightweight personal bindings between displays). Their results indicate that content redirection suffers less from performance loss due to suboptimal seating positions.

Application and Content Redirection

If input redirection is one key aspect of multi-device environments, then content (output) redirection is another. In fact, studies show that content redirection is a more powerful concept than input redirection for multi-user settings [72].

Allowing the user to control the content redirection is an important feature and a myriad of different schemes have been proposed. The ICrafter [48] framework can be used to define and build services and their user interfaces in ubiquitous computing environments. The ARIS [9] space window manager allows for controlling applications across heterogeneous devices using an iconic map of the physical space to enable the user to visually relocate applications (content) and input. A novel toolset [10] supports explicitly creating such iconic interfaces for managing content.

Biehl and Bailey [11] evaluate three different application management approaches—textual, map, and iconic—in multi-display/device environments [11]. Their results indicate that the iconic interface was fastest and most preferred by users. Other work uses simulation in a virtual reality engine for prototyping the physical and temporal distribution of interface components for a distributed user interface [68].

Distributed User Interface Models

A few recent reference models refer explicitly to the term “distributed user interface.” First of these reference models was CAMELEON-RT, a middleware software infrastructure for distributed, migratable, and plastic interfaces [4, 16]. Demeure et al. [18] proposed another model in 2005, and later refined it into the 4C model in 2008 [19]. The 4C model consists of four basic components—computation, communication, coordination, and configuration—that captures the what, when, who, and how aspects of the distribution.

Another approach is to apply software engineering to DUI models. As early as 1996, Graham et al. [24] proposed a distributed version of the model-view-controller (MVC) paradigm. Vandervelpen and Coninx [69] apply model-based methods to user interface design, similar to Mori et al. [40]) but specifically targeted at heterogeneous device environments. Luyten and Coninx [36] also target such environments, but take a bottom-up approach focused on designing user interface elements that support seamless distribution. Finally, the recent views, instruments, governors, and objects (VIGO) model [33] can be used to build distributed ubiquitous instrumental interaction applications.

Ubiquitous Computing

Finally, DUIs also touch upon ubiquitous computing, where data and computation is integrated into everyday objects and activities. A workshop was held at CHI 2001 on distributed and disappearing interfaces in ubiquitous computing [20].

Some research in this domain focus on the hardware. The ConnecTable is a table-centric information appliance for seamless coupled/decoupled collaboration [64]. iStuff is a physical UI toolkit for UbiComp that incorporates a wide range of physical components to be used in an interactive workspace [3]. Similarly, the u-Texture [35] physical panel can be used to effortlessly build smart environments.

While hardware aspects are important, the software architecture is critical in ubiquitous environments. The BEACH system [63] is one such infrastructure and was used for the ConnecTable project. Aura [57] is a software architecture for supporting dynamic variability of computational resources in UbiComp environments. Similarly, the Gaia [55] middleware infrastructure supports resource management in physical and interactive computing spaces (called Active Spaces). Additional frameworks include MediaBroker [38, 39] and a recent toolkit for building peer-to-peer DUIs [37].

CONCLUSIONS

We have presented a definition and an exhaustive survey of the emerging field of *distributed user interfaces*. Our treatment connects a wide range of HCI topics, including perva-

sive and ubiquitous computing, multi-display environments, and multi-device interaction. We believe this treatment will help solidify terminology and theory in DUIs and serve as a useful foundation for both new and established researchers, students, and practitioners in this novel area of research.

REFERENCES

1. N. Aquino, J. Vanderdonckt, N. Condori-Fernández, Ó. D. Tubío, and O. Pastor. Usability evaluation of multi-device/platform user interfaces generated by model-driven engineering. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, 2010.
2. R. M. Baecker. *Readings in Groupware and Computer-Supported Cooperative Work*. Morgan Kaufmann Publishers, San Francisco, 1993.
3. R. Ballagas, M. Ringel, M. Stone, and J. Borchers. iStuff: a physical user interface toolkit for ubiquitous computing environments. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 537–544, 2003.
4. L. Balme, A. Demeure, N. Barralon, J. Coutaz, and G. Calvary. CAMELEON-RT: A software architecture reference model for distributed, migratable, and plastic user interfaces. In *Proceedings of the Symposium on Ambient Intelligence*, volume 3295 of *Lecture Notes in Computer Science*, 291–302. Springer, 2004.
5. R. Bandelloni and F. Paterno. Flexible interface migration. In *Proceedings of the ACM Conference on Intelligent User Interfaces*, 148–155, 2004.
6. P. Baudisch, E. Cutrell, K. Hinckley, and R. Gruen. Mouse ether: accelerating the acquisition of targets across multi-monitor displays. In *Extended Abstracts of the ACM CHI Conference on Human Factors in Computing Systems*, 1379–1382, 2004.
7. H. Benko and S. Feiner. Multi-monitor mouse. In *Extended Abstracts of the ACM CHI Conference on Human Factors in Computing Systems*, 1208–1211, 2005.
8. K. A. Bharat and L. Cardelli. Migratory applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 133–142, 1995.
9. J. T. Biehl and B. P. Bailey. ARIS: An interface for application relocation in an interactive space. In *Proceedings of the Graphics Interface Conference*, 107–116, 2004.
10. J. T. Biehl and B. P. Bailey. A toolset for creating iconic interfaces for interactive workspaces. In *Proceedings of IFIP INTERACT*, volume 3585 of *Lecture Notes in Computer Science*, 699–712. Springer, 2005.
11. J. T. Biehl and B. P. Bailey. Improving interfaces for managing applications in multiple-device environments. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, 35–42, 2006.
12. J. T. Biehl, W. T. Baker, B. P. Bailey, D. S. Tan, K. M. Inkpen, and M. Czerwinski. IMPROMPTU: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 939–948, 2008.
13. R. A. Bolt. “Put-That-There”: voice and gesture at the graphics interface. *Computer Graphics (SIGGRAPH ’80 Proceedings)*, 14(3):262–270, July 1980.
14. K. S. Booth, B. D. Fisher, C. J. R. Lin, and R. Argue. The ‘mighty mouse’ multi-screen collaboration tool. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 209–212, 2002.
15. G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308, 2003.
16. J. Coutaz, L. Balme, C. Lachenal, and N. Barralon. Software infrastructure for distributed migratable user interfaces. In *Proceedings of the UbiHCISys Workshop on UbiComp*, 2003.
17. J. Coutaz, C. Lachenal, and S. Dupuy-Chessa. Ontology for multi-surface interaction. In *Proceedings of IFIP INTERACT*, 447–454. Springer, 2003.
18. A. Demeure, G. Calvary, J.-S. Sottet, and J. Vanderdonckt. A reference model for distributed user interfaces. In *Proceedings of the International Workshop on Task Models and Diagrams for User Interface Design*, 79–86, 2005.
19. A. Demeure, J.-S. Sottet, G. Calvary, J. Coutaz, V. Ganneau, and J. Vanderdonckt. The 4C reference model for distributed user interfaces. In *Proceedings of the International Conference on Autonomic and Autonomous Systems*, 61–69, 2008.
20. A. K. Dey, P. Ljungstrand, and A. Schmidt. Workshop: Distributed and disappearing user interfaces in ubiquitous computing. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 487–488, 2001.
21. K. Everitt, C. Shen, K. Ryall, and C. Forlines. MultiSpace: enabling electronic document micro-mobility in table-centric, multi-device environments. In *Proceedings of IEEE Tabletop*, 27–34, 2006.
22. S. Feiner and A. Shamash. Hybrid user interfaces: Breeding virtually bigger interfaces for physically smaller computers. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 9–17, 1991.
23. A. Fox, B. Johanson, P. Hanrahan, and T. Winograd. Integrating information appliances into an interactive workspace. *IEEE Computer Graphics and Applications*, 20(3):54–65, May/June 2000.
24. T. C. N. Graham, T. Urnes, and R. Nejabi. Efficient distributed implementation of semi-replicated synchronous groupware. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 1–10, 1996.
25. D. Grolaux, P. V. Roy, and J. Vanderdonckt. Migratable user interfaces: Beyond migratory interfaces. In *Proceedings of the IEEE/ACM Conference on Mobile and Ubiquitous Systems*, 422–430, 2004.

26. J. Grudin. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 458–465, 2001.
27. F. Guimbretière, M. C. Stone, and T. Winograd. Fluid interaction with high-resolution wall-size displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 21–30, 2001.
28. R. Han, V. Perret, and M. Naghshineh. WebSplitter: A unified XML framework for multi-device collaborative web browsing. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, 221–230, 2000.
29. C. Harrison. Appropriated interaction surfaces. *IEEE Computer*, 43(6):86–89, 2010.
30. B. Johanson, G. Hutchins, T. Winograd, and M. Stone. PointRight: experience with flexible input redirection in interactive workspaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 227–234, 2002.
31. B. Johanson, T. Winograd, and A. Fox. Interactive workspaces. *IEEE Computer*, 36(4):99–101, 2003.
32. B. Johanson, T. Winograd, and A. Fox. Invisible computing: Interactive workspaces. *IEEE Computer*, 36(4):99–103, 2003.
33. C. N. Klokmoose and M. Beaudouin-Lafon. VIGO: instrumental interaction in multi-surface environments. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 869–878, 2009.
34. M. Kobayashi and T. Igarashi. Ninja cursors: using multiple cursors to assist target acquisition on large screens. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 949–958, 2008.
35. N. Kohtake, R. Ohsawa, T. Yonezawa, Y. Matsukura, M. Iwai, K. Takashio, and H. Tokuda. u-Texture: self-organizable universal panels for creating smart surroundings. In *Proceedings of the International Conference on Ubiquitous Computing*, volume 3660 of *Lecture Notes in Computer Science*, 19–36. Springer, 2005.
36. K. Luyten and K. Coninx. Distributed user interface elements to support smart interaction spaces. In *Proceedings the IEEE International Symposium on Multimedia*, 277–286, 2005.
37. J. Melchior, D. Grolaux, J. Vanderdonckt, and P. V. Roy. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In *Proceedings of the ACM Symposium on Engineering Interactive Computing System*, 69–78, 2009.
38. M. Modahl, B. Agarwalla, G. D. Abowd, U. Ramachandran, and T. S. Saponas. Toward a standard ubiquitous computing framework. In *Proceedings of the Workshop on Middleware for Pervasive and Ad-hoc Computing*, 135–139, 2004.
39. M. Modahl, I. Bagrak, M. Wolenetz, P. W. Hutto, and U. Ramachandran. MediaBroker: An architecture for pervasive computing. In *Proceedings of the IEEE Conference on Pervasive Computing*, 253–262, 2004.
40. G. Mori, F. Paternò, and C. Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Transactions on Software Engineering*, 30(8):507–520, 2004.
41. B. A. Myers, J. Nichols, J. O. Wobbrock, and R. C. Miller. Taking handheld devices to the next level. *Computer*, 37(12):36–43, 2004.
42. B. A. Myers, H. Stiel, and R. Gargiulo. Collaboration using multiple PDAs connected to a PC. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, 285–294, 1998.
43. M. A. Nacenta, D. Aliakseyeu, S. Subramanian, and C. Gutwin. A comparison of techniques for multi-display reaching. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 371–380, 2005.
44. M. A. Nacenta, R. L. Mandryk, and C. Gutwin. Targeting across displayless space. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 777–786, 2008.
45. M. A. Nacenta, S. Sakurai, T. Yamaguchi, Y. Miki, Y. Itoh, Y. Kitamura, S. Subramanian, and C. Gutwin. E-conic: a perspective-aware interface for multi-display environments. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 279–288, 2007.
46. M. A. Nacenta, S. Sallam, B. Champoux, S. Subramanian, and C. Gutwin. Perspective cursor: perspective-based interaction for multi-display environments. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 289–298, 2006.
47. C. Pinhanez. The everywhere displays projector: A device to create ubiquitous graphical interfaces. *Lecture Notes in Computer Science*, 2201:315–??, 2001.
48. S. Ponnekanti, B. Lee, A. Fox, P. Hanrahan, and T. Winograd. ICrafter: A service framework for ubiquitous computing environments. In *Proceedings of the International Conference on Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, 56–75. Springer, 2001.
49. R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. *Computer Graphics (SIGGRAPH '98 Proceedings)*, 32:179–188, Aug. 1998.
50. J. Rekimoto. Pick-and-drop: A direct manipulation technique for multiple computer environments. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 31–40, 1997.
51. J. Rekimoto and M. Saitoh. Augmented surfaces: A spatially continuous work space for hybrid computing environments. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 378–385, 1999.

52. J. Rekimoto, B. Ullmer, and H. Oba. DataTiles: a modular platform for mixed physical and graphical interactions. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 269–276, 2001.
53. S. Robertson, C. Wharton, C. Ashworth, and M. Franzke. Dual device user interface design: PDAs and interactive television. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 79–86, 1996.
54. Y. Rogers and S. E. Lindley. Collaborating around vertical and horizontal large interactive displays: which way is best? *Interacting with Computers*, 16(6):1133–1152, 2004.
55. M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. Gaia: A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74–83, 2002.
56. C. Shen, K. Everitt, and K. Ryall. UbiTable: impromptu face-to-face collaboration on horizontal interactive surfaces. In *Proceedings of the International Conference on Ubiquitous Computing*, volume 2864 of *Lecture Notes in Computer Science*, 281–288. Springer, 2003.
57. J. P. Sousa and D. Garlan. Aura: an architectural framework for user mobility in ubiquitous computing environments. In *Proceedings of the IEEE/IFIP Conference on Software Architecture*, 29–43, 2002.
58. M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. G. Tatar. WYSIWIS revised: Early experiences with multiuser interfaces. *ACM Transactions on Office Information Systems*, 5(2):147–167, 1987.
59. N. A. Streitz, J. Geissler, T. Holmer, S. Konomi, C. Müller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. i-LAND: An interactive landscape for creativity and innovation. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 120–127, 1999.
60. N. A. Streitz, P. Rexroth, and T. Holmer. Does ‘roomware’ matter? investigating the role of personal and public information devices and their combination in meeting room collaboration. In *Proceedings of the European Conference on Computer-Supported Cooperative Work*, 297–312, 1997.
61. N. A. Streitz, P. Tandler, and C. Müller-Tomfelde. *Human-Computer Interaction in the New Millenium*, chapter Roomware: Towards the Next Generation of Human-Computer Interaction based on an Integrated Design of Real and Virtual Worlds, 553–578. Addison Wesley, 2001.
62. D. S. Tan, B. Meyers, and M. Czerwinski. WinCuts: manipulating arbitrary window regions for more effective use of screen space. In *Extended Abstracts of the ACM CHI Conference on Human Factors in Computing Systems*, 1525–1528, 2004.
63. P. Tandler. Software infrastructure for ubiquitous computing environments: Supporting synchronous collaboration with heterogeneous devices. *Lecture Notes in Computer Science*, 2201:96–115, 2001.
64. P. Tandler, T. Prante, C. Müller-Tomfelde, N. Streitz, and R. Steinmetz. ConnecTables: Dynamic coupling of displays for the flexible creation of shared workspaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 11–20, 2001.
65. L. Terrenghi, A. J. Quigley, and A. J. Dix. A taxonomy for and analysis of multi-person-display ecosystems. *Personal and Ubiquitous Computing*, 13(8):583–598, 2009.
66. D. Thevenin and J. Coutaz. Plasticity of user interfaces: Framework and research agenda. In *Proceedings of IFIP INTERACT*, 110–117, 1999.
67. J. Vanderdonckt. Distributed user interfaces: How to distribute user interface elements across users, platforms, and environments. In *Proceedings of the International Conference on Interaction*, 2010.
68. J. Vanderdonckt, H. Mendonca, and J. P. M. Massó. Distributed user interfaces in ambient environment. *Communications in Computer and Information Science*, 11(3):121–130, 2008.
69. C. Vandervelpen and K. Coninx. Towards model-based design support for distributed user interfaces. In *Proceedings of the Nordic Conference on Human-Computer Interaction*, 61–70, 2004.
70. P. G. Villanueva, R. Tesoriero, and J. Gallud. Multi-pointer and collaborative system for mobile devices. In *Proceedings of the ACM Mobile HCI Conference*, 435–438, 2010.
71. J. Wallace, V. Ha, R. Ziola, and K. Inkpen. Swordfish: user tailored workspaces in multi-display environments. In *Extended Abstracts of the ACM CHI Conference on Human Factors in Computing Systems*, 1487–1492, 2006.
72. J. R. Wallace, R. L. Mandryk, and K. M. Inkpen. Comparing content and input redirection in MDEs. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 157–166, 2008.
73. M. Weiser. The computer for the twenty-first century. *Scientific American*, 3(265):94–104, Sept. 1991.
74. P. Wellner. Interacting with paper on the DigitalDesk. *Communications of the ACM*, 36(7):86–96, July 1993.
75. P. Wellner, W. Mackay, and R. Gold. Introduction to the special issue on computer-augmented environments: Back to the real world. *Communications of the ACM*, 36(7):24, July 1993.
76. D. Wigdor, H. Jiang, C. Forlines, M. Borkin, and C. Shen. WeSpace: the design development and deployment of a walk-up and share multi-surface visual collaboration system. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 1237–1246, 2009.
77. A. Wilson and S. Shafer. XWand: UI for intelligent spaces. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 545–552, 2003.

Approach for Enriching Distributed User Interfaces with Awareness

Montserrat Sendín

GRIHO: HCI research group, University of Lleida
69, Jaume II St., 25001- Lleida, Spain
msendin@eps.udl.es

Juan Miguel López

College of Engineering,
University of the Basque Country
Nieves Cano 12, E-01006 Vitoria, Spain
juanmiguel.lopez@ehu.es

ABSTRACT

Last technological advances have brought drastic changes affecting the way interactive systems are conceived. Designers have to tackle the fact that UIs could be controlled by multiple end users on diverse computing platforms in assorted environments. Novel UIs enable end users to distribute any widget and piece of information across different contexts. However, these kinds of facilities acquire an added value when they are ruled by the well-known awareness. It is necessary to devise new mechanisms to support distributed UIs flexible enough to deal with the diversity of contexts and group concerns. Presented work provides a series of contributions in DUIs support, with respect to introducing awareness.

Author Keywords

Distributed User Interface, Awareness, Plasticity, Context

ACM Classification Keywords

D.2.13 Reusable Software, Reuse Models. D.2.11 Software Architectures, Domain-specific architectures. D.1.m Programming techniques. Miscellaneous

General Terms

Management, Design.

INTRODUCTION

Distributed User Interfaces (DUIs) enable end users to distribute in time and space any widget and piece of information across different contexts. Thus, designers have to tackle the fact that user interfaces (UIs henceforth) can be controlled by multiple users on diverse computing platforms in assorted environments. The need of introducing and supporting work in group becomes obvious. It is necessary to devise new mechanisms to support DUIs flexible enough to cope with the increasing diversity of contexts and able to deal with group concerns.

DUI development has been a research area of interest in

recent years. As relevant works in this field we can mention a few. Thus, [5] proposed a reference model for DUIs that examines them according to four dimensions: computation, communication, coordination, and configuration. It is called 4C framework for DUI. The work by [4] provides a system of interaction with multiple surfaces creating an environment for collaborative work. The UI is physically distributed over heterogeneous platforms, being reconfigurable dynamically allowing agents to migrate from one platform to another. In [8] a software tool for rapid prototyping of interactive systems is introduced, allowing UIs being distributed. This system provides designers with a means for generating ideas about how a UI can be distributed in a context of use, and also helps evaluation. Penichet et al. [10] present a novel approach and a methodology to gather requirements for groupware applications. Luyten et al. [7] present a task-centered approach to develop DUI for ambient intelligent environments based on the model-based approach. To support situated tasks distribution and context influences they introduce an environment model.

Except the work of [10], all of these approaches are mainly centered on physical characteristics of DUIs, but they do not consider *awareness*¹ elements. It is long recognized that successful group work is not simply the union of individual tasks, but an organized set of coherent activities with good strategies of communication, cooperation and coordination among group members [1]. Awareness reduces the meta-communicative effort needed to collaborate across physical distances in distributed environments and promotes real collaboration among group members [9]. The integration of these aspects in the development of DUIs is not an easy task. An ideal infrastructure for DUIs should provide the capacity to adapt interactive widgets and data to each particular context regarding, among others, the working group.

In this paper we present how the guidelines and software approach defined in the *Dichotomic View of Plasticity* can be applied to support DUIs that also consider and integrate group awareness elements. The aim is to provide an infrastructure to deal collaborative tasks in DUIs with

¹ Understanding of the others' activities, which provide a context for the own activity [6].

little or no extra effort for the UI designer. Resulting DUIs promote distributed interaction and real time coordination among remote users, contributing to real collaboration and a deeper understanding in multi-environment distributed scenarios. These contributions provide extensions to the previous infrastructure based on the *Dichotomic View of plasticity*, trying to integrate awareness in DUIs in an efficient and systematic way.

Next section explains abovementioned approach, taking a concrete groupware platform as an example of how the system works. Finally, some conclusions end the paper.

APPROACH FOR ENRICHING DISTRIBUTED USER INTERFACES WITH AWARENESS

Software Infrastructure for Supporting DUIs

Terms as mobility, heterogeneity and adaptation are important elements to be managed in DUIs. These challenges are collected under the term *plasticity*. Our approach, which is based on the *Dichotomic View of plasticity* [11], reuses and specializes some already existing plasticity tools to work in group. It consists in integrating and exploiting awareness by an existing infrastructure of plasticity, as an integral part that is embedded in the adaptation process. In other words, awareness information is incorporated in the characterization of the context of use. As a result, contextual information for these kinds of interfaces and the subsequent adaptation are considerably enriched and resulting interfaces provide the benefits from plasticity and awareness jointly. Next, in this section, the process to be carried out to provide awareness during execution, without neglecting plasticity is presented. Then, the system architecture operating in the server is described.

The first step to put in practice this approach is embedding a specific *Implicit Plasticity Engine* (IPE henceforth) on a given DUI on the client side. IPE is a runtime adaptive engine to provide runtime adaptations, and specific awareness mechanisms focused on promoting collaboration that enrich the default operative in the client side. Details on the software structure for the IPE, specially detailed for components aimed at supporting work in group can be consulted in [11]. However, we would like to mention the most important component of these kinds of components, the so-called *particular group-awareness* –pGA in Figure 1-, which represents the individual perceptions and particular understanding about the group from each working group member. It includes all the information generated during the performance of the combined tasks that could affect to whole group state. Additionally, it registers interactions of the user at hand with the other group members and also any data related to communication events and coordination actions.

Once the IPE has been embedded on the DUI at hand, we can proceed with its execution. The information in the *particular group-awareness* component will be kept

updated on the fly for further use. It can influence not only to local decisions, but also the evolution of the working group because each group member has to share his/her individual understanding with the group, by means of the server. In this line, the server gathers all the individual perceptions in a common group memory towards the construction of an overall perspective about the group (the so-called Shared-Knowledge Awareness [2] -SKA henceforth-).

The possibility of sharing all group perceptions makes possible to deduce global properties, as well as suitable considerations in benefit of the group. A reliable SKA allows exploiting overall group constraints and implications during the server inference process. Thereby, the client is responsible for communicating to the server any relevant change in the individual perception of the group circumstances by sending a request. Figure 1 shows the overview of the process described.

The server intercepts requests from group members and filters information regarding awareness. Following artificial intelligence foundations, it supports decision making by means of an inference engine specially prepared to derive some inferences about group constraint combinations. Thus, the server applies the necessary inferences in order to determine what adaptation is the most convenient for being applied in the client-side, according to the current group situation. During the inference process certain global properties and suitable considerations in benefit of the group can be deduced.

The results of the inference are returned to the group members' target platforms by a reply expressed as a set of adaptations in conformance to the global properties obtained, which are finally executed in the client.

Figure 2 displays how the entire server component works. In particular, the inference engine and the SKA conform the so-called *awareness manager* -(b) in Figure 2. The delivery system module -(a) in Figure 2- is in charge of preparing and delivering DUIs without groupware capacities, providing them with awareness elements. Henceforth it is explained how it works.

Once target DUI application is uploaded in the system, the IPE derivation tool proceeds to prepare it. First step is instantiating the *Implicit Plasticity Framework*² –IPF in Figure 2. As a result, a specific component for supporting work in group is obtained. Then, the system prepares and links this component together with the original DUI application. The result is the expected IPE, that is to say, the original DUI application now with awareness support embedded on it. Then, it is registered and made accessible in the repository of applications, so that the client is able to download and install it.

² A generic client-side application framework to obtain components for supporting work in group by a simple process of instantiation.[11].



Figure 1. Overview of the adaptation process in collaborative scenarios under the DVP.

It must be taken into account that the system does not perform this process in a fully automatic way. The IPE designer conducts the instantiation process, according not only to application's specific particularities, but also considering the adaptation requirements, the contextual information to be considered and finally the target computing platform or device where is being to be executed, completing thus a complete plasticity process. We are referring thus to a semi-automatic process.

Case Study

The entire infrastructure presented in this paper has been proved in a real case for a particular DUI. We are referring to Lucane, an open source groupware platform developed in Java programming language and designed with extensibility capabilities. This platform is based on client-server architecture, with the server being the responsible for the management of information among users.

In order to illustrate our approach in this case study we have selected one of the functionalities that have been incorporated to Lucane. It shows automatically in each involved group member's Calendar the foreseen end date of a task introduced in the TODO-List by a particular group member, provided other members of the group are implied in this task. Then, when this task is completed, a new event is automatically introduced in the Calendar to visualize the effective completion of the task. In the line of introducing visual awareness components, the idea is to distinguish situations in which the completion date is posterior to the foreseen date –marked using a red colour-, from the opposite situation –marked using a green colour-, being both cases visible in the Calendar. The visualization of these events by the entire group allows knowing the evolution of the rest of the members in their particular assignments, thus promoting the communication and coordination among them.

In order to manage all of this information, the awareness manager has to generate a complete historical of finalized tasks for every member, what implies registering a

complete log of each one of them. Once this information has been recorded, in order to encourage even more group members to complete their group tasks and to collaborate, we have introduced a distributed widget consisting in visualizing a ranking of what we have called level of compliance. By level of compliance we are referring to the personal compliance degree regarding the scheduled dates for tasks. The widget showing the level of compliance for each member is visualized in the main interface once the user accesses the system. The level of compliance is a global property inferred in the awareness manager that is distributed to each particular UI in each target platform. The management of this ranking related to the level of compliance helps to apply certain general considerations, such as rewarding the winner with certain privileges while using the system. This could imply an extra motivation and thus a benefit for the group. Moreover, as this kind of information is not related to user presence, it is appropriate independently of the real situation, that is, both in remote collaboration and face-to-face scenarios.

CONCLUSIONS

Generally speaking, work on DUIs is mainly centred on physical characteristics of DUIs, but not much attention has been paid to work in group concerns. DUIs can clearly be improved if they are ruled by awareness elements.

Presented approach supports adding group awareness capabilities to existing DUIs that do not handle work in group concerns in a completely transparent way regarding their core functionalities. Furthermore, the generic application framework and the infrastructure of the delivery system offer a certain level of systematization. Moreover, the application framework used in the instantiation process is independent of the underlying technology used in DUI development. Currently, it can operate both with Java developments and also with .NET based tools, considering both, mobile and desktop environments. Illustrated case study has been implemented in Java, and serves to prove the achievement of abovementioned advantages.

The approach, based on the *dichotomic view of plasticity*, follows a client-server distribution model, but instead of being server-centered, it provides a balanced strategy. This strategy can be synthesized by an operational balance between both sides, and the consequent reduction of networking dependences by providing two levels of awareness appropriately synchronized. We are referring to a trade-off between the degree of awareness and the network usage identified by [3].

As further work, we are preparing proposed solution to be applied in real peer-to-peer based platforms, in order to check its validity in these systems. In addition, we are currently evaluating how users manage a number of new group features added on Lucane using presented approach.

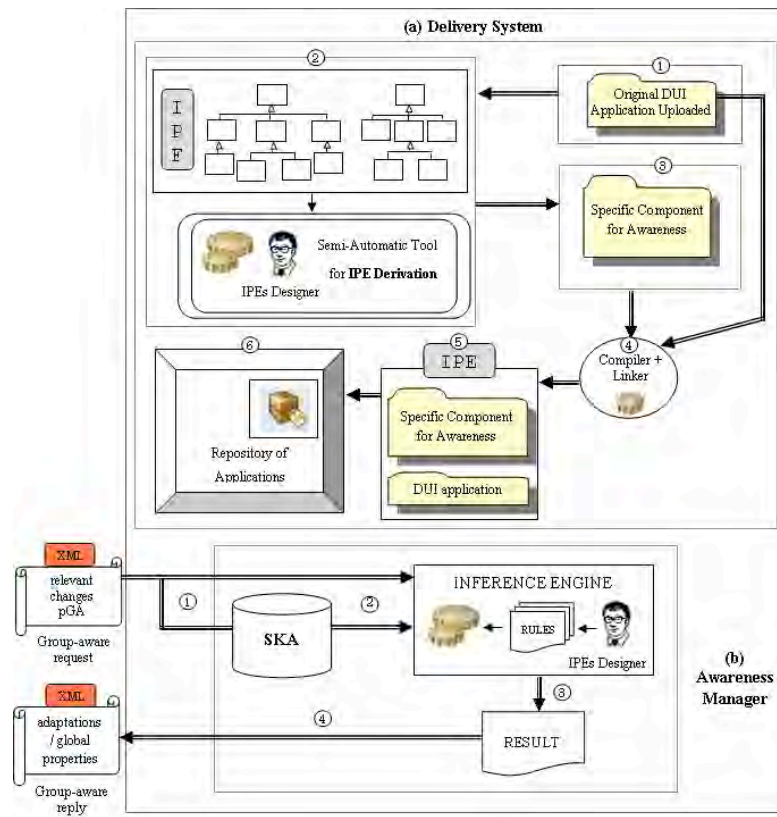


Figure 2. System architecture for the server.

ACKNOWLEDGMENTS

This work has been partially funded by Spanish Ministry of Science and Innovation through TIN2008-06228 and TIN2008-06596-C02-01 research projects.

REFERENCES

1. Alarcón, R.A., Guerrero, L.A., Ochoa, S.F., Pino, J.A.: Context in Collaborative Mobile Scenarios. *Workshop on Context and Groupware*, Vol. 133 (2005)
2. Collazos, C., Guerrero, L., Pino, J., Ochoa, S.: Introducing Shared-Knowledge- Awareness. *IASTED International Conference: Information and Knowledge Sharing*. (2002) 13-18
3. Correa, C., Marsic, I.: A Flexible Architecture to Support Awareness in Heterogeneous Collaborative Environments. *Proceedings of CTS 2003* (2003) 69-77
4. Coutaz, J., Lachenal, C., Calvary, G., Thevenin, D.: Software Architecture Adaptivity for Multisurface Interaction and Plasticity. *IFIP Workshop on Softw. Arch. Req. for CSCW-CSCW* (2000)
5. Demeure, A., et al. The 4C Reference Model for Distributed User Interfaces. *ICAS 2008*: 61-69
6. Dourish, P., Bellotti, V.: Awareness and Coordination in Shared Workspaces. *Proc. of the CSCW* (1992)
7. Luyten, K., et al. Designing distributed user interfaces for ambient intelligent environments using models and simulations. *Computers & Graphics* 30(5) (2006)
8. Molina Massó, J.P., et al. Rapid Prototyping Of Distributed User Interfaces. *CADUI 2006*: 151-166
9. Palfreyman, K.A., Rodden, T.: A protocol for user awareness on the World Wide Web. *Proc. of CSCW'96*, (1996) 130-139
10. Ruiz Penichet, et al. Requirement-based Approach for Groupware Environments Design. *Journal of Systems and Software* 83(8): 1478-1488 (2010)
11. Sendin, M., Collazos, C.A. Implicit Plasticity Framework: A Client-Side Generic Framework for Collaborative Activities, *CRIWG 2006, Lecture Notes on Computer Science. Groupware: Design, Implementation and Use*, V. 4154, 2006, pp: 219-227.

Munin: A Peer-to-Peer Middleware for Ubiquitous Visualization Spaces

Niklas Elmqvist
Purdue University
elm@purdue.edu

ABSTRACT

We present Munin, a software toolkit for building highly interactive visualization environments consisting of multiple input and output surfaces—such as tabletop displays, wall-mounted displays, and mobile devices—in the same physical space. Designed for ease of application development, the toolkit uses a peer-to-peer (P2P) architecture with shared state that is replicated across peers using an IP multicast protocol and accessed through a publish/subscribe mechanism. Input is managed through a shared event channel that allows input and output surfaces to be fully decoupled. The key strength and distinguishing factor of the toolkit is a shared table construct that is specifically designed for visualization applications. In this paper, we describe Munin’s general design principles, its system architecture, and give two concrete examples of how we are using it for data visualization.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Interaction styles*; I.3.6 Computer Graphics: Methodology and Techniques—*Interaction techniques*

Author Keywords

Distributed user interfaces, visualization, multi-display environments (MDEs), multi-surface environments.

INTRODUCTION

Visualization—the graphical representation of data to amplify cognition and facilitate understanding [4]—has long relied on standard personal computers equipped with mice, keyboards, and monitors for virtually all applications and purposes. However, this state of affairs is slowly changing for several reasons: (a) analysts are starting to tackle problems collaboratively as a team rather than alone [13]; (b) problems and datasets are growing in size, complexity, and time-sensitivity [21]; and (c) novel computing hardware, such as large wall-sized displays [8], tabletop displays [22], and reality-based [14] interaction, is opening up exciting new potential for the effective overview, analysis, and sense-

making of data. Now is the time to start leveraging all this potential into truly ubiquitous computing environments for visual reasoning, analysis, and sensemaking.

In this paper, we present **Munin**,¹ a peer-to-peer (P2P) distributed user interface (DUI) middleware for building such ubiquitous visual reasoning environments. The toolkit transforms all networked devices in the same physical environment into Munin *peers* that communicate using a multicast mechanism instead of a dedicated server. This increases the robustness and fault-tolerance of the system, decreases its coupling, and improves its ease and convenience of use. To achieve this, the toolkit is based on two main components:

- Shared and replicated **associative memory**, that contains shared objects to which peers can subscribe and publish.
- A common **event channel** where peers can produce and consume real-time events (for input and synchronization).

Naturally, the Munin toolkit is similar in scope and goals to many existing middleware solutions for distributed user interfaces, such as CAMELEON-RT [3], iStuff [2], and a peer-to-peer DUI toolkit [19]. However, Munin is unique in that it is tailored specifically for high-performance visualization: whereas most existing DUI systems tend to use a shared file system for distributing data between peers, Munin’s shared memory is designed essentially as a structured database where peers can create, upload, and modify entire datasets (relational data tables with schemas).

We have implemented Munin in Java to facilitate easy deployment across different devices, and the toolkit relies only on the JGroups [15] P2P multicast library for networking. This enables the toolkit to run on a range of devices such as smartphones, tablets, laptops, desktops, and all the way up to large tabletop displays as well as individual computers powering tiled-LCD display walls. We present two examples where we are using Munin to build software for a tiled display wall (Figure 2(a)) as well as for integrating mobile and tabletop devices (Figure 2(b)), and we anticipate building more advanced visual reasoning environments in the future.

RELATED WORK

Despite the prevalence of new computing hardware and the rising needs of the data analysis communities—professional and casual alike—the field of visualization is

¹Munin is one of the Norse god Odin’s twin ravens (Hugin is the other); serving as Odin’s eyes and ears, they fly out into Midgard every morning and return at night with tidings about the world.

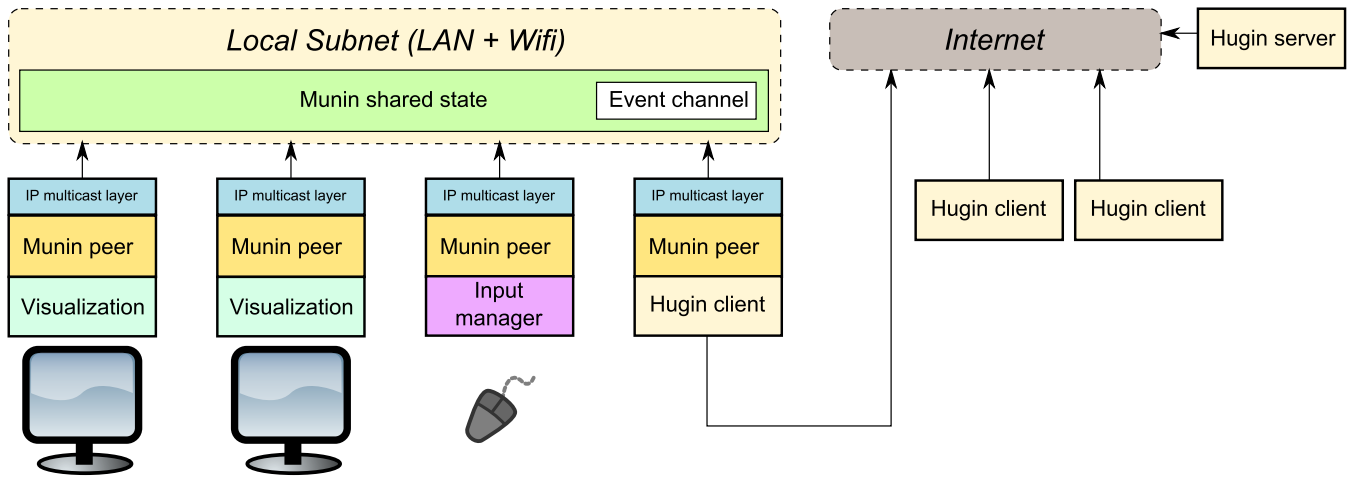


Figure 1. Munin system architecture with two display peers and one input peer. Hugin [16] is a toolkit for distributed visualization over the internet.

still largely confined to standard computing hardware. Existing work on distributed visualization on the network (such as WireGL [10], Chromium [11], and Equalizer [6]) remain mostly focused on desktop computers. While large displays have seen increasing use in visualization applications (e.g., [1, 8]), it is only recently that novel devices such as tabletop displays were adopted for visualization [12, 16].

THE MUNIN TOOLKIT

Munin is a Java toolkit for building distributed visualization software that executes on multiple computers located in the same physical space. These computers become Munin *peers*, and communicate using a decentralized peer-to-peer protocol implemented using JGroups [15]. The purpose of the toolkit is to enable developers to quickly and easily build *ubiquitous visualization spaces* where multiple surfaces, devices, displays, and modalities for input and output are combined into a single, coherent space for visual reasoning.

The Munin design philosophy is to optimize communication between peers and to delegate as much as possible of the data management and rendering to the peers themselves. For this reason, the toolkit does **not** support any form of streamed visualization; all peers are provided with the dataset (or a subset) to visualize, and are expected to render their own visual representations. This minimizes the coupling between devices and enables a robust, fault-tolerant, and truly decentralized architecture with no central server. The peers are also not required to run the exact same software, but may have customized software optimized for the rendering, input, and computational capabilities of each particular device.

System Architecture

Munin is based on a peer-to-peer (P2P) system architecture with no central server and with all state distributed among the peers (Figure 1). Input, output, application logic, and network communication can be independently managed by different peers, enabling the construction of truly *distributed* user interface (and visualization) applications. In the next few subsections, we will discuss the salient components of the toolkit, including (i) its shared state and data model, (ii)

its event channel, (iii) its governor subsystem, and (iv) its connection to the Hugin [16] toolkit for remote visualization.

Shared State

Munin uses a shared and replicated **associative memory**, inspired by earlier work on tuple spaces [7] and T-Spaces [18], that contains shared objects to which peers can subscribe and publish. This enables different peers to easily create, modify, and update shared state that is replicated across all of the devices and surfaces that constitute the visual environment.

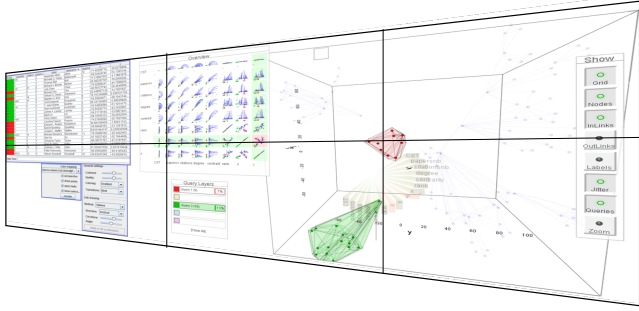
The Munin shared state consists of a single namespace addressed using UUIDs. Shared objects consist of dynamically typed properties, and can be created, read, modified, and deleted by any peer. They can also be serialized and deserialized for persistence across sessions. To support data sharing, Munin contains a special type of object called a *shared table*. Tables can be published and subscribed to like normal objects, but they enforce a column-centric data model with a common schema. Table state is represented by rows in the table, with each column having the same data type. Furthermore, to fully support common visualization design patterns [9], shared tables can be queried (for searching for specific entries in the data), cascaded (for creating views of the data), and linked (for combining several tables into one).

Event Channel

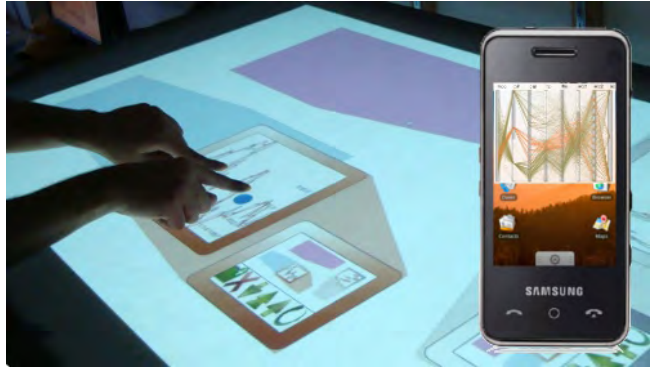
Munin incorporates a shared **event channel** where peers can produce and consume real-time events. Exposing the event system in the shared state allows for easily decoupling input and output subsystems, which is common in ubiquitous computing environments—consider, for example, a tiled display wall consisting of multiple computers responsible for output, but only a single computer managing gestural input from a Vicon motion-capture camera system.

Governors

All shared object properties in the Munin shared state may have attached governors (as in the VIGO [17] model). Governors do not encapsulate or control access to a property, but



(a) Tiled-LCD display wall.



(b) Tabletop display with a Samsung mobile device.

Figure 2. Envisionments of two ubiquitous visualization spaces that we are building using the Munin toolkit.

will respond to requests whether a property change is acceptable or not. The governor may also be invoked when the property is changed to execute any logic associated with the change. This provides a decoupled yet integrated way of building distributed user interface applications [17].

Remote Collaboration

There are many situations where analysts in one ubiquitous visualization space will want to connect to analysts in a visualization space in another location to collaborate remotely on a particular problem [13]. Munin is transparently integrated with our Hugin [16] framework for remote visualization by simply integrating Hugin clients (Hugin uses a client/server architecture) as Munin peers (Figure 1). Munin shared objects that should be exported to a remote site are simply tagged with reserved Hugin properties that signify the access control and ownership of a particular object.

APPLICATION EXAMPLES

To showcase the capabilities of Munin, we describe two example ubiquitous visualization spaces that we are building in our laboratory using the toolkit. Even though these two spaces are currently separate, our ultimate goal is to connect every single display and device in our laboratory into a single visualization environment. We see Munin as a future standard toolkit for building visualization software that scales seamlessly to multiple devices, displays, and surfaces.

Tiled-LCD Display Wall

High-resolution, wall-sized displays have been proven effective for viewing and manipulating large amounts of data [8]. This is largely because they support physical navigation [1] where the user can simply move around in the physical space to see an overview (by stepping back) or details (by walking forward) of the visual display. A common way of building such a wall-sized display is to tile several smaller displays, either projectors or monitors, into a regular grid and *stitching* together the images on each individual display to a single, integrated display space. This requires special software for distributed rendering such as WireGL [10], Chromium [11], or Equalizer [6]. However, these APIs typically enforce a centralized architecture to an application running on the

wall and requires it to be built in OpenGL, which makes retrofitting existing visualization applications difficult.

Following the Munin design philosophy, we are building a small (3×2) tiled-LCD display wall in our lab (see environment in Figure 2(a)) where individual computers (one to every two LCD monitors) run a visualization application as individual peers using the Munin toolkit for shared memory and input management. Because Munin, unlike Chromium and WireGL, does not rely on per-primitive distributed rendering, the visualization application is free to use any graphics API for the actual rendering. Each peer knows the configuration of the whole display wall as well as its own position, and will only render and consume events in that part of the visual space, entrusting the other parts to other peers.

While the physical setup is still a work-in-progress, the display wall software has been completed and can execute on a variable number of computers on the same local network (even in several different windows on the same computer). Using this software layer, we have built a simple drawing application in Java 2D that stitches a high-resolution image across a given physical arrangement of displays. A mouse (one for each computer powering the wall) can be used to control a multi-display cursor that can move between displays given this physical arrangement. These cursors can be used to draw on the image using simple pen, paintbrush, and airbrush tools with colors chosen from a common palette.

Tabletop and Mobile Device Integration

Another research project in our lab focuses on integrating mobile devices with a tabletop display to explicitly support flexible coupling [20] in collaborative work. The basic idea behind this project is simple: because mobile devices by their nature are personal and thus can be regarded as private, whereas a tabletop display is intrinsically shared and public, we want to explore an approach to coupling where the mobile device is used for loose coupling, and the tabletop for tight coupling. Both modes require real-time communication between the tabletop and the mobile devices: for synchronizing a mobile device with the tabletop, and for replicating visual representations across devices, respectively.

Our implementation uses an 81-inch DSI [5] multitouch

tabletop display that we have built in our lab (Figure 2(b)). The mobile devices are represented by three 7-inch Samsung Galaxy Tab multitouch tablets. Each collaborator is equipped with a personal tablet, and can join or leave the collaboration at any time. Both the tabletop display and the mobile devices use Munin to connect to the shared state and to manage input events and coordination. The tabletop software is built using Piccolo2D (<http://www.piccolo2d.org/>), whereas the Galaxy Tabs run software implemented using the Android SDK.

CONCLUSIONS AND FUTURE WORK

We have presented the Munin toolkit for building distributed user interfaces for powering special-purpose multi-surface environments that we call *ubiquitous visualization spaces*. Beyond supporting the basic functionality of existing middlewares for multi-surface environments, Munin is unique in that it uses a structured data model for its shared state that is specifically designed for high-performance visualization.

While the Munin toolkit software is almost feature-complete, this project remains very much a work in progress. In future improvements, we plan on modifying and refining the toolkit further as we build larger and more complex distributed user interface software that exercise the limitations of the toolkit. These new software will also serve as additional examples showcasing the utility of the toolkit.

REFERENCES

1. R. Ball, C. North, and D. A. Bowman. Move to improve: promoting physical navigation to increase user performance with large displays. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 191–200, 2007.
2. R. Ballagas, M. Ringel, M. Stone, and J. Borchers. iStuff: a physical user interface toolkit for ubiquitous computing environments. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 537–544, 2003.
3. L. Balme, A. Demeure, N. Barralon, J. Coutaz, and G. Calvary. CAMELEON-RT: A software architecture reference model for distributed, migratable, and plastic user interfaces. In *Proceedings of the Symposium on Ambient Intelligence*, volume 3295 of *Lecture Notes in Computer Science*, 291–302. Springer, 2004.
4. S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, San Francisco, 1999.
5. G. Çetin, R. Bedi, and S. Sandler, editors. *Multi-touch Technologies*. NUI Group, 2009. Online book - <http://www.nuigroup.com/>.
6. Equalizer. <http://www.equalizergraphics.com/>.
7. D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, Jan. 1985.
8. F. Guimbretière, M. C. Stone, and T. Winograd. Fluid interaction with high-resolution wall-size displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 21–30, 2001.
9. J. Heer and M. Agrawala. Software design patterns for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):853–860, 2006.
10. G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan. WireGL: A scalable graphics system for clusters. In *Computer Graphics (Proceedings of ACM SIGGRAPH)*, 129–140, 2001.
11. G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters. *ACM Transactions on Graphics*, 21(3):693–702, July 2002.
12. P. Isenberg and S. Carpendale. Interactive tree comparison for co-located collaborative information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1232–1239, 2007.
13. P. Isenberg, A. Tang, and M. S. T. Carpendale. An exploratory study of visual information analysis. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 1217–1226, 2008.
14. R. J. K. Jacob, A. Girouard, L. M. Hirshfield, M. S. Horn, O. Shaer, E. T. Solovey, and J. Zigelbaum. Reality-based interaction: a framework for post-WIMP interfaces. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 201–210, 2008.
15. JGroups. <http://www.jgroups.org/>.
16. K. Kim, W. Javed, C. Williams, N. Elmqvist, and P. Irani. Hugin: A framework for awareness and coordination in mixed-presence collaborative information visualization. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, 231–240, 2010.
17. C. N. Klokmoose and M. Beaudouin-Lafon. VIGO: instrumental interaction in multi-surface environments. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 869–878, 2009.
18. T. J. Lehman, S. W. McLaughry, and P. Wyckoff. T spaces: The next wave. In *Proceedings of the Hawaii International Conference on System Sciences*, 1999.
19. J. Melchior, D. Grolaux, J. Vanderdonckt, and P. V. Roy. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In *Proceedings of the ACM Symposium on Engineering Interactive Computing System*, 69–78, 2009.
20. A. Tang, M. Tory, B. Po, P. Neumann, and S. Carpendale. Collaborative coupling over tabletop displays. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 1181–1190, 2006.
21. J. J. Thomas and K. A. Cook, editors. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, 2005.
22. P. Wellner. Interacting with paper on the DigitalDesk. *Communications of the ACM*, 36(7):86–96, July 1993.

Distributed Dynamic Peephole Interfaces for Mobile Map Navigation on Projector Phones

Markus Löchtefeld, Sven Gehring and Antonio Krüger

German Research Center for Artificial Intelligence (DFKI)

Stuhlsatzenhausweg 3, 66123 Saarbrücken Germany

{markus.loechtefeld, sven.gehring, krueger}@dfki.de

ABSTRACT

Due to the miniaturization of projection technology it is now possible to integrate a projection unit into a mobile phone. These so called projector phones have the ability to project and explore large information spaces. On modern mobile phones the exploration of such information spaces is connected to panning and zooming, this could be eased with a projected interface. The size of the projection depends on the distance between the projection area and the projector phone. However, to create a projected area of a decent size one has to be at a certain distance from the projection. This distance makes direct interaction with the projection in a single user scenario impossible. This leads to a small-distributed user interface where on the one side is the device with in- and output capabilities and on the other side the large projection. In this paper, we present an interface design approach that tries to cover common problems that arise from the distributed user interface of a projector phone. Therefore we utilize the dynamic peephole metaphor. We introduce a prototype of a map application that implements the proposed design ideas and validates that the design simplifies mobile map navigation.

Author Keywords

Dynamic peephole, projector phones, distributed user interfaces.

INTRODUCTION

In the last years the miniaturization of projection technology has made a significant progress in the field of LED projectors as well as laser projectors. It is now possible to produce projection units that are small enough to be integrated into a mobile phone and power-saving enough to last for several hours. Such mobile phones with integrated projector units are called projector phones. Several projector phones are already commercially available such as e.g. the Samsung Beam or the LG eXpo.

The small size of mobile phones provides the convenience of mobility at the expense of reduced screen real estate such that the exploration of maps or websites on mobile devices is often connected with zooming and panning to find the desired part of the information space. Projector phones have the ability to extend the screen real estate with the projection and by that make such large amounts of information easily explorable at once, despite the small

form factor of the device. The state of the art projector phones that are available today only project a mirrored image of the devices normal screen. This leaves the focus of interaction on the device itself and does not empower the whole capabilities of projector phones. They are a new class of devices that deserve to have a custom tailored user interface to maximize their benefits for the user.

The disadvantage of the projection-based user interface of a projector phone is that direct manipulation is close to impossible. Due to physical constraints (e.g. required distance) the user is forced to be a certain distance away from the projection surface to create a larger projection. There is often no possibility reach the projection surface with one of the extremities while holding the projector phone and that is why indirect manipulation techniques are needed in single user scenarios. In such cases the user is facing a distributed user interface where parts of the interface remain on the display of the device while other parts are moved to the projection. In that case, an attention shift occurs between the projection and the controls for manipulation on the mobile device. Even though the distance and with that the distribution is rather small it is still a problem that restrains projector phones from exploiting their full capabilities. With them becoming ubiquitous in the next years such small-distributed user interfaces will become ubiquitous as well.

In this paper, we describe an expanding user interface design that tries to tackle common problems of projector phones e.g. attention shift that originates from the distribution of the user interface between the device and the projection employing a dynamic peephole metaphor. Peephole Displays are closely related to the category of spatially aware displays. Dynamic peephole interaction is a technique where a spatially aware display is moved and through this the viewport showing a part of the virtual information space is changed. In this way it is possible to make the whole information space visible which otherwise would not fit within the display. This technique creates a positional mapping between the virtual information space that the user wants to explore and the real world. Doing so they enable the use of the users spatial memory for navigation in the virtual information space. This technique has in several different use cases already been proven to be superior to static peepholes [4,7,14]. Furthermore, we present a prototype of such a distributed user interface

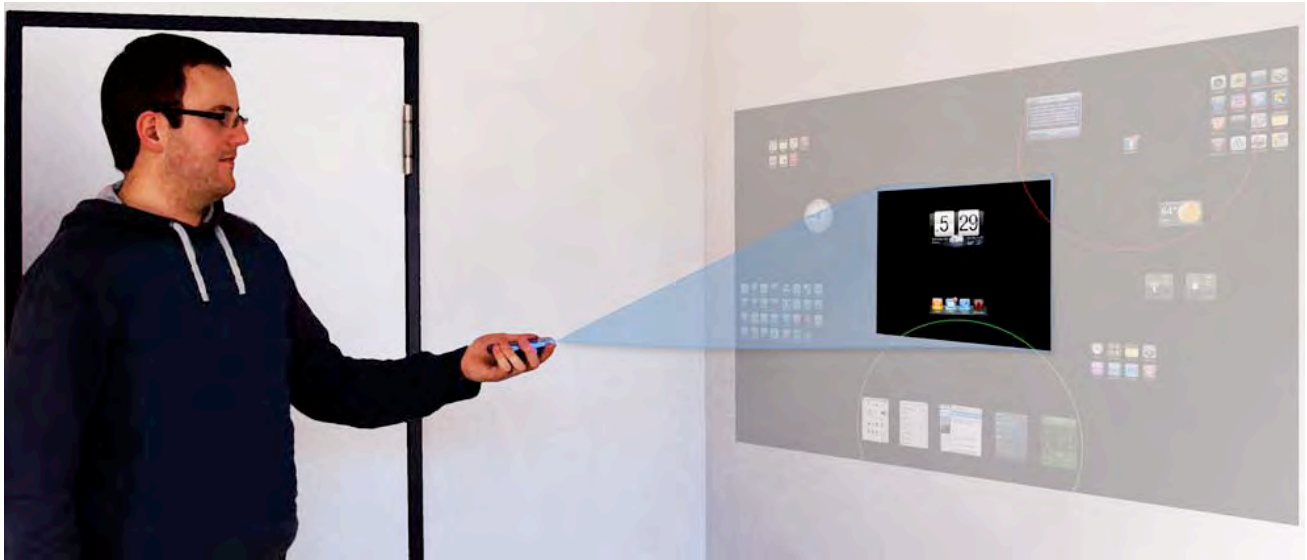


Figure 1: Design of a prototypical peephole projection interface where only a part of the whole information space is visible (in this figure opaque, invisible parts transparent). Messages that are off-projection are indicated using the Halo technique [1].

based on a map navigation application using the dynamic peephole metaphor.

RELATED WORK

To maintain a small form factor, mobile devices have a limited screen real estate. Mehra et al. therefore compared the advantages of static and dynamic peephole interfaces in terms of navigational speed on small mobile devices [7]. One example on today's mobile phones for static peephole navigation are map applications. Normally the window showing the map is static and the content is moved behind it either by using a joystick or touch-input. On the contrary with dynamic peephole navigation, the content stays static and window (the device) is moved. Mehra et al. found evidence that dynamic peephole interfaces are superior to static peephole interfaces which backs up the initial research on peephole interface by Fitzmaurice [4] and Yee [14]. Rohs et al. again found dynamic peephole navigation to be superior for augmented reality map navigation tasks [11]. Hürst et al. also found dynamic peephole navigation to be faster and favored by the users for the exploration of panoramic images [6]. A model for pointing tasks on peephole interfaces that is valid for a variety of display size was developed by Cao et al. [2].

Initial research on mobile projection interfaces was conducted by Raskar et al. with iLamps [9]. While iLamps mainly focused on creating distortion free projection on various surfaces, or using multiple projectors to create a larger projection, the follow-up of the iLamps, the RFIGLamps [10], were used to create object adaptive projections. How map navigation can benefit from a projected interface was shown by Hang et al. [5]. They initially investigated the problems that appear of the distribution of the interface on projector phones. Schöning et al. presented with MapTorchlight a technique to augment paper maps with additional information using a projector

phone [12]. Rapp et al. presented with spotlight a dynamic peephole interface called SpotLight for a handheld projector that relayed on movements [8]. The prototype presented in this paper is related to SpotLight but advances the peephole movements and adds additional controls on the device itself. With Halo [1] Baudisch et al. presented a technique to visualize off-screen objects on the screens of small mobile devices. Therefore a circle is drawn around the off-screen object in the virtual information space that reaches the edges of the displayed part of the information space. From the visible part of the circle the user can estimate direction and distance of the off-screen object.

APPLICATION CONCEPT

While modern mobile phones such as Apples iPhone rely on direct manipulation user interfaces [13] - possible through the touch screen - these techniques are not feasible on future projector phones for manipulating the projection due to the already mentioned reasons. To enable interaction with the projected display new user interfaces and interaction techniques are needed to handle the whole information space of the device. With the user interface being distributed between the device and the projection the user has to shift his attention from the content of the projection to the content on the devices display or the input constraints on the device. The principal goal of the design presented in this paper is to reduce the amount of attention switches to a minimum by setting the focus of interaction to the projected display. This is achieved by making moving most of the interface elements to the projection and making the interactions needed to the device as intuitive as possible. The devices display is only included for information display if necessary. Since the total information space is often even larger than the projection and to assure that everything is readable and visible from distance only an excerpt of the whole can be projected. To navigate in the projected virtual information space we propose to make use of the dynamic

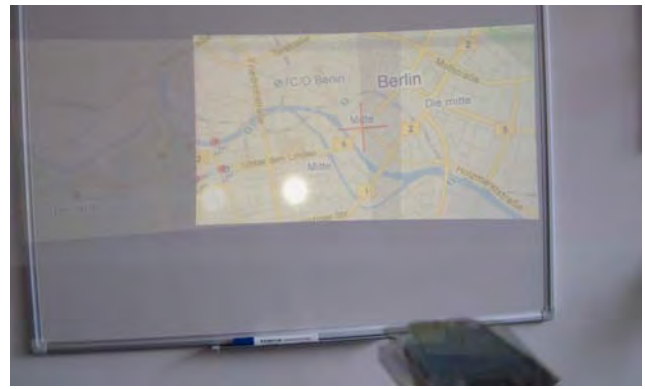
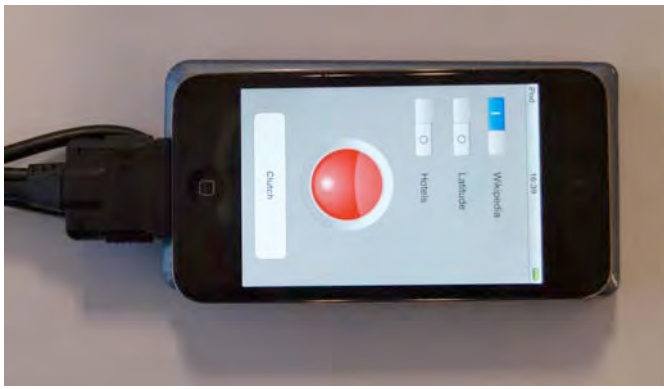


Figure 2: Prototype of the discussed interface exemplarily implemented for a map application. The hardware prototype showing the interface on the devices screen (left) and movement of the map – two images blended over each other (right)

peephole display metaphor, so that only a part of the whole information space is shown (compare **Figure 1**).

Since bigger movements of the device to change the viewport of the peephole interface are sometimes not possible we propose to use the devices orientation rather than movement. By changing the device orientation the projection on a wall would move and could cover the whole wall as well. The drawback would be a distorted image but therefore solutions already exist [3]. Most modern mobile phones are equipped with sensors like accelerometer and compass and Apples iPhone 4 additionally hosts a gyroscope. These sensors can be used to exactly determine the devices orientation and to measure even the slightest changes. By using the devices orientation rather than movement it is also possible for the user to retain a comfortable position and still explore the whole information space. Sometimes the projection surface maybe limited and this technique may not be feasible therefore an easy switching method to static peephole should be integrated as well. This static peephole could be shifted using the touchscreen of the device (or an isometric joystick). When using the touchscreen either a designated area should be at ones disposal or the whole touchscreen should be used. When using the whole screen as soon as the user starts to move the touching finger the input should not be processed by the underlying UI-element but be interpreted as a desired movement of the static peephole.

To change the center position of the information space when using a dynamic peephole we argue for adding a clutching method. By pressing the clutch button on the devices screen the movement detection is stopped and the user can move the center to the desired position and when he releases the button the center of the information spaced is shifted to this position. The selection of a specific object on the projected display is problematically since direct interaction is not possible. A possible solution would be indirect input with a mouse pointer controlled by the touchscreen. This solution seems to be cumbersome because it would require precise input on the touchscreen while focusing on the projection. Therefore we propose the cursor to be static in the center of the projection and by that it would move over the information space according to the peephole metaphor. This would make everything easily accessible. Objects that are not shown in the projection -

which are off-screen - but desire the attention of the user e.g. an incoming text message are indicated with a Halo [1]. For more detailed operations we propose to use the devices screen. One example is selecting an item from a combo box or a list, the user select the combo box on the projection and then the different selectable possibilities are shown on the devices screen where one can select a specific item. This would for example allow the interaction with private data that then would only be shown on the devices screen. Furthermore it could be a solution for private information that needs the users attention. A tactile feedback could indicate that the user now has to focus on the devices screen. The usage of a zoomable interface could facilitate the interaction with private data in such a way, that private information is only visible when the user is close enough to the projection surface to shield the projection for uninvolved passersby.

MAP INTERACTIONS FOR PROJECTOR PHONES

To show the validity of this design and how it could simplify everyday tasks like finding a hotel on a map we created a map application that adapts nearly all design ideas presented in the section above. Hang et al. already showed that map interaction can benefit from a projection [5] but they relayed on panning using buttons for their study. Our application allows easy interaction with the map using the presented design concept.

To explore the whole map the user simply changes the orientation of the device. By this the application than reveals the whole map part by part using the peephole metaphor (compare **Figure 2**). The map can be enhanced with different layers containing Points of Interest (POI) like georeferenced Wikipedia articles, Hotels or the Google Latitude position of your friends. The user can explore their content by moving the crosshair on the projection over the points and click on the select button. The information connected to the POI is then either shown on the projection or the devices screen when it is private data like the Google Latitude position of a friend. To select more than one point the user can simply press down the select button and drag the crosshair just like a mouse cursor. Layers can be added or removed to the map using either a drop-down menu or checkboxes. They are located on the devices screen to maintain easy and fast access (compare **Figure 2**). Off-

screen POIs are indicated by Halo's [1]. With that most of the discussed design concepts were realized inside one example application. Of course a normal phone interface with all the capabilities of modern mobile phones would be much more complicated to integrate.

IMPLEMENTATION

Since up to now no suitable projector phone is commercially available that is able to show different contents on the screen of the device and the projection we had to create our own prototype. On account of this we use a Microvision ShowWX laser projector connected to a fourth generation iPod Touch (compare **Figure 2**). The two devices together weigh 240g and with that are easy to handle. The projector provides only 10 lumens but since its laser technology it provides the advantage to be always in focus. The iPod touch was chosen since it is one of the few devices that allows to present different content on the display and the TV-Out and additionally contains a gyroscope. We used the gyroscope and the accelerometer in the device to determine its orientation highly precise and with that be able to control the viewport of the projected peephole. The projected image is not corrected before it is projected, so that depending on the angle between the device and the projection surface, it can get distorted. This problem could be overcome using the method developed by Dao et al. [3], which was created for a similar use case. The application for the iPod touch is based on the iOS UIMapKit, which already include many basic map functionalities.

CONCLUSION & FUTURE WORK

In this paper we discussed an initial design for a distributed peephole interface for projector phones. This design concept is a first try to facilitate the integration of projector phones in every day-life task by minimizing the drawbacks that evolve from the distributed interface of a projector phone. We presented our exemplarily implementation of this prototype. For future work we want to compare different selection techniques for objects on the projection as well as analyzing context switches to get a better understanding how to distribute the interface between the phone and the projection.

REFERENCES

1. P. Baudisch and R. Rosenholtz. 2003. Halo: a technique for visualizing off-screen objects. In *Proceedings of CHI '03*. ACM
2. X. Cao, J. J. Li, and R. Balakrishnan. 2008. Peephole pointing: modeling acquisition of dynamically revealed targets. In *Proceeding of CHI '08*. ACM
3. V. N. Dao, K. Hosoi, and M. Sugimoto. 2007. A semi-automatic realtime calibration technique for a handheld projector. In *Proceedings VRST '07*, Stephen N. Spencer (Ed.). ACM
4. G. W. Fitzmaurice. 1993. Situated information spaces and spatially aware palmtop computers. *Commun. ACM* 36, 7 (July 1993)
5. A. Hang, E. Rukzio, and A. Greaves. 2008. Projector phone: a study of using mobile phones with integrated projector for interaction with maps. In *Proceedings of MobileHCI '08*. ACM
6. W. Hürst and T. Bilyalov. 2010. Dynamic versus static peephole navigation of VR panoramas on handheld devices. In *Proceedings of MUM '10*. ACM
7. S. Mehra, P. Werkhoven, and M. Worring. 2006. Navigating on handheld displays: Dynamic versus static peephole navigation. *ACM Trans. Comput.-Hum. Interact.* 13, 4 (December 2006)
8. S. Rapp. 2010. Spotlight Navigation: a pioneering user interface for mobile projection. In *Ubiprojection 2010, Workshop on Personal Projection*.
9. R. Raskar, J. v. Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. 2005. iLamps: geometrically aware and self-configuring projectors. In *ACM SIGGRAPH 2005 Courses (SIGGRAPH '05)*, ACM
10. R. Raskar, P. Beardsley, J. v. Baar, Y. Wang, P. Dietz, J. Lee, D. Leigh, and T. Willwacher. 2005. RFIG lamps: interacting with a self-describing world via photosensing wireless tags and projectors. In *ACM SIGGRAPH 2005 Courses (SIGGRAPH '05)*, John Fujii (Ed.). ACM, New York, NY, USA
11. M. Rohs, J. Schöning, M. Raubal, G. Essl, and A. Krüger. 2007. Map navigation with mobile devices: virtual versus physical movement with and without visual context. In *Proceedings of ICMI '07*. ACM
12. J. Schöning, M. Rohs, S. Kratz, M. Löchtefeld, and A. Krüger. 2009. Map torchlight: a mobile augmented reality camera projector unit. In *Proceedings of the CHI '09*. ACM
13. B. Shneiderman. 1987. Direct manipulation: A step beyond programming languages. In *Human-computer interaction*, R. M. Baecker (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
14. K.-P. Yee. 2003. Peephole displays: pen interaction on spatially aware handheld computers. In *Proceedings of CHI '03*. ACM

Distributed User Interfaces for Smart Environments: Characteristics and Challenges

Marco Blumendorf, Dirk Roscher, Sahin Albayrak

DAI-Labor, TU-Berlin

Ernst-Reuter-Platz 7, 10587 Berlin, Germany

firstname.lastname@DAI-Labor.de

ABSTRACT

The availability of a multitude of networked interactive devices within smart environments leads to new possibilities for human-computer interaction. We can freely choose any device or combination of devices that suits our interaction needs. This however has to be supported by future applications and their user interfaces. A user interface that possesses the ability to be distributed amongst different devices is called Distributed User Interface (DUI). In this paper we analyze key issues for the usability of DUIs and the challenges for the development of DUIs at design time as well as the infrastructure at runtime based on a smart home scenario. We give an overview of related work in this area and briefly sketch our current model-based approach to address the challenges.

Author Keywords

Distributed User Interfaces, Smart environments

ACM Classification Keywords

H.5.2 [Information Interfaces and Presentation]: User Interfaces – *Interaction Styles*; H.1.2 [Models and Principles]: User/Machine Systems – *Human Factors*

General Terms

Human Factors

INTRODUCTION

Distributed user interfaces (DUIs) aim at providing the best possible interaction capabilities to users by utilizing multiple devices and considering the current context of use (user, platform and environment). Therefore DUIs have to support the usage of different interaction devices at the same time (simultaneous usage) and/or the usage of different interaction devices over time (sequential usage) by one or several users. A simultaneous usage of several devices e.g. allows using multiple modalities and utilizing the different features of each device. This enables users to better exploit the available interaction capabilities within one context of use. An additional sequential usage of

different (sets of) interaction devices allows further adapting the interaction to changing context of use. Thereby a seamless and continuous (best possible) interaction can be realized. Furthermore, a DUI can also be delivered to different users to support collaboration issues and support the combination of applications by taking multiple applications into account.

The next section illustrates how users can benefit from DUIs in their every-day life on basis of a smart home scenario. In the section thereafter we give a definition of a DUI and highlight some important characteristics of DUIs, as there is still no commonly accepted definition and consensus on the features and capabilities of DUIs (although they exist for over 15 years [4]). Supporting the illustrated usage of DUIs poses many challenges on the whole user interface (UI) lifecycle from the development of a DUI to its utilization at runtime. We elaborate on the challenges that can be derived from the scenario and the definition. Thereby we focus on the UIs. An overview of challenges for distributed systems (which a DUI also is) can be found in [13]. Finally, we briefly sketch our model-based approach to address these issues which we are currently pursuing in our work.

SZENARIO

The following scenario shows a possible usage of DUIs and we marked 8 key points which we use to illustrate DUI properties in the subsequent sections.

Every morning, when Dieter enters the kitchen, he is greeted by his smart home with the appointments for today. (1) On the more hectic weekdays the appointments are read out via voice, whereas at weekends the information is passively displayed on the kitchen screen. (2) The voice output follows Dieter while he moves on to the bathroom. Similarly on weekends, the displayed information moves to the living room with him until he confirms to be done with his daily planning. (3) This initial behavior of the system has been configured (and trained) by Dieter once, after he received his new calendar application. Additionally, Dieter sometimes needs to reschedule appointments to avoid conflicts. (4) To do so he orders the system to change from voice output to a presentation on the screen, synched with the display and controls of his smartphone, which allows him to interact and check details while keeping the overview on the big screen. Rescheduling appointments occasionally raises the need to contact colleagues and

customers to agree on a different date or timeslot. (5) For this purpose Dieter's calendar application allows to set up video calls to contacts while sharing the relevant calendar information with the called person if needed.

Leaving the house, the calendar application remains accessible via Dieter's PDA, which also comes in handy if he visits his customers in their homes or offices. (6) If available, he uses their smart home infrastructure to find timeslots for follow-up appointments or to display details and notes that have been agreed on in previous appointments. (7) When negotiating on follow-up appointments Dieter can select which parts of his calendar should be shared and (8) due to the ability to store such a configuration is able to reactivate the configuration whenever he revisits the customer.

A DEFINITION AND CHARACTERIZATION OF DUIS

The given scenario illustrates some key points of DUIS. It shows (1) automatic choice of interaction devices depending on context information, (2) the dynamic change of devices based on context information, (3) the configuration of this behaviour by users, (4) user-control to change devices on demand, (5) the sharing of information with other users, (6) the simultaneous usage of different devices to provide multimodal interaction, (7) the user-controlled sharing of parts of the application, and (8) storing and loading of configurations.

Next, we are presenting a definition and selected properties of DUIS which are based on our vision of and experiences with DUIS (illustrated in the scenario) as well as related work in this area. The reference model proposed in [11] classifies mouldable, distributable, and migratable UIs and is extended in the 4C reference framework [12] to four general dimensions for DUIS: computation, communication, coordination, and configuration. Berti et al. [3] present a taxonomy for migratory UIs comprising 11 dimensions of the migration process. However, we are not aware of a common and widely accepted definition of DUIS.

A basis for the understanding of DUIS and the support of multimodal interaction is the consideration that interaction devices are composed of several interaction resources (IRs), whereas each IR is a "one-way interaction channel (in- or output) that is used to communicate with an interactive system" [18]. For example, microphones and keyboards are input IRs and displays and loudspeakers are output IRs. Based on this notion we define DUIS as UIs, that support any combination (>1) of different input and output IRs to simultaneously interact with a single application. Distribution thus describes the process of splitting UIs across multiple IRs [6].

A basic observation is that DUIS in this definition are *multi- & cross-device*, *multi- & cross-modal*, *multi- & cross-user* and *multi- & cross-application*. "Multi" here refers to the fact that the UI addresses more than one element per dimension (i.e. device, modality, user, application) at once,

while "cross" refers to the fact that an element can change dynamically at runtime (e.g. moving from speech to graphical output or transferring controls to a different user).

Addressing the fact of these changes, we distinguish *static distribution*, i.e. a predefined configuration of DUIS at runtime, and *dynamic distribution*, i.e. system or user triggered configuration changes at runtime. A distribution configuration in this sense defines which parts of the UI are presented via which IRs and modalities. A static configuration is preconfigured by the developer, while a dynamic configuration gives control to user or system at runtime and might be based on context information. Depending on the usage of the devices, we can further distinguish *migration* (the UI moves from one IR to another) and *replication* (the UI is replicated on another IR). This also leads to the notion of *complete* (the whole UI) or *partial* (only part of the UI) migration/replication. Finally, an important property of the DUIS we examine here is *context sensitivity*, which allows dynamic distribution based on the usage situation, but also leads to various challenges for user and developer.

CHALLENGES

Based on the definition and the identified DUIS properties, we can now analyze the related challenges the development and utilization of DUIS pose to users and systems at runtime as well as to developers at design time.

User Perspective

Considering the dimensions of DUIS (i.e. varying devices, multiple modalities, different users, many application), it might be difficult for users to keep their mental model of the interactive system up-to-date. The user is challenged by interacting with a dynamic and widely distributed system of (maybe) not obviously related IRs that can also change for not immediately obvious reasons. This requires special consideration for the usability and intelligibility of DUIS.

One major aspect of this consideration is the controllability of DUIS (see points (3) to (5), (7) and (8) of the scenario). Controllability denotes the capability of users to always have full control about the distribution of a UI. Thus users should always have the possibility to adapt the distribution to their needs and store and reactivate static and dynamic distribution configurations. One approach for providing such functionalities would be a meta-UI [10].

Controllability here also includes the understanding of the current and changing distributions and especially their cause (as a new distribution could also be automatically initiated by the applications or an underlying runtime system), which means making the distribution status and process transparent for the user. The currently supported modalities and IRs used for a DUIS have to be comprehensible and as obvious as possible for the user at all times. Whenever a new (partial) distribution is performed it should be traceable for the user which parts of

the UI are presented on which IR and how to use the DUI. The user must be able to follow and understand the distribution alteration by detailed feedback.

Thereby another important aspect is the continuity of the interaction. The performed distributions are required to provide a continuous and consistent user experience without confusion and should conserve the state of the interaction. A distribution across multiple devices should be consistent during the whole usage of the application (e.g. using two screens for one task, the next task should use the same two screens). When changing applications, distribution changes have to be reduced to a minimum.

Runtime Infrastructure Perspective

Due to the application independent features and the goal to integrate multiple applications into the distribution, the need for a (shared) runtime system that handles interaction resources, context and application behavior arises. This runtime infrastructure has to provide functionalities for tackling the challenges from the user perspective and need to support changing multimodal interaction (point (6) of the scenario) and the automatic selection or calculation of distributions based on context information (points (1) & (2)).

This requires the handling of all dimensions of DUIs (devices, modalities, users and applications) centrally or in distributed, self-organizing manner. From a runtime infrastructure perspective, modalities, users, and applications are abstract concepts that need appropriate consideration in terms of UI variant selection or adaptation, whereas devices and their IRs need to be manageable and addressable by the runtime infrastructure. Access to these entities and knowledge about them is crucial for the handling of DUIs.

During any interactive dialog with the user (based on the distributed UI) the runtime infrastructure needs to process (multimodal and possibly distributed) user input and generate consistent and meaningful responses as output. Especially dynamically changing distributions pose a big challenge to the interaction management. The DUI has not only to be managed, recalculated and kept at a consistent state, but the migration of the different UI parts has to be performed, the UI has to be split and user input has to be fused to process multimodal information. This whole process is based on context information and requires identifying those parts of the available context information that is relevant for the current reconfiguration.

Developer Perspective

The main challenge at design time is the need to define the application and its UI for various (combinations of) devices, modalities, users and in line with other applications. This becomes even more challenging when context sensitivity and dynamic distribution at runtime come into play. The support for different variants of UIs is

a major goal of model-based development, which we briefly sketch as a possible solution later. However, the manifold configuration variants which are required to address context sensitive device-, modality-, user- and application- spanning dynamic DUIs are nearly impossible to predefine at design time. So another main challenge is the specification of the capabilities and borders of a DUI to allow the dynamic handling, selection and adaptation of different UI variations at runtime. A key factor is the user who is required to be able to influence this process and override any kind of developer configuration. At design time this raises the well-known challenge of uncertain and unknown runtime contexts, which requires to work with assumptions and more or less detailed simulations.

All these problems require massive support for the developer and thus the provisioning of qualified methodologies and tools.

RELATED WORK

As mentioned before, there is no common definition of DUIs and thus different approaches pursue different goals and address different issues. However, several approaches are available in the literature, each with their own advantages and drawbacks.

The DynAMITE (Dynamic Adaptive Multimodal IT-Ensembles) project [17] aims at the development of software-infrastructures that use self-organizing ad-hoc appliance ensembles. It facilitates the utilization of different devices and modalities that can be dynamically assembled to support user interaction at runtime. In the same context, Elting and Hellenschmidt [14] present conflict resolution strategies when distributing output across graphical UIs, speech syntheses and virtual characters. Braun and Mühlhäuser [8] describe an approach for pattern based distribution of UIs at runtime. The I-AM (Interaction Abstract Machine) system [1] presents a software infrastructure for distributed migratable UIs. It provides a middleware for the dynamic coupling of IRs (screens, keyboards and mice) to form a unified interactive space. Berti et al. [2] describe a migration process based on the interactive system model, defining the application and the context model, defining the outside world, as the two main entities. Selected other approaches are Grolaux et al. [15] presenting detachable interfaces, Bandelloni and Paternò [19] presenting migratable UIs and sketching a runtime system for their realization and WebSplitter [16] supporting the web-based distribution to multiple devices.

ENGINEERING DUIs

In the sections above we introduced and defined DUIs and analyzed the challenges for their development and implementation. In this section we briefly sketch a model-based system (detailed information are available in [6]), providing dynamic DUIs.

Our approach is based on the utilization of models@run.time [5]. The UI developer is thus able to work with a set of models defining the relevant aspects of the UI from different perspectives. The same models are utilized at runtime to preserve the design decisions and information and derive the actual UI based on the information. Following the Cameleon Reference Framework [9] we utilize a task and domain model, refined by an abstract and concrete UI model. Separating abstract interaction as well as concrete input and concrete output, the models provide the required separation of input and output for DUIs, while still keeping the semantic connection via the abstract interaction. The developed approach allows the specification of task relations as well as modality, temporal and spatial relations as constraints for the distribution. Furthermore, we utilize a distribution model to store distribution configurations created by user, developer or the runtime infrastructure. The configurations are incorporated into the distribution process in this order [7]. The distribution process in turn, similar to the approach described in [10], is based on a repeating procedure: discover & manage the available IRs, select a set of IRs to distribute the UI to, split the UI into the different segments and assign the segments to IRs, deliver the different segments to the selected IRs, receive input from input IRs and update output IRs according to the UI state [6]. Once the distribution has been calculated for all currently active elements of the UI model, the elements for each IR are passed to the layout component for arrangement and styling. Afterwards, a communication channel takes care of their suitable presentation on the connected IR.

So far the described approach allows the creation of dynamically distributed multimodal UIs and a meta-UI provides control elements for the user. However, multi-user and multi-application scenarios are not yet supported and the controls of the meta-UI are not fully developed for some parts of the illustrated scenarios.

CONCLUSIONS

In this paper we provide a definition and selected properties of DUIs. Based on the observation that dynamic DUIs are *multi- & cross-device*, *multi- & cross-modal*, *multi- & cross-user* and *multi- & cross-application* we described challenges from the user, runtime infrastructure and developer perspective. These challenges have to be solved to provide DUIs as illustrated in the scenario. At last we briefly sketched our model-based approach which is the basis for tackling the presented challenges in the future.

REFERENCES

1. Barralon, N., Coutaz, J., and Lachenal, C. Coupling interaction resources and technical support. In HCI International 2007, pp. 13-22.
2. Berti, S., and Paternò, F. Migratory multimodal interfaces in multidevice environments. In Proc. of ICMI'05..
3. Berti, S., Paternò, F., and Santoro, C. A taxonomy for migratory user interfaces. In Proc. of DSV-IS'05.
4. Bharat, K., and Brown, M. H. Building distributed, multi-user applications by direct manipulation. In Proc. of UIST 1994, ACM, pp. 71-80.
5. Blair, G., Bencomo, N., and France, R. B. Models@run.time. Computer 42, 22-27.
6. Blumendorf, M. Multimodal Interaction in Smart Environments - A Model-based Runtime System for Ubiquitous User Interfaces. PhD thesis, Technische Universität Berlin, 2009.
7. Blumendorf, M., Roscher, D., and Albayrak, S. Dynamic user interface distribution for flexible multimodal interaction. In Proc. of ICMI 2010.
8. Braun, E., and Mühlhauser, M. Automatically generating user interfaces for device federations. In ISM '05, IEEE Computer Society, pp. 261-268.
9. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. A unifying reference framework for multitarget user interfaces. Interacting with Computers 15, 3 (2003), 289-308.
10. Coutaz, J. Meta-user interfaces for ambient spaces. In TAMODIA 2006, Springer, pp. 1-15.
11. Demeure, A., Calvary, G., Sottet, J.-S., and Vanderdonckt, J. A reference model for distributed user interfaces. In Proc. of TAMODIA '05, ACM, pp. 79-86.
12. Demeure, A., Sottet, J.-S., Calvary, G., Coutaz, J., Ganneau, V., and Vanderdonckt, J. The 4c reference model for distributed user interfaces. In ICAS 2008.
13. Dollimore, J., Kindberg, T., and Coulouris, G. Distributed Systems: Concepts and Design, 4 ed. Addison-Wesley, 2005.
14. Elting, C., and Hellenschmidt, M. Strategies for self-organization and multimodal output coordination in distributed device environments. In Workshop on Artificial Intelligence in Mobile Systems 2004.
15. Grolaux, D., Vanderdonckt, J., and Roy, P. V. Attach me, detach me, assemble me like you work. In INTERACT 2005.
16. Kirste, T. Dynamite - dynamisch adaptive multimodale it ensembles. In Tagungsband: Forschungsoffensive "Software Engineering 2006" (2004).
17. Han, R., Perret, V., and Naghshineh, M. Websplitter: a unified XML framework for multi-device collaborative web browsing. In CSCW'00, pp. 221-230.
18. Luyten, K., and Coninx, K. Distributed user interface elements to support smart interaction spaces. In Proc. of ISM '05, IEEE Computer Society, pp. 277-286.
19. Bandelloni, R., Paterno, F. Migratory user interfaces able to adapt to various interaction platforms. IHCS 60 (2004), 621-639

Distribution Primitives for Distributed User Interfaces

J          ¹, Jean Vanderdonckt¹, and Peter Van Roy²

Universit   catholique de Louvain

¹Louvain School of Management, Place des Doyens, 1 – B-1348 Louvain-la-Neuve, Belgium

²Computer Science Department, Place Sainte-Barbe, 2 – B-1348 Louvain-la-Neuve, Belgium

+32 10 47 {8379, 8525, 8374} – {jeremie.melchior, jean.vanderdonckt, peter.vanroy}@uclouvain.be

ABSTRACT

This paper defines a catalog of distribution primitives for Distributed User Interfaces. These primitives are classified into four categories and represents operations that the developers and/or the users can executes to distribute the UI.

Author Keywords

Distributed User Interfaces, Distribution Primitives.

ACM Classification Keywords

C2.4. Distributed Systems. H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms

Design.

INTRODUCTION

The domain of Distributed User Interfaces (*DUI*) is still in evolution and there exist no toolkit allowing the creation of DUIs. In most pieces of work, there is almost *no genuine DUI*. There exists toolkits to create UI such as Java Swing or Microsoft Foundation Classes, but they do not support DUIs [4]. The UI elements simply remain in their initial context, while communicating with each other, but without redistribution. There is some distribution of UI elements, but it is mainly *predefined* and *opportunistic*: no configuration of the distribution at run-time. In Sj  lund [1], the repartition of UI elements across the Smartphone and the TV is fixed. It is not possible to rearrange their distribution. Some works allow distribution at run-time but with some limitations. The UI elements subject to this redistribution are mainly *containers*, such as windows or dialog boxes. The problem is that the *granularity* of UI distributed elements is often *coarse-grained*; it is not possible to distribute at the widget level.

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

transferred to other platforms.

In Luyten[3], there are already attempts to model the distribution. The granularity is however limited to tasks that are predefined before the application starts.

To sum up, we are looking for a way to support distribution at both design-time and run-time with very fine and coarse-grained granularity and to support replicable distribution while being compliant with the DUI goals as in [5]. This paper tends to help understanding and managing DUI.

CATALOG OF DISTRIBUTION OPERATIONS

We propose a catalog of distribution operations and a toolkit based on this catalog.

Toolkit

A toolkit has been developed upon the catalog. It creates application with the UI separated in two-parts: the proxy and the rendering. A command line interface is provided to allow manual redistribution at run-time, see Fig. 1.



Figure 1. Command line interface provided by the toolkit

In Fig. 2, the proxy is represented as a separate part of the application than the rendering. The first keeps the state of the application and ensures the core functionalities, while the second displays the user interface. Applications that support DUI allow the rendering part to be distributed on other platforms while the proxy stays on the platform where the application has been created. The toolkit works in Mozart environment supported by Microsoft Windows operating systems (XP and newer), Apple Mac OS X, Linux and Android. We are currently working on the full support for Apple iOS. As Mozart is a multi-platform

environment, the applications created with this toolkit are also multi-platform. Each graphical component is described as a record containing several keys and values. It ensures compatibility with the XML format because the keys/values become the name/value pairs for the XML markup. The DUI can be controlled by a command line interface, a meta-UI or even by the applications themselves using distribution scenario. A model-based approach closely related to the toolkit has already been described in a paper that has been accepted to EICS 2011 [2]. The definitions and examples presented in this paper come from this paper.

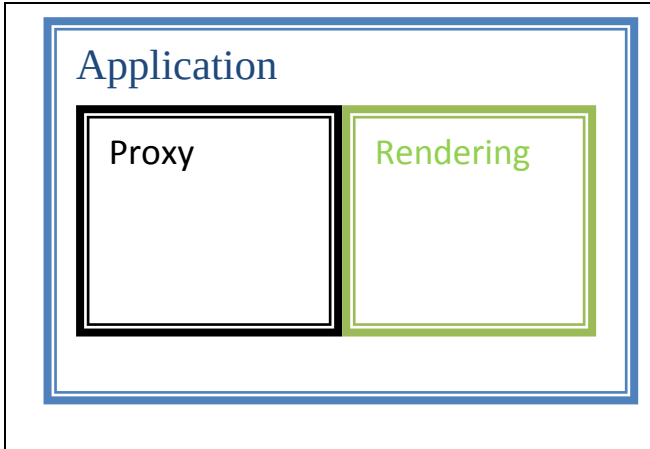


Figure 2. Structure of a DUI application.

Definitions

As observed in the related work section, the distribution logic of DUIs is often hard-coded and is not represented explicitly, which prevents us from reasoning on how distribution is operated. In order to address this issue, we now provide a catalogue of *distribution primitives* that will operate on CUI models of the cluster. We first define these distribution primitives in natural language, then in an Extended Backus-Naur Form (EBNF) format. In this notation, brackets indicate an optional section, while parentheses denote a simple choice in a set of possible values. In the following definitions, we use only one widget at a time for facilitating understanding. In the EBNF, we will use the four selector mechanism standard from W3C for CSS for generalization to all widgets.

SET <Widget.property> TO {value, percentage} [ON <Platform>]: assigns a value to a CUI widget property or a percentage of the actual value on a platform identified in a cluster. For instance, SET “pushButton_1.height” TO 10 will size the push button to a height of 10 units while SET “pushButton_1.height” TO +10 increases its height by 10%. Note that the platform reference is optional: when it is not provided, we assume that the default platform is used.

DISPLAY <Widget> [AT x,y] [ON <Platform>]: displays a CUI widget at a x,y location on a platform identified in a cluster, where x and y are integer positions (e.g., in characters or pixels). For instance, DISPLAY

“pushButton_1” AT 1,1 ON “Laptop” will display an identified push button at coordinates 1,1 on the laptop. UNDISPLAY <Element> [AT x,y] [ON <Platform>] is the inverse operation. DISPLAY <Message> [AT x,y] [ON <Platform>] displays a given message on a designated platform in the cluster (mainly for user feedback in an optional console).

COPY <Widget> [ON <SourcePlatform>] TO [<Widget>] [ON <TargetPlatform>]: copies a CUI widget from a source platform identified in a cluster to a clone on a target platform, thus creating a new identifier. This identifier can be provided as a parameter to the primitive or created automatically by the primitive to handle it.

MOVE <Widget> TO x,y [ON <TargetPlatform>] [IN n steps]: moves a CUI widget to a new location indicated by its coordinates x and y, possibly in a fixed amount of steps, on a target platform in the cluster.

REPLACE <Widget1> BY <Widget2>: replaces a CUI widget Widget1 by another one Widget2. Sometimes the replacement widget could be determined after a (re-)distribution algorithm, thus giving the following definition: REPLACE <Widget1> BY <Algo>:. This mechanism could be applied to contents and image transformations: images are usually transformed by local or remote algorithms (e.g., for resizing, converting, cropping, clipping, repurposing), thus giving the following definition: TRANSFORM <Image1> BY <ImageAlgo:URL>.

MERGE <Widgets> [ON <SourcePlatforms>] TO [<Widget>] [ON <TargetPlatform>]: merges a collection of CUI widgets from a source platform identified in a cluster into a container widget on a target platform, thus creating a new identifier. Again, when source and target platforms are not provided, we assume that the default platform is used. SEPARATE is the inverse primitive. SEPARATES <Widgets> [ON <SourcePlatforms>] TO [<Widgets>] [ON <TargetPlatforms>]: splits a collection of CUI widgets (typically, a container) from a source platform identified in a cluster into CUI widgets on one or many target platforms.

SWITCH <Widget> [ON <SourcePlatforms>] TO [<Widget>] [ON <TargetPlatform>]: switches two CUI widgets between two platforms. When the source and target platforms are equal, the two widgets are simply substituted.

DISTRIBUTE <Elements> INTO <Containers> [BY <DistribAlgo:URL>]: computes a distribution of a series of UI Elements into a series of UI Containers, possibly by calling an external algorithm, local or remote.

EBNF Grammar. In order to formally define the language expressing distribution primitives, an Extended Backus Naur Form (EBNF) grammar has been defined. EBNF only differs from BNF in the usage of the following symbols: “?” means that the symbol (or group of symbols in parenthesis) to the left of the operator is optional, “*”

means that something can be repeated any number of times, and “+” means that something can appear one or more times. EBNF has been selected because it is widely used to formally define programming languages and markup languages (e.g., XML and SGML), the syntax of the language is precisely defined, thus leaving no ambiguity on its interpretation, and it is easier to develop a parser for such a language, because the parser can be generated automatically with a compiler (e.g., YACC). Instances of distribution primitives are called by *statements*. The definitions of an operation, a source, a target, a selector and some other ones are defined in Fig. 3 and 4 (excerpt only). The definitions could be extended later to support more operations or features.

```
statement = operation , white_space , source , white_space , "TO" ,
white_space , target ;
operation = "SET" | "DISPLAY" | "UNDISPLAY" | "COPY" |
"MOVE" | "REPLACE" | "TRANSFORM" | "MERGE" | "SWITCH" |
"SEPARATE" | "DISTRIBUTE";
source = selector ;
target = displays | selector , white_space , "ON" , white_space ,
displays ;
displays = display_platform , { "," , display_platform }
display_platform = display , [ white_space , "OF" , white_space ,
platform ] ;
selector = identifier , { "," , identifier } | universal ;
display = identifier ;
platform = identifier ;
```

Figure 3. EBNF grammar for distribution primitives (excerpt).

Examples

In order to illustrate how distribution primitive could behave, we hereby provide a series of increasingly complex examples. In Fig. 4, a display of the platform by default has been modified in the following way: DISPLAY “pushButton_1” AT 5,5 ON “defaultPlatform” followed by SET “pushButton_1.label” TO “B”, thus creating a CUI model attached to this platform.

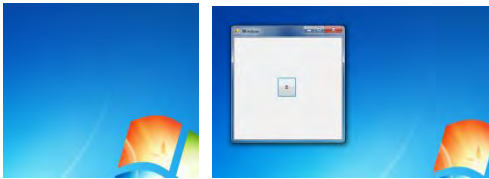


Figure 4. Example of a simple display primitive.

Distribution operations can be more complex than the example provided in Fig. 4. Here is a series of examples for the COPY primitive:

1. COPY button_1 TO shared_display: simple copy of button_1 sent to shared_display without specifying neither an identifier nor a container
2. COPY button_1 TO button_2 ON shared_display: copy button_1 on shared_display and identify it as button_2
3. COPY button_1 TO button_2 ON shared_display of

shared_platform: the same but we specify the shared_platform to avoid searching through all the platforms

4. COPY button_1, button_2 TO shared_display: copy button_1 and button_2 to shared_display in a single operation
5. COPY button_1 TO shared_display, my_display: copy button_1 to shared_display and also to my_display
6. COPY button_1 TO shared_display OF shared_platform AND my_display OF my_ipad: copy button_1 to both shared_display and my_display, specifying on which platform is each display
7. COPY * TO shared_display: copy all the graphical components from the current UI to shared_display
8. COPY ALL buttons TO shared_display: copy all buttons to shared_display
9. COPY individuals TO shared_display: copy any individual CUI widgets to shared_display

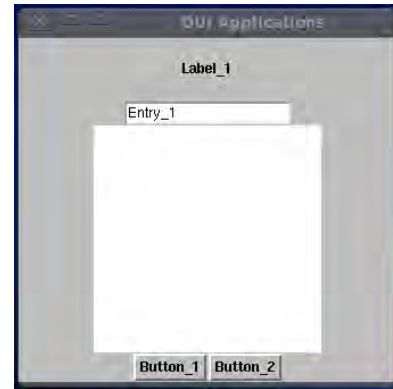
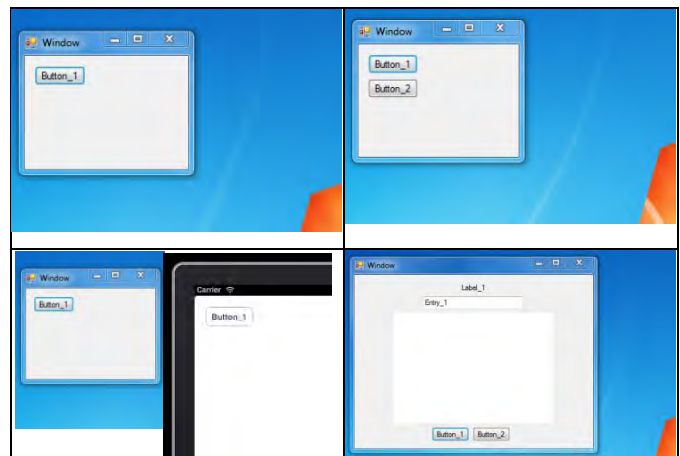


Figure 5. Source CUI for the COPY examples.



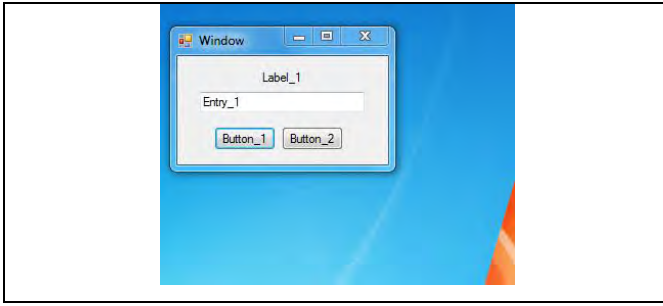


Figure 6. Results of examples 1, 2 & 3 (top left), 4, 8 (top right), 5 & 6 (middle left), 7 (middle right), 9 (bottom).

The source CUI associated to these examples is reproduced in Fig. 5, while the results of the nine above copy operations are reproduced respectively in the different regions of Fig. 6.

Meta-User Interface for Distribution Primitives. The distribution primitives defined in the previous subsection can be called in two ways:

1. *Interactively* through a meta-UI [9] providing a command line equipped with a command language: in this way, one can type any distribution primitive through statements that are immediately interpreted and provide immediate feedback. This meta-UI adheres to usability guidelines for command languages (such as consistency, congruence, and symmetry), but does not provide for the moment any graphical counterpart of each statement or graphical representation of the platforms of the cluster. Actually, each platform is straightforwardly addressed at run-time. It is of course possible to see the results of a distribution primitive immediately by typing it by an “errors and trials” process until the right statement is reached. Fig. 7 reproduces a screen shot of this meta-UI, which also serve as a tutorial to understand how to use the distribution primitives. Indeed, any statement type in the command language can be stored in a list of statements that could be recalled at any time.



Figure 7. Source CUI for the COPY examples.

2. *Programmatically*: each statement representing an

instance of a distribution primitive can be incorporated in an interactive application in the same way since the parser will be called to interpret it. It is therefore no longer needed to program these primitives.

FUTURE WORK

These distribution primitives have been defined in such a way that it allows distributing any graphical widget without focusing on how the value is shared when distributed. There are two important aspects in a User Interface: the *presentation* and the *behavior* of the UI. The presentation part is the graphical representation of a widget. If the widget is moved to another platform, it means that the same widget will be displayed on the destination platform. This leads to an ambiguous solution about the behavior. Is the widget still acting like it was on the other platform? Is the action taking place locally (to the new platform) or globally (in the same way than before being moved, like a remote)? The behavior is an important future work to keep in mind.

CONCLUSION

In this paper, we have introduced the concept of distribution primitives and a toolkit based on them. The goal is to provide a catalog of distribution primitives as a common base for researchers on DUIs. They now have the same set of primitives. It allows them to share the same possibilities independently of the UI implementation.

A toolkit based on this catalog has also been introduced and allows developers to see the powerfulness of this catalog.

ACKNOWLEDGMENTS

The author would like to acknowledge the support of the ITEA2-Call3-2008026 USXML (User Interface extensible Markup Language) European project and its support by Région Wallonne.

REFERENCES

1. Sjölund, M., Larsson, A., Berglund, E. Smartphone Views: Building Multi-Device Distributed User Interfaces. In: *Proc. of MobileHCI'2004*. LNCS, Springer (2004), 507-511.
2. Melchior, J., Vanderdonckt, J., Van Roy, P. A Model-Based Approach for Distributed User Interfaces. Full paper accepted for *EICS'2011*.
3. 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.
4. Tan, D.S. and Czerwinski, M. Effects of Visual Separation and Physical Discontinuities when Distributing Information across Multiple Displays. In: *Proc. of INTERACT'03*. IOS Press (2003), 252-260.
5. Vanderdonckt, J. Distributed User Interfaces: How to Distribute User Interface Elements across Users, Platforms, and Environments. In: *Proc. of Interacción'2010*. AIPO (2010), 3-14.

Humane Machine Interaction Design: DUIs Design Thinking for Social Innovation

Niki Lambropoulos

Intelligence

1 Dale Grove, N128EE, London, UK

niki@intelligence.eu

Sophi Danis

Intelligence

1 Dale Grove, N128EE, London, UK

sophi@intelligence.eu

ABSTRACT

In the age of ubiquitous computing and facing a global financial crisis, three main issues appear to concern the current Human Computer Interaction (HCI) research: the artificiality of information input and output, the lack of human values and the variety of technological devices. In this paper we revisit these HCI issues from both a social/human values and design perspective. Such Humane Machine Interaction Design is implemented in Social Innovation via Design Thinking. These frameworks target specific communities' needs and visions for Positive User eXperience based on members' real stories, inspiration, improvisation and experience within their individual contexts. Lastly, we provide an abstract model of such implementation in Social Innovation Management using Distributed User Interfaces so to tackle the diversity of monitors, devices, computing platforms, users, physical environments, contexts of use, disciplines and tasks.

Author Keywords

HCI, Design Thinking, Social Innovation, DUI.

ACM Classification Keywords

H5.m. Information interfaces and presentation (HCI): DUIs.

General Terms

Design, Human Factors, Theory.

INTRODUCTION

Since the dawn of civilization, humans have been developing tools in order to enhance their knowledge and abilities. Social imitation facilitated the utilisation of such knowledge and tools thus creating our cultures [1]. Rather than using tools to extend *physical* abilities, the initial target of computing pioneers was to augment the human *intellect* [2]. In the 21st century, we acknowledge that the power and impact of social interaction and the ensuing social imitation, fashion cultural trends. However, despite the

current technological developments, there is evidence that human intellect is actually aging and creativity is declining [3,4]. Social Innovation can provide a solution for this critical situation to a more sustainable and socially responsible future by directly addressing social needs. Consequently, the involvement of all users in Social Innovation is critical and thus it raises issues for multiple contexts and interfaces. Therefore, this paper aims to shed light on such research by exploring the Humane aspect in Machine Interaction, Design Thinking and Distributed User Interfaces to support Social Innovation.

HUMAN COMPUTER INTERACTION (HCI)

In "The Shallows", Carr objects to the idealization of the machines and proposes that we need to be aware of what we lose in order to gain; we can see far away with a telescope but we miss the world around us. In other words, Carr suggests that machines create particular ways of receiving and elaborating information in the human brain; as a result, our knowledge becomes artificial. Tackling such an artificial perspective of the world is a challenge for the future of design and education in particular.

Also, Greenfield [3] is concerned about the ways our children learn with the current technologies mainly because they only alter the children's iconic representation and visual-spatial skills. Conversely, there is strong evidence of severe weaknesses in higher-order cognitive and emotional processes, such as mindfulness, reflection, inductive problem solving, abstract vocabulary, critical thinking, social sensitivity, empathy, emotion, engagement, tenderness, creativity and imagination.

Such human aspects consist an essential part in HCI. HCI is the discipline that involves the design, implementation and evaluation of interactive computer systems in the context of the users' task and work [5] as well as the study of major phenomena surrounding them [6]. Other than effective interaction, the most recent HCI definition [7] was further refined with the addition of the human aspect. The authors suggested that HCI should include the understanding of human values in order to guide and direct systems design. For example, the social and the individual are not different levels but inexorably interconnected 'in situ' [8]. Therefore, humane aspects in design may need to be considered.

Humane Machine Interaction Design (HMID)

It appears that human machine interaction turned us more artificial rather than making the machines more human. As with the need for ethics in creativity, bringing the cognitive aspect, emotion and morals back to consideration in research is also stressed in the Opinion Special section of October's 2010 New Scientist. A Humane Machine Interaction Design (HMID) can be a framework to encompass effective user - machine interaction by specifically intentionally directing design and development towards human values. It aims to augment human cognition, intellect and emotion by employing the machine within the users' context, creating a Positive User eXperience (UX+).

UX+ is based on community members' way of working, stories, improvisation and experience within their own contexts to support agile software development. Such iterative approach supports the proposed principles via accelerating formative and summative substantive testing and feedback, iterating the design fundamentals in a quick-and-dirty (guerilla) way. Overall, HMID advocates five measureable *milestones*:

1. *Individual Capability* may measure the quality of human self-management, in terms of advancing him/her self alone or within a group or the society in general.
2. *Sociability* may measure the quality of human skill and experience when interacting with other humans in terms of effectiveness, efficiency, and satisfaction within a group or the society in general.
3. *Usability* may measure the quality of human experience when interacting with a machine, in terms of utility, effectiveness, efficiency, acceptability and satisfaction.
4. *Direction* may measure the quality of the previous three principles towards a specific target defined by a human, a group, the society or a director (e.g. a project manager or a teacher). Direction is in the form of orchestration and management of *individual capability*, *sociability*, and *usability* towards a specific target; different targets of the system lead to different behaviors.
5. *Sustainability* may measure the quality of effectiveness and impact of the previous principles on a time continuum (e.g. the lifespan of a project or a learning course).

The first and second principles involve the interaction between the social and the individual; the third technical

principle directly refers to ISO standardisations [9, 10]; and the last two, aim toward a more inclusive persistence of quality.

DESIGN THINKING FOR SOCIAL INNOVATION

Any innovation has impact on particular social groupings and – more or less – individuals in their respective capacities, e.g. as consumers, entrepreneurs, administrators, family members. Social innovations are new concepts and measures to resolve societal challenges, adopted and utilised by social groups concerned [11]. According to Hochgerner [12], there is a distinction between idea/invention and dissemination, a social idea becomes a social innovation in the process of implementation and dissemination. Socio-cultural foundations for innovations are society's value systems, and in particular aspects as problems or needs (e.g. equity/disparities, inclusion/exclusion). These consist of the local objectives as observed patterns that can become both social and economic objectives. On the surface we may observe innovation types, such as more/less, faster/slower, growth/decline etc. Social innovations (like any innovation) meet resistance and must compete with other traditional or newly proposed solutions to social issues.

In order to turn such knowledge into action, innovation management systems aid and enhance the derivation of concepts, competencies and capabilities as innovation resources and provide the needed data, abstraction levels, processes, figures and facts. Other than ideas, needs, solutions and visions, great importance is attached to the specific community culture that produces benefits for as many community member-groups as possible. Also, they need to support interrelation of information and real users' feedback, pattern recognition, as well as attributions, conflicts and forecasts or scenarios. In other words, they aim at stimulating, facilitating and advancing the utilisation of cognition, intellect and emotion to allow specific innovation strategies to arise.

Design Thinking (DT) is a design model derived from IDEO, a creativity-in-business company. It relies on our inherent human ability to be imaginative and intuitive, recognize patterns, construct ideas, and to express ourselves in several ways. DT *four overlapping spaces* are *inspiration*, *ideation*, *implementation* and *impact*. The first provides the initial motivation and intention, the second ideas generation, development and testing, and the third the transition and realisation from the project stage into people's lives.

Also DT incorporates constituent or consumer insights in depth and rapid prototyping as in agile software development in order to avoid assumptions and conventional problem-solving practices that block effective and high impact solutions [13]. All three DT spaces can be supported by Distributed User Interfaces (DUIs).

DISTRIBUTED USER INTERFACES (DUIS) FOR SOCIAL INNOVATION

As users currently input and output information from different devices, the surrounding world becomes an interface to virtually any type of interactive system [14]. DUIS tackle such unknown diversity of monitors, devices, computing platforms, users, physical environments, contexts of use, disciplines, domains and tasks for social innovation. Combinations and interrelations of individually taken resources as user cues create the users' surroundings and function as one logical interaction space. Such resources are nowadays provided by applications that are networked, which in turn are interoperable in federations of devices that automatically support the users' tasks across disciplines and domains [15].

DUIS can provide the digital, organically expanding, ecological environment for Social Innovation to occur, by supporting the four Design Thinking (DT) stages. In the *inspiration stage*, instead of creating adaptive systems only based on questionnaires and focus groups, capturing actual behaviors in situ with context awareness devices can provide us with invaluable clues about their range of unmet needs. Observation of specific unique users' improvisation of their way through their daily lives (shadowing) can be visualized into narrative paths. Summaries of such inspirational resources are made visible as "briefs" in the ideation process. Solutions or opportunities for change are studied to multiply options. The purpose is to create interactive experiences, choices and different insights about human behavior in order to initiate new theories, products or processes. In this phase, a multi-platform system can control budgets and monitor timelines in order to narrow down solutions providing collaboration across disciplines. Through a brainstorming process, different groups may generate hundreds of ideas ranging from the absurd to the obvious via crowdsourcing. The system can support the *implementation* phase by tracking the paths of mock and actual prototyping, production, communication, marketing and distribution, especially if the social innovation is occurring within the community. Lastly, in order to maintain sustainable results we added another I (*Impact*) creating the *4Is model for Design Thinking*.

Design Thinking DUI Guidelines for Social Innovation

Specific DUI guidelines for social innovation support not only different networks, computers and interfaces but also diverse users and ideas formats, users' paths, tasks, contexts of use and disciplines, presentation of ideas and innovations as well as the different modalities of interaction. Anchored in previous research [16], the Design Thinking DUI Guidelines are:

1. *Presence & Co-presence*: profiles based on local time, space and situation (demographics, individual prior associated knowledge, technical information, associated competences, time use during the week and associated activities).

2. *Time continuum*: timeline overview and group calendar depending on the different ways users view and utilize time (linear, cyclic and branching time [17]).
3. *Context continuum*: information, interactivity, activities' overview, organization, coordination, support and control the pace and cyclical time rhythms of the group or community, impact and sustainability.
4. *Situation experience*: local information on group activity via techniques such as visualization to enhance users' local experience.
5. *Intention and experience continuum and convergence*: gradual transitions without focusing on abrupt changes or discontinuities, denoting relations from a bird's eye view, supporting branching time, identification of intentions that cause transitions and anticipations in relation to in situ experience, and organization & knowledge convergence.
6. *Dynamic plasticity, migration, adaptation, and interface continuity* to support input and output for diverse information and across monitors, devices, computing platforms, users, physical environments, contexts of use, domains and tasks.
7. *Adaptation to users' natural gestures, learnability, habits and multitasking (soft modeling)*: changes to positions and sizes when context is changing (e.g. bar sizes for drop-down menus can be hidden, vertically or horizontally presented), adaptation to individual use on the interface (e.g. adaptation of buttons, menus and tools to Parkinson disease patients' movements on the interface).
8. *Real-time evaluation*: utility, usability and acceptability of the proposed system, self, and group evaluation and shadowing availabilities, mirroring actions for individuals, small-groups and overall communities.
9. *Motivation and enjoyment*: enjoyable interface design to promote enthusiasm and inspiration on individual, small-group and community level.

The above Design Thinking DUI Guidelines can function on a multiplatform and sustain actions taken by individuals, small-groups and communities within an unknown spectrum of space-time continuum.

A technical implementation overview for such Social Innovation Management System is presented in Fig. 1:

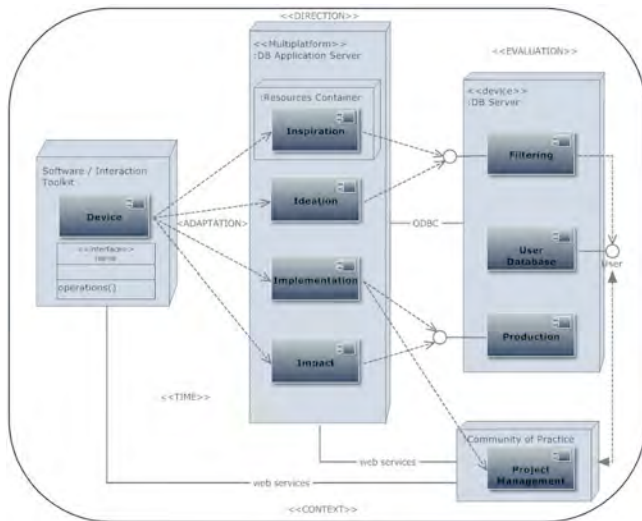


Figure 1. DUIS Design Thinking for Social Innovation

CONCLUSION

The need for knowledge actualization and limitation of the artificiality of information input and output around us is evident in the early 21st century. For such humane oriented approach we needed to create several innovative frameworks to support the diversity of users, information, devices, interfaces, operating systems and user interaction toolkits towards a more holistic perspective. Adaptability and personalization to handle such diversity is feasible via Distributed User Interfaces (DUI) modeling. An HCI approach targeted at human values and the augmentation of human cognition, intellect and emotion was introduced; this Humane Machine Interaction Design was followed by 5 measurable milestones. Then, based on the inherent human need for imagination, co-creativity, learning, work and fun, and the need for a viable solution to the current economic crisis, we suggested that Social Innovation can address these individual and social needs. Lastly, Design Thinking is found to be useful to accommodate Social Innovation based on the 4Is, Inspiration, Ideation, Implementation and Impact. Such multifaceted approach was supported by a set of Design Thinking DUI Guidelines.

REFERENCES

1. Tarde, G. (1903). *The Laws of Imitation*, translated by E.C. Parsons with introduction by F.Giddings. New York, NY: Henry, Holt and Co.
2. Englebart, D. C. (1962). A conceptual framework for the augmentation of man's intellect. In P. Howerton (Ed.), *Vistas in Information Handling* (Vol. 1). Washington, D.C.: Spartan Books.
3. Greenfield, P.M. (2009). Technology and Informal Education: What Is Taught, What Is Learned. *Science* 2, 323, No5910, 69-71.

4. Bronson, P. & Merryman, A. (2010). The creativity crisis. On *Newsweek Magazine*. Available online: <http://www.newsweek.com/2010/07/10/the-creativity-crisis.html>. Last access 03/10/2010.
5. Dix, A., Finlay, J., Abowd, G. & Beale, R. (1998). *Human-computer interaction*. Hempstead, UK: Prentice Hall.
6. Preece, J. (2000): *Online Communities - Designing Usability, Supporting Sociability*. New York: John Wiley and Sons, Chichester.
7. Harper, R., Rodden, T., Rogers, Y. & Sellen, A. (2008). *Being Human: Human Computer Interaction in 2020*. Microsoft Research, Cambridge, U.K., 2008
8. Suchman, L.A. (1987). *Plans and situated actions: The problem of human-machine communications*. Cambridge, UK: Cambridge University Press.
9. ISO FDIS 9241-11. (2007). *Ergonomic Requirements for Office Work with Visual Display Terminals, Part 11: Guidance on Usability Specification and Measures*.
10. ISO 9126-1. (2001). *Quality model - Product quality*. Software engineering.
11. Zentrum für Soziale Innovation (Centre for Social Innovation) Report (2008). *Stimulating Social Development* (p. 2). Download available at <http://www.zsi.at/de/publikationen/346/5056.html>
12. Hochgerner, J. (2010). The Concept of Social Innovation. Presentation in the Workshop Knowledge and Innovation Culture, Academic Conference Centre, October 11-12, Prague.
13. Brown, T. & Wyatt, J. (2010). Design Thinking for Social Innovation. *Stanford Social Innovation Review*, Winter 2010.
14. Vanderdonckt, J, 2010, Distributed User Interfaces: How to Distribute User Interface Elements across Users, Platforms, and Environments. *Interraccion 2010*, 7-10 September, Valencia, Spain.
15. Luyten, K., Coninx, K.(2004). ImogI: Take Control over a Context-Aware Electronic Mobile Guide for Museums. In the Workshop on HCI in Mobile Guides, in conjunction with *6th International Conference on Human Computer Interaction with Mobile Devices and Services*, 3-16 September 2004. University of Strathclyde, Glasgow, UK.
16. Lambropoulos, N. (2010). Group-Awareness Guidelines to Support Online Collaboration in CSCW-L. Workshop presentation in the *24th HCI 2010 Conference*, 6-10 September 2010, Dundee, Scotland.
17. Müller, W. & Schumann, H. (2005). Visualization of time-dependent Data: Methods & Challenges. In the *Workshop on Visual Analytics*, June 4th, 2005, Darmstadt, Germany.

ZOIL: A Design Paradigm and Software Framework for Post-WIMP Distributed User Interfaces

Michael Zöllner, Hans-Christian Jetter, Harald Reiterer

University of Konstanz, Human-Computer Interaction Group

{michael.zoellner,hans-christian.jetter,harald.reiterer}@uni-konstanz.de

ABSTRACT

We introduce ZOIL (Zoomable Object-Oriented Information Landscape), a design paradigm and software framework for post-WIMP distributed, zoomable and object-oriented user interfaces. This paper presents ZOIL's design principles, functionality and software patterns to share them with other researchers. Additionally, ZOIL's implementation as a software framework for C# and WPF (Windows Presentation Foundation) is available as open source under the BSD License (see <http://zoil.codeplex.com>).

Author Keywords

Distributed user interfaces, zoomable user interfaces, post-WIMP user interfaces, software framework.

ACM Classification Keywords

H5.2. [Information interfaces and presentation]: User Interfaces.

General Terms

Design, Human Factors, Languages

INTRODUCTION

To this day, most DUI research has focused on the distribution of standard Graphical User Interfaces (GUIs) and their widgets (e.g. [2, 14]). We introduce ZOIL, a design paradigm and software framework targeted at post-WIMP (post-“Windows Icons Menus Pointer”) multi-touch or tangible UIs that are distributed over multiple devices and displays. Such DUIs are particularly challenging: While the essence and the building blocks of a standard desktop “WIMP” GUI are well established, there's no such concise pattern or blueprint for the post-WIMP world. Thus, the implementation of such UIs is not explicitly supported by the established UI toolkits and programming models and requires great expertise in many heterogeneous soft- and hardware technologies. ZOIL provides help for design and implementation: In the initial stage of design, the ZOIL design paradigm suggests possible patterns of solution as heuristics for a post-WIMP DUI architecture and its visualization and interaction techniques. The prototyping and implementation phases are supported by the ZOIL software framework that facilitates an efficient implementation of ZOIL's defining features [8, 18].

THE ZOIL DESIGN PARADIGM

Applications following the ZOIL design paradigm integrate all kinds of information items from the application domain with all their connected functionality and with their mutual relations into a single visual workspace under a consistent interaction model [10]. This visual workspace is called the “information landscape” and its interaction model follows the principles of Zoomable and Object-Oriented User Interfaces (ZUIs [16, 20], OUIs [3, 13]): All information items are integrated into a single object-oriented domain model and appear as visual objects at one or more places in the information landscape. This way, the domain model becomes directly user-accessible similar to the “naked objects” pattern of Pawson [15]. Unlike in today's desktop metaphor, the information landscape is not limited to the visible screen size but resembles a virtual canvas of infinite size and resolution for persistent visual-spatial information management. In the information landscape all items and their connected functionality can be accessed directly in an object-oriented manner [3] without the use of application windows simply by panning to the right spot in the information landscape and zooming in [20]. ZOIL thereby uses “semantic zooming” [16] which means that the geometric growth in display space is not only used to render more details, but to reveal more and semantically different content and all available functionality “on-the-spot”. This smooth transition between iconic representation, meta-data, and full-text/full-functionality prevents the problems of information overload and disorientation which are caused by traditional WIMP approaches with multiple overlaying windows or occluding renderings of details-on-demand.

Although the previous description focuses mainly on a single user's view, ZOIL's key strength lies in its great flexibility in distributed and collaborative scenarios: Using ZOIL, DUIs can be designed either as *homogeneous* or *heterogeneous* [2], i.e. the UIs can either offer multiple instances of the same user interface at different sites or the functionality provided at each site may be different. This is achieved by the real-time distribution of the object-oriented data model of the information landscape to all devices in an interactive space (Fig. 1). Every connected client is free to choose which part of the landscape it provides to the user. Thus, users can navigate and manipulate the shared landscape independently using the full breadth of available device-specific input and output modalities. A client is also free to choose other visualization styles, e.g. for a device-, user-, or task-specific screen representation.

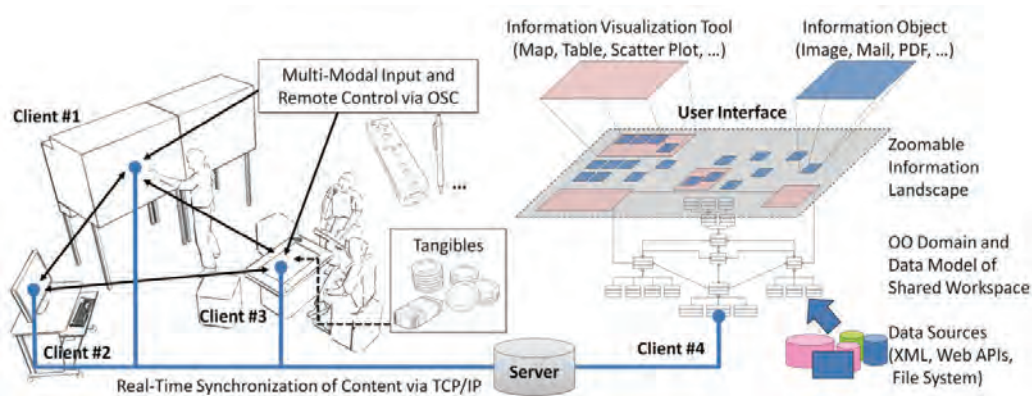


Figure 1. A ZOIL-based multi-user, multi-surface, and multi-device interactive space (left). The right section shows the architecture and user interface of a ZOIL client application in greater detail.

THE ZOIL SOFTWARE FRAMEWORK

Implementing a post-WIMP distributed user interface following the ZOIL design paradigm without tool support is a difficult task that demands a wide range of expertise ranging from distributed databases to vector-based rendering of UI controls. To facilitate ZOIL's application in practice, we have developed the ZOIL software framework [19] that provides the necessary tool support as an extensible collection of software components and classes for C#/WPF. Although individual aspects of ZOIL's implementation are already covered by existing frameworks such as Piccolo [1] or DiamondSpin [21], incompatibilities between languages, UI frameworks, and device SDKs have led us to build a new custom framework for our purposes. The framework has been used for the implementation of various user interfaces (e.g. [9, 17]) and its API usability was evaluated in a longitudinal study [5]. In the light of the framework's volume and broad scope, we focus our description here on defining features to share our lessons learned from the implementation. We describe successful software patterns and architectures used in our framework that can be employed by other researchers in own projects to realize important functionality for post-WIMP UI design.

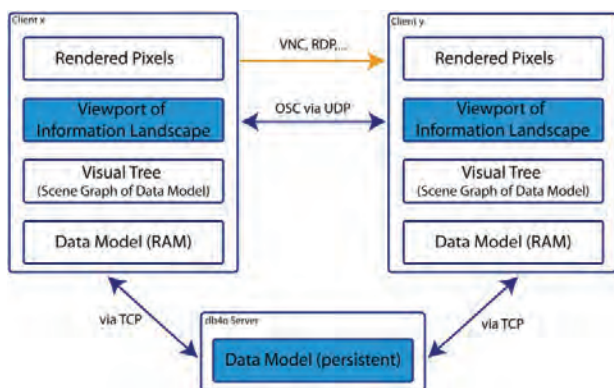


Figure 2. Overview of ZOIL's distribution capabilities

Client-Server Architecture with Transparent Persistence

For realizing the distribution from Fig. 1, ZOIL employs a central database server for real-time synchronization and

persistence. Following the nature of the real world, ZOIL's objects and information landscape do not lose their individual state, properties, or location when disappearing from the screen or after closing the application. Furthermore they are synchronized with all other connected clients in real-time to allow for multi-device and multi-user interaction. We have decided to distribute the UI by synchronizing the data model of the shared visual workspace instead of the UI's visual appearance using protocols on the "pixel level" such as VNC or RDP. Thus, the overall network load of a ZOIL setup is relatively small, since only small changes of the data model are transmitted (Fig. 2). To achieve this, the ZOIL software framework uses the object database db4o [4] with its Transparent Persistence (TP) mechanism. The aim of TP is to allow programmers to access or change persistent data in their application with the same ease as non-persistent data models in main memory. This means that with TP and db4o, ZOIL developers are not concerned with persistence at all since persistence simply becomes a natural property of all ZOIL objects. Using TP, programmers can simply change the states and locations of objects on the client-side using standard C# code. Invisible to the programmer, all changes are observed by db4o's TP implementation. It collects all changes and transmits them via TCP/IP to the database server in regular intervals (typically 100ms). The server persists the new state of the landscape and informs all other clients about the changes which are then automatically retrieved and executed on the remote clients within 100-300ms (in typical setups). Thereby the use of the object database db4o instead of traditional Object-Relational Mapping libraries (e.g. Hibernate) also facilitates iterative design, since changes in the objects' class definitions in C# are directly recognized by db4o without the need to update external XML mappings.

Model-View-ViewModel Pattern (MVVM)

TP offers low-threshold persistency for data objects, but it cannot be applied directly on WPF's visual hierarchies or user controls. Moreover such exact replications of widgets or controls would not be desirable in ZOIL, since each

client should provide the user with an individual view on the landscape's shared data model, so that users can independently navigate and manipulate content. For this reason, the ZOIL framework uses the MVVM Pattern [7] to provide a novel way of separating the persistent data *model* of an object from the non-persistent *view* of the object in the landscape. Using TP, each object's *model* is shared via the server with all other clients, but the corresponding *view* is not shared and only resides on the client-side. As soon as a *model* is created (or destroyed) in a local or remote client, the corresponding *views* on all other clients are created (or destroyed) accordingly. The key advantage of integrating MVVM in ZOIL over using a traditional MVC approach lies in the *ViewModel*, which plays the role of an adapter between the XAML (Extensible Application Markup Language) markup of the *view* and the C# *model* by tailoring the data of the model to a view-friendly format. Properties and commands of the *ViewModel* are bound via two-way data binding to the *view*. *Model* and *ViewModel* always stay synchronized, so changes in the model get routed to the view and vice versa. Using MVVM, *views* can also be defined more conveniently in declarative XAML, without the need for C# code.

```
<ZEmail x:Class="ZComponent" ... >
  <ZComponentFrames>
    <ZComponentFrame WidthNeeded="0">
      ... appearance at 1st zoom level ...
    </ZComponentFrame>
    <ZComponentFrame WidthNeeded="150">
      ... appearance at 2nd zoom level ...
    </ZComponentFrame>
    ... add appearances for more zoom levels here ...
  </ZComponentFrames>
</ZEmail>
```

Code Sample 1. XAML code for defining the semantic zoom of an email object.

Declarative Definition of Zoomable UI Components

One key feature of our framework is the comprehensive support for authoring and implementing rich ZUIs with semantic zooming using WPF and its declarative XAML language. Unlike Java's SWT, Swing, or WinForms, WPF natively supports vector-based and hardware-accelerated rendering of controls. This enables ZOIL designers to use the full range of WPF controls (e.g. sliders, player controls) in ZUIs without pixelation or the need for wrapper classes as in [1]. This also allows programmers to easily integrate rich media content such as video streams or 3d models and increases the prototyping efficiency in comparison to other hardware-accelerated graphics APIs without any or only very basic sets of user controls (e.g. OpenGL, DirectX). A further advantage of WPF is the possibility to separate the object's implementation logic in C# from its visual design in XAML. This allows designers to use a low-threshold HTML-like language and visual editors for defining an object's appearance. As a consequence, ZOIL supports typical designer-developer workflows and lowers the threshold for non-programmers to use the framework.

Following this declarative paradigm, we have extended the available set of controls and behaviors in XAML with a collection of ZOIL-specific components that enables designers to define semantic zooming and other behaviors without the need for procedural C# code (Code Sample 1). The different appearances are selected by ZOIL's `ZComponentFrames` container depending on the current display size and are exchanged using a smooth zooming and opacity animation.

```
<ZPhoto
  ZObjectDragDropBehavior.IsDraggable="True"
  ZObjectResizeBehavior.IsResizable="False"
  ZObjectRotateBehavior.IsRotatable="True"
  ZObjectPhysicsBehavior.HasInertia="True"
  ZInformationLandscape.ZoomTarget="True">
  ZInformationLandscape.ZoomMargin="5,10,5,5">
  <!-- further definition of ZEmail -->
</ZPhoto>
```

Code Sample 2. XAML code for making ZPhoto a draggable, resizable, and rotatable zoom target.

Attached Behavior Pattern (ABP)

For designing object-oriented interaction it is not sufficient to define an object's visual appearance alone. Objects also have to be assigned the desired behaviors: Can an object be dragged, resized, or rotated using multi-touch manipulations? Does an object simulate physical behavior such as inertia or friction during these manipulations? Is the object a zoom target, so that a click or tap on it starts a zooming animation into the object until it covers the screen? Or should there be a small margin remaining around the object? To help designers to easily assign such behaviors to an object without reinventing the wheel for every type, our framework makes extensive use of the Attached Behavior Pattern (ABP) [6]. ABP encapsulates behaviors into a class outside the object's class hierarchy and allows it to be applied on classes or individual instances of objects. When using WPF this threshold can be further lowered using XAML instead of C# (Code Sample 2). We believe that the Attached Behavior Pattern introduces a very natural view of interactive behavior into post-WIMP programming that facilitates iterative design without the need for changes in procedural C# code or class hierarchies. Furthermore by using ABP, a framework can provide designers with an extensible library of potentially useful behaviors that can be easily assigned in visual editors.

Input Device and Client-Client Communication with OSC

The ZOIL framework provides simple ways to connect to multi-modal input libraries or other kinds of input device middleware [11, 12]. Applications can use the stateless UDP-based OSC protocol to connect to open source tools like Squidy [11] that facilitate the integration of external post-WIMP devices such as Anoto pens or the Nintendo Wiimote. OSC has several advantages for prototyping: First, clients stay tolerant to unavailable devices or changing infrastructures during their lifetime, since no permanent connections have to be established. Second,

UDP packets can be broadcasted to all clients in the subnet and therefore sending to specific IP addresses is unnecessary. Instead, packet destinations can be specified above the network layer using OSC addresses. This is desirable in early phases of prototyping and experimentation where devices are often added or removed. In some scenarios (e.g. in an augmented meeting room) it might be important to tightly couple two or more clients' views. By sending OSC commands between clients it is possible to control the current camera or viewport of a remote client from a local client. This allows scenarios of collocated collaboration between multiple users and devices. For example, a view on a region of the landscape can be transferred from a personal device to a public large display or vice versa. OSC can also be used to implement an overview and detail coupling between two clients.

CONCLUSION

We have introduced the ZOIL design paradigm and software framework. ZOIL is unique in its focus on distributed post-WIMP user interfaces that are based on a real-time synchronized multi-user/multi-display/multi-device visual workspace. A ZOIL setup therefore follows Melchior et al.'s notion of a Distributed User Interface: "A DUI consists of a UI having the ability to distribute parts or whole of its components across multiple monitors, devices, platforms, displays, and/or users" [14]. Given the broad design space of post-WIMP user interfaces, a generic design or implementation of "optimal" distribution is very difficult, if not impossible. In a ZOIL setup, it is possible to leave such decisions to each connected client, thereby avoiding the risk of losing usability or functionality and getting the most out of every client's input and output modalities. While ZOIL is inherently multi-monitor, multi-display, multi-device and multi-user, it is not truly multi-platform yet, since it only supports the MS Windows platform. We will evaluate if future versions of ZOIL can leverage Microsoft's Silverlight platform to come closer to a multi-platform solution.

REFERENCES

1. Bederson, B., Grosjean, J., and Meyer, J. Toolkit design for interactive structured graphics. *IEEE Trans. Softw. Eng.* 30 (August 2004), 535–546.
2. Bharat, K., and Brown, M. H. Building distributed, multi-user applications by direct manipulation. In *Proc. UIST 1994*, 71–80.
3. Collins, D. *Designing Object-Oriented User Interfaces*, 1st ed. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1994.
4. db4o. Native java & .net open source object database. <http://www.db4o.com>.
5. Gerken, J., Jetter, H.-C., Zoellner, M., Mader, M., and Reiterer, H. The concept maps method as a tool to evaluate the usability of apis. In *Proc. CHI 2011*.
6. Gossman, J. The attached behavior pattern. <http://blogs.msdn.com/johngossman>.
7. Gossman, J. Introduction to the model view viewmodel pattern. <http://blogs.msdn.com/johngossman>.
8. Jetter, H.-C., Gerken, J., Zöllner, M., and Reiterer, H. Model-based design and prototyping of interactive spaces for information interaction. In *Proc. HCSE 2010*, 22–37.
9. Jetter, H.-C., Gerken, J., Zoellner, M., Reiterer, H., and Milic-Frayling, N. Materializing the query with facet-streams – a hybrid surface for collaborative search on tabletops. In *Proc. CHI 2011*.
10. Jetter, H.-C., König, W. A., Gerken, J., and Reiterer, H. Zoil - a cross-platform user interface paradigm for personal information management. Personal Information Management: The disappearing desktop (a CHI 2008 Workshop).
11. König, W., Rädle, R., and Reiterer, H. Interactive design of multimodal user interfaces - reducing technical and visual complexity. *Journal on Multimodal User Interfaces* 3, 3 (Feb 2010), 197–213.
12. Lawson, J., Al-Akkad, A., Vanderdonckt, J., and Macq, B. An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. In *Proc. EICS 09*, 245–254.
13. Mandel, T. *The GUI-OOUI War, Windows vs. OS/2: the designer's guide to human-computer interfaces*. Van Nostrand Reinhold Co., New York, 1994.
14. Melchior, J., Grolaux, D., Vanderdonckt, J., and Van Roy, P. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In *Proc. EICS 2009*, 69–78.
15. Pawson, R. *Naked Objects*. PhD thesis, Trinity College, Dublin, Ireland, 2004.
16. Perlin, K., and Fox, D. Pad: an alternative approach to the computer interface. In *Proc. SIGGRAPH 93*, 57–64.
17. Project Deskpiles. <http://research.microsoft.com/en-us/projects/deskpiles>.
18. Project Permaedia/ZOIL. <http://hci.uni-konstanz.de/permaedia>.
19. Project ZOIL on Codeplex. <http://zoil.codeplex.com>.
20. Raskin, J. *The humane interface: new directions for designing interactive systems*. ACM Press/Addison-Wesley Publishing Co., 2000.
21. Shen, C., Vernier, F., Forlines, C., and Ringel, M. Diamondspin: an extensible toolkit for around-the-table interaction. In *Proc. CHI 2004*, 167–174.

Improving E-Learning Using Distributed User Interfaces

Habib M. Fardoun, Sebastián Romero López, Pedro G. Villanueva

Laboratory of Interaction Systems Everywhere

University of Castilla-La Mancha, Albacete, Spain

{Habib.Moussa, Sebastian.Romero, Pedro.Gonzalez}@uclm.es

0034 967 599 200 ext. 2689

ABSTRACT

As we can observe day after day the m-Learning methodology has become an exciting art of using mobile technologies to enhance learning skills. Mobile phones, PDAs, Pocket PCs and the Internet can be joined together in order to engage and motivate learners, anytime anywhere. The society is entering a new era of m-Learning, which makes important to analyze and innovate the current educational tools so they can be used correctly and as it is needed in the application domain of the work “learning”. This article proposes the way in which MPrinceTool, that aims to improve the deficiencies identified in the usual analysis, can provide a new means to interact via mobile, desktop and Web by applying distributed user interfaces to the application design, so it facilitates users to participate in educational activities and communication with working groups. These advantages will be clearly adapted by our tool and part of it will be explained.

Author Keywords

M-Learning, distributed user interfaces (DUI), HCI technology, mobile devices, context awareness, and collaborative environments.

ACM Classification Keywords

HCI; DUI; MDA; MB-UIDE; user interfaces, mobile devices, context awareness, collaborative environments, and user-centred design.

General Terms

Design, experimentation, human factors.

INTRODUCTION

By the end of 2009, there were an estimated 4.6 billion mobile cellular subscriptions, corresponding to 67 per 100 inhabitants globally [5]. This is more than three times the number of personal computers (PCs), and today’s most sophisticated phones have the processing power of a mid-1990s PC.

The main idea of this proposal is to present an innovated

interaction way, while using MPrinceTool [4], where users can be able to interact with the environment to build knowledge from relationships acquired from the ambient. And after searching for the best way to improve the implementation of this system, we find that it can be done by supporting distributed user interfaces by the educational tool. The system with several modifications in its user interface design by making it a distributed user interface design to enrich the classroom study allowing students to work in a collaborative way as it is indicated in European Higher Education Area [3].

One of the main concerns for distributed user interfaces is the complex and dynamic world of managing user interactions, from different types of devices. One of the main proposals for the development of new devices and systems that support DUI is to provide users an interface divided between devices, introducing new forms of interaction and collaboration [1, 2, 8 and 9].

WallShare (one of the tools that MPrinceTool is based on it) itself [6], implements DUI, so MPrinceTool as well. In this paper we have improved the DUI part of the system by adding new users’ interfaces and functionalities which are more appropriate for educational environments.

The structure of the paper is as follows: first we describe the system (server and client functionality and exhibit its architecture), secondly, the state of art is presented, thirdly, the system and the different user interfaces, fourthly, how MPrinceTool supports DUI is discussed and the advantages are explained with a case study. And finally, the conclusions that have been observed and obtained after using the tool and the future work upon it will be presented.

STATE OF ART AND MOTIVATION

There are more and more educational tools for mobile device use, and each is specialized in a particular field. The two tools that we developed and merged together are: Interactive Learning Panel [8] and WallShare [6].

Interactive Learning Panel

Using this device, the students can relate information that is presented by the mobile application to a physical region of a panel. It is based on the idea of relating concepts with lines, or multiple choice questionnaires.

Concretely, they have implemented a PDA web based application where users have to relate a flag to a region on a

map. For instance, if a flag is given on the PDA screen, students have to relate it to the country depicted on the map and vice-versa.

These panels are equipped with RFID tags that represent concepts on the panel and the PDA is equipped with a RFID reader that is able to read these tags and detect how user relate these concepts through their readings. So, to relate the concepts exposed in the PDA screen to those on the panel, users have to put the reader (mobile device) over the graphical representation in the panel.

WallShare

Is a system based on a shared zone, that is projected on a wall or screen, which is clearly visible by the participants of the meeting.

In order to interact with the system, each participant connects his mobile device to the system.

Once the connection procedure was accomplished successfully, a pointer representing the participant is shown on the wall or screen.

An interesting aspect of WallShare is the possibility of providing each participant with the capability of controlling the movement of his cursor on the screen by performing dragging gestures on the mobile device screen. Thus, users can use the mobile device as an enhanced X-Pointer device.

Therefore, connected participants can download and upload all kind of resources from and to the shared zone.

Thus, participants are able to share a resource just by uploading it to the shared space. To perform this task, users select a resource on the mobile device through the client application and upload it performing a simple gesture. When the resource was uploaded, it is shown on the screen.

Distributed User Interfaces (DUI)

A Distributed User Interface is a user interface with the ability to distribute part of its components, or all of them, across multiple screens, devices, platforms and users. DUI is divided into several parts that cooperate to facilitate the user work. Its main objective is to provide the users, of the mobile devices, all the tasks they need, providing them an optimal configuration of the available interaction resources around them [7 and 9].

DUI can be used for several reasons, such as: sharing information between users, assigning tasks, divide the public and private information and display it on different screens, dividing the work space into several parts and on multiple screens. Here in this paper we will demonstrate that MPrinceTool take into consideration all these points which make it supports DUI correctly.

Following, the explanation of how we have used the distributed user interfaces in the system in order to improve it.

SYSTEM DESCRIPTION

MPrinceTool system is presented in a projected area onto a wall or large screen, which is clearly visible to students who use the system.

The functionality of the system can be divided into two parts. First, we have the functionality of the client and, second, we have the server functionality (see Figure 1).

The Server Functionality

The server application (MPTServer) is responsible for controlling the customer interaction and display all the information that students and teachers need.

MPTServer shows, or projects, in a split screen the questions that teachers asked to students, who are connected to the system; they make use of the chat room to comment on the required question at a given time and about the resources that the teachers share with their students.

Thus the shared area that is projected is divided into four parts as shown Figure 1.

The upper left region is the reserved area for the presented question by the teachers. It shows the formulation of the question and all its possible answers.

One of the uses of the tool is that, next to each answer there is a counter that indicates the number of students who have selected this response. Each user can select from his device the answer that he think it is the correct choice using his associated pointer in the shared pool.



Figure 1. General view of MPrinceTool

The upper right region is reserved to indicate students who are currently online at MPTServer. Each student, who is connected to the server, is represented by his name and an image in this region. So the participants can easily identify each other.

The bottom left corner contains the resources (images, video, audio, or other file type) that teachers and students share; these materials are usually uploaded to help them in answering the questions.

Finally the lower right region is a chat room where students can talk to each other to interchange ideas or to explain

their opinion from the current question in order to answer the question correctly.

The Client Functionality

It has six client applications, some versions that offers the functionality that the student can perform are: MPTStudent, MPTW Student and MPTD Student; and the others versions includes additional functionality that only teachers can do: MPTTeacher, MPTW Teacher and MPTD Teacher. The mobile interfaces: MPTStudent and MPTTeacher were presented in [4]. In this paper we present the new interfaces of the system. The Figure 2 shows the whole system.

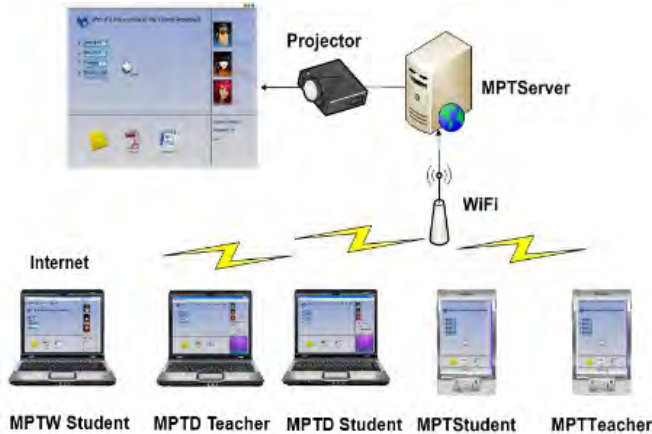


Figure 2. Functionality of MPrinceTool system.

The server system runs on a desktop computer that it is connected to a wireless network via a Wi-Fi connection. It is also connected to the visualization system, i.e. a projector or large screen, which supports the shared zone visualization by the participants.



Figure 3. MPTD Student interface

The desktop-client has two different interfaces, one for the teacher and other for students: MPrinceTool Desktop Teacher Interface and MPrinceTool Desktop Student Interface, which are identical (see Figure 3).

Both desktop interfaces have the control of the pointer from the shared pool at the bottom left. Before using these interfaces users have to install the software on their personal computers.

In order to make the system multi-platform **Web Interfaces** have been created for both users: MPrinceTool Web Teacher (see Figure 4) and MPrinceTool Web Student (MPTW Student). Both interfaces have the same functionality as desktop applications, with the exception of the control panel of the pointer for the system's shared interface as the Web implementation of this functionality is more complex. Unlike desktop and mobile applications, the Web applications don't connect with the Wifi server, they connect to the system via internet.

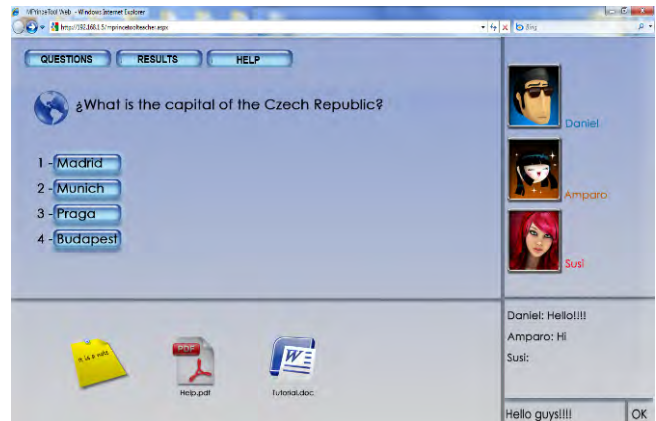


Figure 4. MPTW Teacher interface

In the four interfaces presented in this paper, users have to log on before starting to use the application.

SUPPORTING DISTRIBUTED USER INTERFACES

As we can observe, in all the figures presented in this paper in the explanation of the system description, MPrinceTool facilitate the tasks, to be used by the users, by implementing the concept of distributed user interfaces. This is done through the interaction performed by the users on their personal computers and mobile devices affecting the whole system. This improvement is clearly seen by comparing the use of a digital whiteboard in class, used by a single individual student and how many students can make use the system at the same time.

MPrinceTool its build as a distributed user interface system using some of the most widely extend hardware elements: like personal computers and mobile devices, which make it better than the many emerging technologies, and other already established in human-computer interaction, which needs unpopular hardware elements. for example: the Microsoft Kinect, Nintendo Wii., Microsoft Surface etc., by making use of infrared cameras, remote controllers etc. which makes it be used just by specific systems (as example: Wii remote controller).

Case Study

As a case study, the *help support* among students is presented. This functionality has been enhanced thanks to the implementation of DUI.

Applying this task, users can ask for help typing it in the chat as in old versions, the possibility of asking for help is introduced in this new version of the system. Therefore other classmates could offer their help in another way and not just by using the chat tool. Also, since this support is supervised by the teacher in his desktop application, he can accept the help request as well. The way in which it works is like this: first a student asks for a help, and any of the classmates could respond to his request. After that, the help communication takes place at the students' devices, letting the shared zone for chatting free for the rest of the class.

For example, when *Pedro* tries to resolve the mathematics problem (who to obtain the area of a cycle), he was not able to perform it alone. So Pedro asks for a help using the communication area and *Sebastian* offer himself as a voluntarily to lend him a hand. Once the teacher, *Jose*, accepted this communication, Pedro and Sebastian start a chat in a private way, without disturbing the rest of the class.

We can observe that, the tool has progressed by supporting DUI. Now, the private groups of students using MPrinceTool can have freer and cleaner chat than before, letting the rest of the class continue with its usual pace.

CONCLUSION AND FUTURE WORK

This paper presents the improvement of MPrinceTool by supporting distributed user interfaces. The system has six different interfaces which give the system advantages: The Web interfaces offer the application platform independence, and using Mobile interfaces offers all the advantages of the mobile technologies and as well its disadvantages which are compensated by the functionality of the desktop-pc interfaces. Besides this, with the implementation of the Web interfaces we have got a target of past works: that MPrinceTool can be used in another platforms and operating systems.

In the educational part, this tool helps students to participate more in class and join and help each other in the educational activities. It allows them to interact with the learned subjects, and teachers can have more detailed information about student's progress. These details can be extended in future versions adding some kind of gift or note to those students who help classmates.

ACKNOWLEDGMENTS

We would like to thank CENIT project (CENIT 2008-1019-MIO!) for funding this work.

REFERENCES

1. Christian Kray, Daniel Nesbitt, John Dawson, and Michael Rohs. User-defined gestures for connecting mobile phones, public displays, and tabletops. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI '10)*. 2010
2. Dominik Ertl, Hermann Kaindl, Edin Arnautovic, Jürgen Falb. Generating High-Level Interaction Models out of Ontologies and Roman Popp. *IUI SEMAIS Workshop 2011*.
3. EHEA, European Higher Education Area. Last access 01/01/2011. http://ec.europa.eu/education/lifelong-learning-policy/doc62_en.htm
4. Habib M. Fardoun, Pedro G. Villanueva, Juan Enrique Garrido, Gabriel Sebastian Rivera, and Sebastian Romero Lopez. 2010. Instructional m-Learning System Design Based on Learners: MPrinceTool. In *Proceedings of the 2010 Fifth International Multi-conference on Computing in the Global Information Technology (ICCGI '10)*. IEEE Computer Society, Washington, DC, USA, 220-225.
5. ITU. International Telecommunication Union. Measuring the Information Society. 2010. Last access 12/04/2010. http://www.itu.int/ITU-D/ict/publications/idi/2010/Material/MIS_2010_Summary_E.pdf
6. Pedro G. Villanueva, Ricardo Tesoriero, and Jose A. Gallud. Multi-pointer and collaborative system for mobile devices. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI '10)*. ACM, New York, NY, USA, 435-438. 2010
7. Pedro G. Villanueva, José A. Gallud, and Ricardo Tesoriero. 2010. WallShare: a multi-pointer system for portable devices. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI '10)* ITU. International Telecommunication Unio.
8. 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.
9. Jan Van den Bergh, Gerrit Meixner, Kai Breiner, Andreas Pleuss, Stefan Sauer, and Heinrich Hussmann. Model-driven development of advanced user interfaces. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems (CHI EA '10)*. 2010

Lessons Learned from the Design and Implementation of Distributed Post-WIMP User Interfaces

Thomas Seifried¹, Hans-Christian Jetter², Michael Haller¹, Harald Reiterer²

¹Upper Austria University of Applied Sciences

Media Interaction Lab

{thomas.seifried, haller}@fh-hagenberg.at

²University of Konstanz

Human-Computer Interaction Group

{jetter, reiterer}@inf.uni-konstanz.de



Figure 1: Interactive spaces based on post-WIMP DUIs. (a) NiCE Meeting Room [5], (b) DeskPiles [2] and (c) Facet-Streams [10]

ABSTRACT

Creating novel user interfaces that are “natural” and distributed is challenging for designers and developers. “Natural” interaction techniques are barely standardized and in combination with distributed UIs additional technical difficulties arise. In this paper we present the lessons we have learned in developing several natural and distributed user interfaces and propose design patterns to support development of such applications.

Author Keywords

Post-WIMP, natural user interfaces, distributed user interfaces, zoomable user interfaces, design patterns.

ACM Classification Keywords

H5.2. [Information Interfaces and Presentation]: User Interfaces

INTRODUCTION

In the recent years, the ever-increasing miniaturization of interactive products with great computational power and network connectivity has resulted in a proliferation of computing into almost every facet of our physical and social world. This process has turned many aspects of Mark Weiser’s vision of ubiquitous computing into a part of our daily life. However, as HCI researchers, we consider two essential challenges of ubicomp as still unmet: creating novel user interfaces that are natural and distributed.

Our understanding of Distributed User Interfaces (DUI) is based on Melchior et al. [11]: a DUI is a UI with the ability to distribute parts or whole of its components across multiple monitors, devices, platforms, displays and/or users. For our research, this ability is essential to realize interactive spaces [16] in which multiple interactive surfaces and devices act as one distributed UI for co-located collaboration (Figure 1). In these spaces we try to achieve a

“natural” interaction, i.e. the UI is perceived as something unobtrusive or even invisible that does not require the users’ continuous attention or a great deal of cognitive resources. A well-proven approach to achieve this are visual model-world interfaces for “direct manipulation”, in which a tight coupling of input and output languages narrows the gulfs of execution and evaluation [7]. While direct manipulation originates from 1980s desktop computing, its principles are also the foundation of novel post-WIMP (post-“Windows Icons Menu Pointing”) or reality-based UIs [8]: Their interaction styles (e.g. tangible, multi-touch or paper-based UIs) “draw strength by building on users’ pre-existing knowledge of the everyday, non-digital world to a much greater extent than before.” Users can apply the full breadth of their natural, non-digital skills, e.g. the bimanual manipulation of objects, handwriting or their awareness of objects or persons in their physical and social environment.

Our two research groups have designed and implemented a great variety of such post-WIMP distributed UIs for co-located collaboration in augmented meeting rooms [5], on tabletops for scientific discussion [2] or for collaborative product search [10] (Figure 1). Based on these experiences, we have to conclude that the combination of natural and distributed UIs poses a particular hard challenge to UI designers and developers [9]. As discussed by Shaer and Jacob, the typical challenges that creators of natural UIs face are the lack of appropriate interaction abstractions, the shortcomings of current user interface software tools to address continuous and parallel interactions, as well as the excessive effort required to integrate novel input and output technologies [14]. The distribution of natural interactions across device and display boundaries adds greatly to this complexity.

In the following, we summarize our “lessons learned” to share them with DUI researchers and practitioners by extracting two design patterns (DP) and an anti-pattern (AP). These three patterns address both sides of UI creation: interaction design patterns [1] and software design patterns. All were tested extensively during our projects. While the first two patterns have become a part of our open source software framework ZOIL that facilitates DUI implementation [17], the anti-pattern was implemented, tested and discarded as ineffective. We conclude with a brief summary of our findings and formulate research questions for future work.

DESIGN PATTERNS FOR POST-WIMP DUIS

To understand the origin of our patterns, it is important to notice the commonalities of the projects from which they are derived: All of them are aimed at creating interactive spaces for co-located collaboration of multiple users. As shared surfaces we either use large Anoto-pen enabled front-projected surfaces [5] or smaller vision-based multi-touch enabled tabletops (e.g. Microsoft Surface) [9]. Personal and mobile surfaces are realized using sheets of Anoto paper & laptops [5] or tablet PCs [2]. To achieve a natural post-WIMP interaction, the dominant input modalities throughout the projects are multi-touch and/or Anoto pens. Furthermore the design follows a fundamental principle of natural UIs: “the content is the interface” [6]. This means, that the amount of administrative UI controls known from WIMP (e.g. menus, window bars, tool bars) is minimized so that the content objects themselves become the first-class citizen of the UI. This natural provision of content for direct manipulation also has implications on the flexibility of interaction. By abandoning traditional page- or dialog-oriented sequences of interaction (e.g. typical Web applications), users can act directly and flexibly on the objects of the task domain. Apart from multi-touch and pen-based manipulations and gestures, tangible props such as physical tool palettes [5] or glass tokens for query formulation support users in their tasks [10].

DP1: Real-Time Distribution of a Zoomable Workspace

A prerequisite for any kind of collaboration is a shared workspace accessible to all users. As a first interaction design pattern, we therefore suggest the use of a shared visual workspace that uses a 2D virtual plane containing all necessary functionality and content of the application domain as visual objects for direct manipulation. All user changes to the location, orientation, size, annotation or nature of these objects are immediately executed and visualized in real-time. The workspace serves as a model-world representation of the application domain that shares an essential property with the real world: actions on objects lead to immediate feedback and persistent results. Thus the workspace resembles a physical whiteboard for natural, co-located and synchronous collaboration (e.g. in [5]). We extend this pattern further with a Zoomable User Interface (ZUI). ZUIs largely increase the amount of accessible objects because the workspace is not limited to the visible

screen size and becomes virtually infinite in size and resolution. Nevertheless ZUIs still maintain a natural feel during navigation as they tap into our natural spatial and geographic ways of thinking [12]. Thereby “semantic zooming” is employed and geometric growth in display space is also used to render more and semantically different content and functionality. Ideally ZUIs can thereby “replace the browser, the desktop metaphor, and the traditional operating system. Applications per se disappear” [13]. Most importantly, when put on a central server to make it accessible from different clients, such a shared ZUI enables many scenarios of real-time distribution. Every ZUI client can access the local or remote ZUI server to render an arbitrary section of the shared ZUI at an arbitrary zoom level. Thus each client acts as a kind of camera into the shared workspace that users can control using zooming and panning operations. This enables many distribution modes:

(1) By running several instances of a ZUI client on one PC, different views of the real-time synchronized workspace can be displayed simultaneously, e.g. for distributing the workspace to *multiple windows* or *multiple monitors* to create an overview and detail solution. (2) When using *multiple devices* each device can run one or several clients that connect to the central server, so that different devices can access and visualize the same shared workspace. Thus the physical boundaries of devices can be overcome to achieve a cross-device distribution of the workspace. This can be used to provide multiple access points, e.g. several co-located PCs with large touch-enabled vertical or horizontal displays, at a local or even a remote site. Depending on the use case, a device’s view onto the shared workspace can be either tightly coupled with any other device or used completely independent from others. For example a handheld device then always acts as a zoomed out overview of the detail view on a tabletop. (3) The same mechanisms enable the distribution of the workspace to *multiple users*: By introducing personal devices (e.g. smart phones, laptops or tablet PCs) that run a ZUI client, a distribution to multiple users becomes possible (e.g. to several users around a tabletop each carrying a tablet PC for annotation [2]).

We have made extensive use of this interaction pattern in [2] and [10]. We share more details about our software design and implementation choices for this interaction pattern in the following.

DP2: Distributed ViewModels

The Distributed ViewModel simplifies the development of new interaction techniques for DUIs. It provides a transparent mechanism to synchronize view-dependent information among all connected instances of a DUI. Content and interaction are often much closer to each other on Post-WIMP UIs than it used to be in traditional user interfaces. This motivates developers to bring those two properties closer together in the software design. Content and basic view information, such as position and size of a

view, can be easily modeled with standard design patterns, but the interaction itself cannot. Interaction with post-WIMP user interfaces is not as standardized as in WIMP interfaces, therefore UI developers still need **a lot more** freedom to design and test new interaction techniques. But in contrast to non-distributed UIs, designing interaction for DUIs lacks many tools and design patterns and still requires much know-how about the technical background. For example, a DUI developer needs to know how to distribute the UI onto other machines. But network synchronization with all its issues is a very complex topic and a UI developer should not need to worry much about it.

The concept of a distributed view model tries to address those problems by providing a network-synchronized model of a view to ease development of a shared UI. It provides an additional abstraction layer that contains the content of a view as well as view-dependent properties. The distributed view model is based on the Model-View-View-Model (MVVM) design pattern [15]. In the MVVM pattern the data model of an application is separated from its view, similar to the Model-View-Controller (MVC) pattern. In contrast to MVC, MVVM provides an additional abstraction layer, the so-called “ViewModel” which is an abstract description of the view. The “ViewModel” can also be seen as a “Model of the View” containing only view-related information and logic. This allows an UI designer to mainly focus on UI design but still provides a clean interface to the non-UI parts of the application. The Distributed ViewModel pattern, as depicted in Figure 2, facilitates this clean interface to provide a transparent distribution of view-related properties which are defined in the ViewModel. The Distributed ViewModel is much like the ViewModel as it contains the same information, but in contrast its contents and structure are already prepared for network synchronization. All information stored in a Distributed ViewModel is automatically serialized and synchronized with all other connected instances of the application or UI. In practice, the ViewModel can often be completely replaced by a Distributed ViewModel, if data types used in the view are compatible with network synchronization.

In difference to the original MVVM design pattern, the Distributed ViewModel pattern is designed for transparent synchronization of ViewModels among all connected instances. Thereby the Distributed ViewModels are handled as “network shared objects” which update all other networked instances of the same object if a property changes. The update mechanism makes use of a change notification system within the view and ViewModel. If a property of a view and consequently of a ViewModel is modified, the ViewModel fires an event allowing other objects, such as the Distributed ViewModel, to be notified. Consequently, distributed instances of the Distributed ViewModel can be updated accordingly. It is important to note, that the distributed update of these objects needs to

take care of concurrency issues that might arise if two instances of the same objects are changed concurrently.

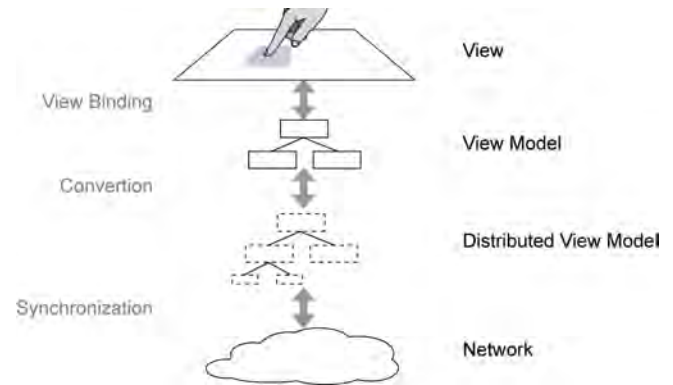


Figure 2: Concept of Distributed ViewModels

The update mechanism of the Distributed ViewModel can be developed in a nearly transparent way. In our DUI applications we provided a base class which hid the network synchronization from the UI development. In our implementation we have used an object database that provides a “transparent persistency” mechanism as backend [17]. Hence a UI developer never came in direct touch with networking issues.

AP1: Input Forwarding (Anti-Pattern)

Not every software design that has been tested was successful. Forwarding of input events from input devices such as the mouse, stylus or keyboard failed. The motivation behind input forwarding is to distribute interaction techniques by simply forwarding the raw input received from the input devices to all other instances of the DUI. For example a touch input event on display A is forwarded onto display B and is processed by both UI instances in the same way. This design is based on the idea that input from a remote UI instance is handled in the same way as local input. Therefore, new interaction techniques on a DUI would be very simple to implement, because the UI developer does not need to care about network synchronization problems at all. The underlying assumption behind this design is that the CPU is a deterministic and standardized system. Hence, the same input always results in the same output. This would also mean that the underlying visualization of the controls does not need to be distributed, because the operations on those would result in the same result on every instance.

This pattern relies on a server-client architecture. All input events are sent to a server on which they are synchronized and ordered. This solves many concurrency problems because every instance gets the same events in the same order. Additionally, since the resolution of the UI may vary on different devices, all input data containing position information (e.g.: mouse-pointer coordinates) need to be normalized before distribution. Accordingly the receiving instance needs to map those normalized input data in its own coordinate space. The UI controls are simply scaled on

such systems, therefore input on a certain control on display A would be also on the same control on display B, even when the resolution is different.

Although this design has been successfully used on single-display, multi-user applications [4] this design failed for DUIs. The state of the UI on distributed instances of the system did diverge after a short time. This was caused by three problems that may occur in such a system: (1) Input coordinates have been normalized by using floating-point values. Since floating-point numbers are imprecise, results of de-normalization on DUIs using displays with different resolutions always contained a small rounding error. This small rounding error can be enough that a button is clicked on one instance and on another it is not. (2) Even when every instance of the DUI used the same resolution, interactions based on relative motion (e.g.: translating a UI element beneath a moving touch point relative to its last position) caused divergence. The GUI system used for rendering also used floating-point numbers to position UI elements. In contrast to our assumption floating-point calculations do not always have the same result on different computers [3]. (3) This design only works if the UI is exactly the same all the time. Even small differences in the layout may result in divergence of the instances.

Therefore, we suggest that input forwarding should only be used if no floating-point values are involved and the layout and resolution of the UI is always exactly the same.

CONCLUSION

In this paper we have summarized our experiences or “lessons learned” from developing several “natural” and distributed UIs. The applications we have developed are all based on either a shared workspace or shared view; therefore our results may be limited to those types of DUIs. Still we believe that some aspects of our results are generalizable and also relevant for other types of DUIs.

In the combination of both, relatively new, fields of UIs a set of new problems arise. Our software and interaction design patterns are the results of our first attempts to solve those problems. However, not all of our attempts were successful. At this early stage of post-WIMP DUI research we therefore think that it is important to also report about these failed approaches, so that other developers and researchers can avoid making the same mistakes.

In the light of the novelty and complexity of the field, several new questions for future research are raised:

(1) The Distributed ViewModel pattern still needs a *developer* but today UIs are usually created by *designers* using UI design tools. Do we need new design tools for DUIs? (2) Some aspects of a user interfaces are command-based, for example undo/redo. Such *command-based interactions* are not handled in our proposed designs. (3) In conclusion we believe it is not enough to research only design patterns and theoretical models, but also develop

real-worlds application on DUI concepts. In our opinion, only by developing, testing and deploying such applications to real-world users we will be able to fully understand the possibilities and constraints of DUIs.

REFERENCES

1. Borchers, J., Buschmann, F. *A Pattern Approach to Interaction Design*. John Wiley & Sons, Inc., 2001
2. Deskpiles. <http://research.microsoft.com/en-us/projects/deskpile>.
3. Goldberg, D. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.* 23, 1 (March 1991), 5-48.
4. Haller, M., Leithinger, D., Leitner, J., Seifried, T., Brandl, P., Zauner, J., Billingham, M. The Shared Design Space. In *SIGGRAPH 06: ACM SIGGRAPH 2006 Emerging technologies*, 2006, 29.
5. Haller, M., Leitner, J., Seifried, T., Wallace, J., Scott, S., Richter, C., Brandl, P., Gokceazade, A. The NiCE Discussion Room: Integrating Paper and Digital Media to Support Co-Located Group Meetings. In *Proc. CHI 2010*. ACM Press (2010), 609-618.
6. Hofmeester, K., Wixon, D. Using metaphors to create a natural user interface for microsoft surface. In *Ext. Abstracts CHI 2010*. ACM Press (2010), 4629-4644.
7. Hutchins, E., Hollan, J., Norman, D. Direct manipulation interfaces. *Human-Computer Interaction* 1, 4 (1985), 311-338.
8. Jacob, R.J.K., Girouard, A., Hirshfield, L.M., Horn, M.S., Shaer, O., Solovey, E.T., Zigelbaum, J. Reality-based interaction: a framework for post-WIMP interfaces. In *Proc. CHI 2008*. ACM Press (2008), 201-210.
9. Jetter, H.C., Gerken, J., Zöllner, M., Reiterer, H. Model-based Design and Prototyping of Interactive Spaces for Information Interaction. *Proc. HCSE 2010*, Springer (2010), 22-37.
10. Jetter, H.C., Gerken, J., Zöllner, M., Reiterer, H., Milic-Frayling, N. Materializing the Query with Facet-Streams – A Hybrid Surface for Collaborative Search on Tabletops. *Proc. CHI 2011*, ACM Press (2011).
11. Melchior, J., Grolaux, D., Vanderdonckt, J., Van Roy, P. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In *Proc. of EICS '09*. ACM Press (2009), 69-78.
12. Perlin, K., Fox, D. Pad: an alternative approach to the computer interface. In *Proc. SIGGRAPH 1993*. ACM Press (1993), 57-64.
13. Raskin, J. *The humane interface: new directions for designing interactive systems*. ACM Press, 2000.
14. Shaer, O., Jacob, R.J.K. A Specification Paradigm for the Design and Implementation of Tangible User Interfaces, *ACM TOCHI* 16, 4, 2009.
15. Smith, J. WPF Apps with the Model-View-ViewModel Design Pattern. In *MSDN Magazine*, 2009. <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>
16. Wigdor, D., Shen, C., Forlines, C., Balakrishnan, R. Table-centric interactive spaces for real-time collaboration. In *Proc. AVI 2006*. ACM Press (2006), 103-107.
17. ZOIL Framework. <http://zoil.codeplex.com/>.

A Model for DUI Development

Anders Fröberg
Department of computer
science
Linköping University
anders.froberg@liu.se

Henrik Eriksson
Department of computer
science
Linköping University
henrik.eriksson@liu.se

Erik Berglund
Department of computer
science
Linköping University
erik.berglund@liu.se

ABSTRACT

Distributed User Interfaces (DUIs) provide new degrees of freedom to the distribution of systems. This work presents a seamless way for developers to handle the event communication structure much in the same way as in traditional applications. Our framework Marve, is the externalization of our experience of developing several DUI systems. To evaluate our framework we developed a DUI system together with SAAB Aerosystem Human-Machine Interaction division. Using our approach to develop the sample application we show that the current model for development of UIs can be extended to incorporate support for DUI development.

Author Keywords

Distributed user interfaces, Frameworks, UAV

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: Miscellaneous

General Terms

Algorithms, Human Factors

INTRODUCTION

Distributed User Interfaces (DUIs) provide new degrees of freedom to the distribution of systems, in particular runtime reorganization and adaption to new environments with varying amounts of devices. DUIs introduce net-aware user interface components that can be moved across the network according to specification set up by programmers or by choices made by users in multi-device settings. With DUIs programmers get new means of enabling distribution and new ways of preparing systems for distribution even for cases where the amount of devices changes during runtime. It also enables programmers to specify how that distribution should be allowed to work. In essence, DUIs propose a natural continuation from single-device GUI systems to multi-device DUI systems. We propose the Marve DUI framework/platform, which focuses on extending current GUI-toolkits, such as Java SWING and .Net Forms, with DUI functionality.

Marve provides a high level support for transporting components between different devices at runtime. By hiding the network programming details, concerning transportation of user-interface components and events between components, Marve creates a layer of abstraction that makes user interface components movable between multiple devices at runtime. The framework also provides a model for describing how a set of components should be distributed for multiple settings of multi-device settings or how components can by users be distributed at runtime.

We stipulate that there is a common design ground for which we can build appropriate tools for further development of DUI applications. To support our claim we intend to outline a model for development of distributed user interfaces, our purpose is to present this through a software toolkit which utilizes our process. The initial goal with the framework is to create a mechanism for programmers to control the distribution, allowing the designing the distribution of ui-components to be managed in a easy fashion, much like the placement of ui-components are conducted in traditional application development. Our goal here is to provide an approach as similar as possible to the traditional way of ui-components placement, allowing for a smooth transition from GUI to DUI development. To test our framework we developed a DUI system together with SAAB Aerosystem Human-Machine Interaction division to both explore DUI in a command and control setting but also as a verification of our framework. The target application was a Unmanned aerial vehicle control station. It is an application domain that requires highly flexible system that can change the number of devices that the user interface is distributed over between runs or even during runtime. The number of stations may vary over time, certain tasks such as route planning, can be temporarily delegated to free devices or certain camera views can be distributed to devices located on the field. This highly flexible user interface requires a lot more from the developers.

Previous work

In multiple application domains we have in several projects [3, 1, 2, 7, 8] developed system with dui functionality. With each project we have by analyzing the software we developed gained a better understanding of the problems and challenges that arises when developing for a DUI environment. the software develop. The code, experience and the knowledge learned from the different projects, was the foundation for a design of a programming model for dui.

RELATED WORK

Pioneering work for distributed user interfaces is presented by Smith and Rodden [9] in a cooperative user interfaces when presenting Shared Object Layer (SOL). SOL allows the individual user's interfaces to be projected to devices from a common shared interface definition. This approach enables users to be presented with only the tools and function that they currently needs, or are allowed to use. Research closely related to our own research is reported in [5, 6] where they present a toolkit that enables developers to write system that allows component to be distributed over a set of devices. Within their toolkit the distribution of ui-components are managed in a peer-to-peer fashion without a central server and the transportation of component between devices are transparent to the developer. Using their approach developers are free to focus on building the user interface and do not need to concern themselves with how to send components over the network between devices, this is taken care of within the framework itself. The framework splits a component into two parts, a stationary part called the Widget proxy, which never leaves the devices it was created on, and a mobile part called the Widget Renderer, which can be moved between devices during runtime. The widget render is the visual part of a component, the part that presents the user with a user interface to interact with. In our approach developers are free to control what functionality is transported with the component and what functionality is kept on the device from which the component is send from. Another contribution with their system is the semi-automatic adaptation of ui-components by attaching what they call a adaptation variable to each component. Adaptation of a component is a mechanism for changing the representation of the component while keeping the the usability on a useful level. Their solution is implemented using the Mozart programming environment, Mozart is the primary implementation of the Qz programming language. Mozart aims to implement a network-transparent distributed programming model for objects. Our two different approaches differs here, we have chosen to base our solution on the Java programming language. Our choice of Java was based on the larger programming community around the language itself [11, 10], regardless of the less evolved network transparency in Java compared to Mozart.

A more theoretical work is presented by Demeure et al. in [4] where they presented a schema for classification of distributed user interfaces, with the aim to provide designers with a mechanism that can express distribution of interfaces over different devices.

THE MARVE FRAMEWORK

Marve is the externalization of our experience of developing DUI systems. Allowing the user interface be spread out over a set of devices or allowing ui-components flow between devices during runtime, requires that developers take in considerations a few core issues regarding the distribution.

- **Component singularity** - Component singularity refers to if a component is visible on only one device or mirrored on more.

- **Component callback coupling** - Components interact with each other and other system parts through the use of callback functionality. In a distributed environment callbacks can either be decoupled or decoupled from the component as the component are transferred between devices.
- **Component representation** - How a component is represented for transferred between devices.

Component singularity

In a distributed environment components can be visible on zero or more devices. This feature requires developers to define in what situations components should only be placed on a single device and when can a component be placed on several different devices. Component singularity is a mechanism for describing how components should be placed, as a placement relation between the devices and components.

- *Atomic presentation* refers to when a component can only be visible on a single devices at a time.
- *Mirrored presentation* refers when a component is placed on two or more devices at any given time.
- *Cloned presentation* refers when a component is placed on two or more devices at any given time, but each component is unique and not interconnected with the source component.

When components are being spread out over a set of devices an *Atomic presentation* is a move operation, where one component is moved between devices, where as a *Mirrored presentation* and *Cloned presentation* are a copy operation.

Component callback coupling

In a user interface when users interacts with a graphical component, such a clicking a button, a specific piece of code is assigned to executed when the event occurs. The piece of code that is to be executed as a reaction of that the button was pressed is referred to as the callback function for the button. A single component can have several callback functions assigned to same or different events generated by the component. As a component is transferred between one device to another these callback functions can either be coupled with the component or decoupled. When the callback functions are coupled, Client call-back, with the component they are send over to the receiving device together with the component itself. A decoupled callback function, Server call-back, is executed on the device the component is initialized on.

- *Server call-back* refers to when the call-back function is decoupled from the component and is not transferred together with the component between different devices. This is usually needed when several components use the same call-back function.
- *Client call-back* refers to when the callback function is coupled with the component and transferred together with the component between devices. The call back function is the always executed on the same devices as the component is currently residing on. This is usually for call-back

function that only gets called by a single component and the function don't have references to other components.

Developers using the framework can control the callback coupling by the use of different type Listeners. The default behavior for a callback function within the framework is Client call-back, but by changing the type of the call-back object the developer can control so the callback function remains on the server when the ui-component is transferred.

Component representation

Moving or copying a component from device to another forces the system to represent the component in away that can be received and interpreted by the receiving device. Representing a component for transfer is not as simple as just passing along the initial code that created the component. Most ui-components are dynamic and they can change over time as a application runs and sharing the initial state of the component the dynamic state of the component is lost. Take a text editor for an example, sharing/distributing the component that visualize the entered text by just it's initial state would loose all the text entered after the component was initialized. Components can be transferred in three general categories:

- Component based representation.
- Intermediate representation.
- Image representation.

A component representation is based on serializing components in a binary format often close in representation to the graphical toolkit the component currently is visualized by. Using a component base representation provides a easy way of developing, where the developer don't need to care about how to interpret and understand what type of components are send or received. The approach has the drawback that confines the sender and receiver to run the same user interface structure.

The intermediate representation of a component is textual format. This approach has the advantage that it don't need to be tied into a a specific framework but components can be visualized in different gui-frameworks.

The major drawback using this approach is that there is a need to formalize a common ground which works for the different devices connected to a specific application. A common ground need to include a set of ui-components, a set of events the components can react upon, and a means of connecting devices together for callback functions. Setting up the callback structure is a twofold approach, this in order to ensure that events and callbacks can be transferred to and from a distributed ui-component. The intermediate representation main benefit is that it allows components be visualized in different programming languages and different gui frameworks but it also provides limitations on how developers can use callback coupling. If there is no intermediate code that can be used on all platforms callback structures only can be used in a decoupled way. Since there is no guarantee that the

callback code will be able to run on the device that receives the ui-component.

Listing 1. A small and simple xml representation of a ui component

```
<type>Button</type>
<name>loginB</name>
<attributes>
  <label>Click to login</label>
</attributes>
<events>
  <outgoing>
    <onClick>192.168.0.2/buttonClick</onClick>
    <onMouseOver>192.168.0.3/mouseOver</onMouseOver>
  </outgoing>
  <incoming>
    <update attribute=label>newValue</update>
  </incoming>
</events>
</component>
```

The image representation is based on representing components as images. This approach allows almost all devices that can display a image receive components from other devices.

Today our framework only has limited support for image and intermediate transfer, the default means of transporting components is using the components base representation.

A SAMPLE APPLICATION

The purpose of the application is two fold, both to explore DUI as a user interface technology in a command and control domain but mainly to verify our model from the development of DUIs. The target application is a Unmanned Aerial Vehicles control-station for SAAB Aerosystems.

- *Unmanned Aerial Vehicle* Unmanned Aerial Vehicles (UAVs) are designed to in a cost effective way be a complement to traditional manned aircraft with regards to perform intelligence, surveillance, reconnaissance and weapon delivery. UAVs are controlled by pilot or a navigator on the ground using a control station, the aircraft itself seldom or never has a crew on-board.
- *UAV Control Station* The UAV control station (UCS) is responsible for controlling the UAV. The UCS sends flight instruction to the UAV, either using vectorization or using flight plans consisting of way-points. The UCS also can control and receive information from the sensors on-board the UAV.

The main task for the target application is the ability to plan and execute flight route plans. The main gui components of the application are: Map tool, flight planner, video player, land tool and a report tool, in Figure 1 you can see the entire application running on a single device. As a new device is connected the application either then adapt itself by place components on the new device, either through Atomic, Mirrored or Cloned representation or operators can manually transport components between devices. Figure 2 shows the application distributed over two devices and one component is mirrored on both devices wheres one component is moved to the new devices.

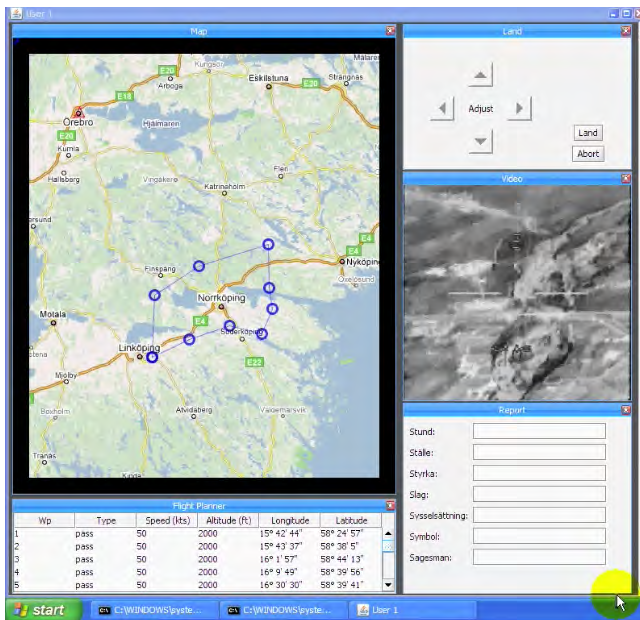


Figure 1. The application running with all components running on the same machine

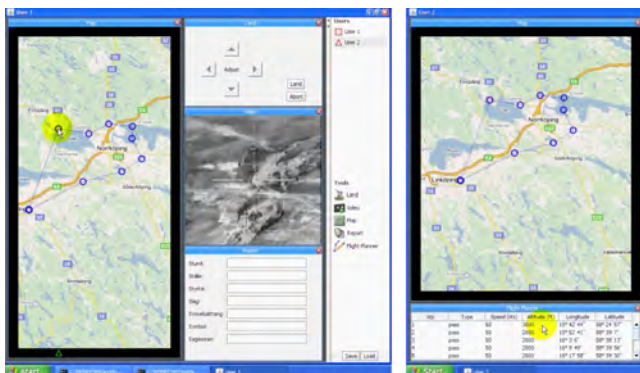


Figure 2. The application distributed over two devices, where the Map tool is mirrored on both devices where as the Flight planner is transferred atomically to the second device.

CONCLUSION

A significant technical challenge for DUIs and a contribution of this work is to support the traditional way for handling user interaction in a network independent model. This work presents a seamless way for developers to handle the event communication structure much in the same way as in traditional applications. In this work we have shown that this larger design space includes new design issues regarding component distribution time, singularity, representation, callback coupling, and constraints. Furthermore we have provided a set of pre-defined choices for each respective issue. We have shown how these choices can be utilized in a software framework. Although developing for a DUI is far more complex than a traditional UI, we believe that with the right tools and process that this complexity can be reduced to be close in complexity with the development of the traditional user interface. We have demonstrated that the current model for development of UIs can be extended to incorpo-

rate support for DUI development, which will enable for a smooth transition to a distributed world.

REFERENCES

1. M. Arvola and A. Larsson. Regulating prominence: A design pattern for co-located collaboration. In *Cooperative Systems Design: Scenario-Based Design of Collaborative Systems, Proceedings of COOP 04, 6th International Conference on the Design of Cooperative Systems*, 2004.
2. A. Berglund, E. Berglund, A. Larsson, and M. Bang. Paper remote: an augmented television guide and remote control. *Univers. Access Inf. Soc.*, 4:300–327, May 2006.
3. M. Bång, A. Larsson, E. Berglund, and H. Eriksson. Distributed user interfaces for clinical ubiquitous computing applications. *International Journal of Medical Informatics*, 74(7-8):545–551, 2005.
4. A. Demeure, G. Calvary, J.-S. Sottet, and J. Vanderdonkt. A reference model for distributed user interfaces. In *TAMODIA '05: Proceedings of the 4th international workshop on Task models and diagrams*, pages 79–86, New York, NY, USA, 2005. ACM Press.
5. D. Grolaux. *Transparent Migration and Adaptation in a Graphical User Interface Toolkit*. PhD thesis, Université catholique de Louvain, <http://www.isys.ucl.ac.be/bchi/publications/Ph.D.Theses/Grolaux-PhD2007.pdf>, 2007.
6. J. Melchior, D. Grolaux, J. Vanderdonckt, and P. Van Roy. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 69–78, New York, NY, USA, 2009. ACM.
7. M. Sjölund, A. Larsson, and E. Berglund. Smartphone views: Building multi-device distributed user interfaces. In *Mobile Human-Computer Interaction - Mobile HCI 2004*, 2004.
8. M. Sjölund, A. Larsson, and E. Berglund. The walk-away gui : Interface distribution to mobile devices. In *IASTED-HCI 2005, 2005*, pages 114–. ACTA Press, 2005.
9. G. Smith and T. Rodden. Sol: a shared object toolkit for cooperative interfaces. *Int. J. Hum.-Comput. Stud.*, 42(2):207–234, 1995.
10. Programming Language Popularity, 2011. <http://langpop.com>.
11. TIOBE Programming Community Index, 2011. <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

Multi-touch Collaborative DUI to create Mobile Services

Gabriel Sebastián, Pedro G. Villanueva, Ricardo Tesoriero

Computing System Department. University of Castilla-La Mancha
Campus Universitario de Albacete
(02071) Albacete, Spain
gabriel.sebastian@alu.uclm.es
[pedro.gonzalez, ricardo.tesoriero]@uclm.es

José A. Gallud

Miguel Hernández University
Avda. de la Universidad s/n,
(03202) Elche (Alicante), Spain
jgallud@umh.es

ABSTRACT

This paper describes a collaborative mobile application to support the creation of services that can be executed on mobile devices. This approach provides users with the ability to manipulate local and shared information at the same time in the same space. To support this feature, we have used mobile devices to handle local information related to a particular user, and a multi-touch surface to handle shared information for collaborative editing. The system is based on client-server architecture. On the one hand, the server is in charge of delivering the information among clients. On the other hand, we have defined two types of clients: the local client and the shared client. While the shared client allows users to interact with shared elements by the means of a multi-touch surface (i.e. the editor canvas, classes, aggregations, etc.), the local client allows users to interact with local elements, such as toolbar and elements' properties). Thus, this system overcomes the problem derived from editing procedures in the small displays on mobile devices using extra displays.

Author Keywords

HCI, Distributed User Interfaces, Coupled Displays, Multi-touch Surfaces, Workflows, Service creation.

ACM Classification Keywords

H.5.1 [Information interfaces & presentation]: Multimedia Information System; H.5.2 [Information interfaces & presentation]: User Interfaces; H.5.3 [Information interfaces & presentation]: Group and Organization Interfaces.

General Terms

Design, Experimentation.

INTRODUCTION

During the last few years there has been a great dissemination of mobile devices; the number of developments and contents for this kind of devices (applications, games, services, etc.) is increasing very

quickly. It is mainly due to the growth of the processing power and communication capabilities that these devices are experimenting lately. As consequence, they are not only capable of consuming services but of providing them too. These services are defined as "mobile services".

The creation of complex services is usually based on the composition of existing services that are logically related. These relationships are usually defined by the means of an editor that is capable of relating a set of already defined services and relate them accordingly. Thus, users are capable of creating services based on logical and physical conditions, such as the localization, the timing and so on.

The idea of sharing information is actually a fact. Social networks and Web 2.0 applications are actually being used from desktop computers, laptops or mobile devices [2, 4, 9]. Based on this vision, we focus on the UI that is employed to create and edit mobile services that can be shared and executed in mobile devices in a collaborative way.

To carry out this task, we employed a workflow based notation [10] to define mobile services in a collaborative way. Users share information that, due to the limited space of mobile device displays, is exposed on a shared canvas supported by a multi-touch surface. Thus, users are able to control local and shared information [1] through different interaction surfaces (mobile devices and the multi-touch surface).

From the DUI perspective, as result of this combination, we created a display ecosystem composed by Inch and Perch scale size displays under a few-to-few social interaction scenario [8].

From the collaborative perspective, according to the taxonomy exposed in [1], this system supports face-to-face collaboration among users through a DUI.

Therefore, we have organized the paper as follows. After the introduction section, we expose the motivation for this work and the main characteristics of the multi-touch surface. The hardware and software architecture of the system is exposed afterwards. Finally, we present the conclusions and future works.

MOTIVATION

Actually there are many applications to create and edit services; however most of them cannot be executed in mobile environments [11, 3]. Besides, none of these approaches allow the creation of mobile services in a collaborative way.

In [6] we developed a text-based UI employing natural language to enable users to create and edit mobile services in a reliable and efficient way. However, the sequential nature of the text based creation limits the interaction among users.

Consequently, we developed a graphical editor based on the workflow notation [10] that runs in a mobile device allowing users to create mobile services on their mobile devices in terms of components and relationships among them. Despite of the limited size of the device display, the edition capabilities were acceptable.

Thus, we tried to create a collaborative version of the workflow based editor. While designing the UI, we realized that we needed extra awareness information to support collaboration, which requires more space from the screen.

Besides, we also realized that the collaborative creation of mobile services was even better when people were using their mobile devices at the same time in the same space.

Multi-touch surfaces provide new ways of interaction where users are able to communicate with the environment using gestures. These surfaces recognize more than one gesture at the same time allowing users to interact each other in the same space at the same time providing face-to-face collaboration [1].

During a face-to-face collaboration, users deal with shared and private information. Therefore, information should be displayed in different displays because the multi-touch display is intrinsically a shared display.

This fact leads to the use of a distributed user interface to cope with the problem. Thus, while the multi-touch display exposes shared information, mobile devices, such as PDAs or mobile phones, expose local information to users.

Therefore, we proposed the expansion of the editing surface by the means of keeping toolbars and data input forms (local information) on the mobile device display, and translating the canvas (shared information) to a bigger display (the multi-touch surface).

THE MULTI-TOUCH SURFACE

Multi-touch surfaces are being used as part of the household and the office furniture (i.e. walls, tables, mirrors, floors). This section is focused on the multi-touch table developed in our research lab.

The multi-touch table we have developed shows a graphical UI on the surface that can be controlled through finger touches or by putting object over the surface.

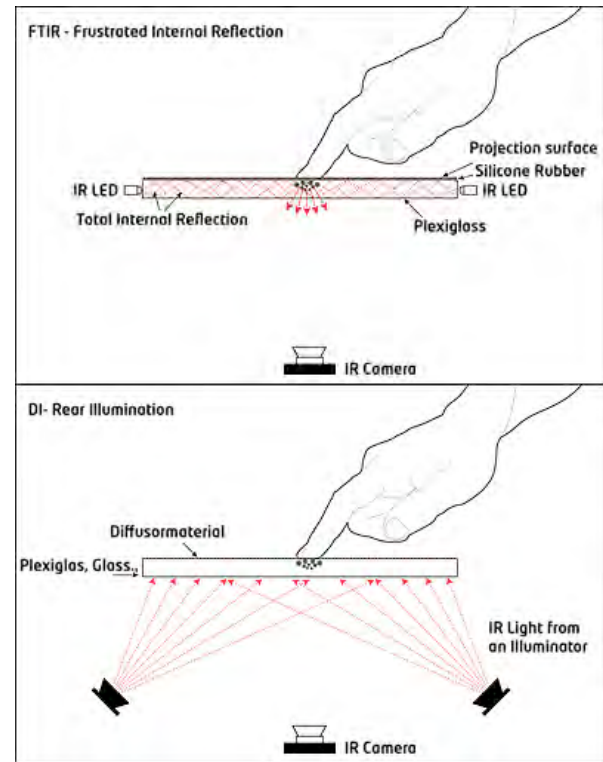


Figure 1. FTIR and DI technologies

This table is also capable of recognizing gestures from many users at the same time.

The two main technologies employed to build multi-touch tables are the Frustrated Total Internal Reflection (FTIR) and the Diffused Illumination (DI) (see Figure 1).

The FTIR technology relies on infrared light beams that are emitted from the side of an acrylic panel. When a finger touches the acrylic surface the light is “frustrated”, causing the light to scatter downwards, where it is picked up by an infrared camera.

The DI technology is based on the emission of infrared light that is emitted from below to the table surface. A diffuser is placed on the top or the bottom of the touch surface. When an object touches the surface, it reflects more light than the diffuser or the objects on the background.

The extra light is then captured by an infrared camera. Depending on the diffuser, this method can also detect hover objects or placed over the surface. Although the DI technology is more unstable regarding to external light conditions, it allows the use of tags to identify tangible objects. In fact, this is the reason we have adopted this approach to develop our multi-touch table.

The table is based on two synchronized independent systems:

1. The visualization system
2. The control system

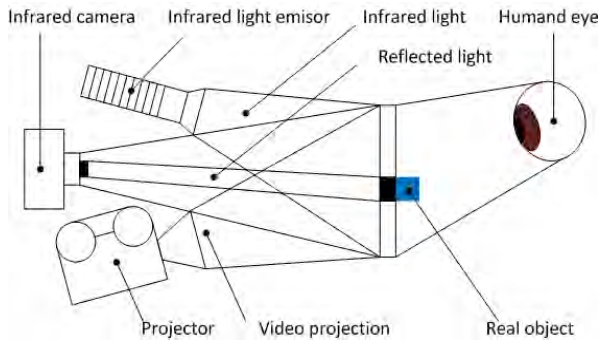


Figure 2. Hardware architecture

The visualization system projects the virtual objects that can be manipulated by users on the semi-transparent surface of the table.

The control system captures real object (i.e. fingers or objects on the surface) reflections by the means of an infrared camera (see Figure 2). Images are processed using some filters provided by the OpenCV library to detect the position of the real objects.

Therefore, the combination of the control and the visualization systems results on the view and controller of the multi-touch table following the Model-View-ViewModel (MVVM) [7].

THE SYSTEM ARCHITECTURE

Communication infrastructure

The system is based on client-server architecture. On the one hand, the server is in charge of delivering the information among clients. On the other hand, we have defined two types of clients: the local client and the shared client. While the shared client allows users to interact with shared elements by the means of a multi-touch surface (i.e. the editor canvas, classes, aggregations, etc.), the local client allows users to interact with local elements, such as toolbar and elements' properties).

The server exchanges XML messages with clients through a socket based application that uses a Wi-Fi network. It is a simple broker of communication among clients that is developed as a Java application. It delivers information in two ways: point-to-point and broadcasting. It is also in charge of the user authentication and authorization.

The local client was developed as a Flash Lite application that gets through the shared client through the server application. The information manipulated by the local client depends on the selected elements in the shared client. It defines a DUI using coupled displays.

The shared client was also developed as a Flash application that gets through the local client through the server application. This client runs in a multi-touch surface that allows users to select elements to be modified using the local client.

One advantage of using a DUI for this system is the capability of introducing text using the mobile device native writing system, instead of displaying a keyboard in the shared client.

The architecture of this server and communication between clients is displayed in Figure 3.

Functionality

This service editor for mobile environments is based on a **Functional View** that expresses the functionality for a new service using the Workflow paradigm (Figure 4). To build services, the user accesses a **Repository of Components** grouped by functionality (Figure 5). A Component is a functional unit or a capability, which defines properties, and input and output values. Thus, a service is composed by components that are connected each other through information flows defining input and output parameters.

To start a collaborative work session, each user connects to a Wi-Fi network. Then he/she performs a **Login** action on the server from his/her mobile device. As consequence, a pointer associated to the user is displayed on the multi-touch surface along with other pointers belonging to the rest of the users.

Once logged in, users are able to create **New** services, or **Open** an existing one to be edited. To edit a service, users can add components on a canvas displayed on the shared surface presenting the service components and the relationships among them, in a graphical way (Figure 4). Besides, users are able to **Edit** component properties (Figure 5), as well as **Move**, **Delete**, and **Link** components. In addition, users are able to use context-sensitive **Help**, and the **Save** the design.

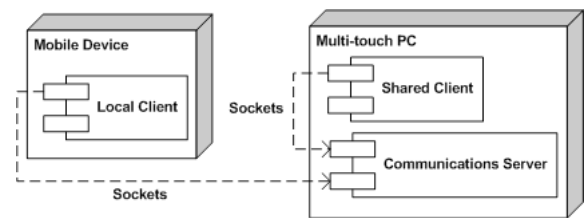


Figure 3. System communication architecture.



Figure 4. Workflow of a Service at the multi-touch table.



Figure 5. Component Repository and Properties Edit Form for a Component.

To interact with the shared working area displayed in the multi-touch table, each user uses the pointer that identifies him/her univocally. The user drags his/her pointer on the surface to select the Component he/she wants to manipulate (Figure 6). Once selected, this component is highlighted causing a change on the user device UI. Thus, the mobile client UI displays a context-sensitive toolbar or form to input data according to the operations that can be performed on the Component (i.e. *Edit, Move, Link, Delete*, etc.)

In addition users can **Publish** Services to be user by other users. Finally, users are also able to **Logout**.

CONCLUSION AND FUTURE WORK

This work is focused on the UI limitation of creating mobile services by non-expert users in a mobile environment. These limitations are originated in the characteristics of mobile devices, such as, small screen size and the different interaction forms used (keyboard, pointer, touch, etc.)

The proposed distributed interface system deals with three of these major challenges that currently affect mobile interfaces: (1) provides mobile interface extension expanding the area of work, (2) provides interaction with the various elements of editing (3) provides an innovative system of collaborative editing for mobile environments.

The approach of the server as a simple broker of communications among clients facilitates the reuse in new collaborative distributed applications in the future.

Afterwards, we plan to integrate the "creation layer" with the "execution layer" within the CENIT project "Tecnologías para prestar servicios en movilidad en el futuro universo inteligente" led by Telefónica I+D.

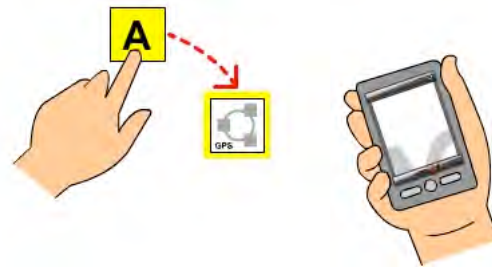


Figure 6. Selection technique based on cursor dragging.

ACKNOWLEDGMENTS

This research has been partially supported by the Spanish CDTI research project CENIT-2008-1019, the CICYT TIN2008-06596-C02-0 project and the regional projects with reference PPII10-0300-4174 and PII2C09-0185-1030.

REFERENCES

1. Ellis, C. A., Gibbs, S. and Rein, G. L. Groupware: Some Issues and Experiences. *Communications of ACM* 34(1), 1991, 39-58.
2. Facebook. <http://facebook.com>.
3. Google App Engine. <http://code.google.com/intl/es-ES/appengine/>
4. Google Docs. <https://docs.google.com/>.
5. Johansen, R., "Groupware: Computer support for business teams," *New York: The Free Press*, 1988.
6. Sebastián, G., Gallud, J. A. and Tesoriero, R. A Proposal for an Interface for Service Creation in Mobile Devices Based on Natural Written Language. *Proc. of 5th International Multi-conference on Computing in the Global Information Technology*, 2010, 232-237.
7. Sorensen, E., Mihailesc, M. I. Model-View-ViewModel (MVVM) Design Pattern using Windows Presentation Foundation (WPF) Technology. *MegaByte Journal*. 2010. ISSN: 1841-7361.
8. Terrenghi, L., Quigley, A. and Dix, A. A taxonomy for and analysis of multi-person-display ecosystems. *Personal and Ubiquitous Computing* 13, (2009), 583-598.
9. Twitter. <http://twitter.com>.
10. Workflow Management Coalition. <http://www.wfmc.org>.
11. Yahoo Pipes. <http://pipes.yahoo.com/pipes/>

Using Multi-touch Technologies to Perform Collaborative Map Exploration

Pedro G. Villanueva, Ricardo Tesoriero, Gabriel Sebastián
Computer System Department. University of Castilla-La Mancha
Campus Universitario de Albacete
(02071) Albacete, Spain
[pedro.gonzalez, ricardo.tesoriero]@uclm.es
gabriel.sebastian@alu.uclm.es

José A. Gallud
Miguel Hernández University
Avda de la Universidad s/n,
(03202) Elche (Alicante), Spain
jgallud@umh.es

ABSTRACT

In this paper we present the Collaborative Map Explorer system, which provides a new learning artifact that allows students to explore different layers of information on a map related to a concrete geographic location using distributed user interfaces in the same space at the same time. Students work with shared information through a multi-touch table where they are able to see different views of a region of the map. This region can be dragged across the map as a floating window where students can see different layers of the map. The layer of map that is shown within the window is selected through the local user interface that runs on the mobile device of the student. The use of a distributed user interface provides students with the ability to manage local and shared information at the same time in the same space.

Author Keywords

HCI, Distributed User Interfaces, Coupled Displays, Multi-touch Surfaces, Geographic Information Systems.

ACM Classification Keywords

H.5.1 [Information interfaces & presentation]: Multimedia Information System; H.5.2 [Information interfaces & presentation]: User Interfaces; H.5.3 [Information interfaces & presentation]: Group and Organization Interfaces.

General Terms

Design, Experimentation.

INTRODUCTION

Multi-touch surfaces provide new ways of interaction where users are able to communicate with the environment using gestures.

These surfaces recognize more than one gesture at the same time allowing users to interact each other in the same space at the same time providing face-to-face collaboration [3].

During a face-to-face collaboration, users deal with shared and local information [1]. Therefore, information should be exposed in different displays because the multi-touch display is intrinsically a shared display.

This fact leads to the use of a distributed user interface (DUI) to cope with the problem. Thus, while the multi-touch display exposes shared information, mobile devices, such as personal digital assistants (PDAs) or mobile phones, expose local information to users.

According to the taxonomy described in [7], the combination of personal displays with shared displays generates a display ecosystem composed by Inch and Perch scale size displays that define a few-few social interaction relationship among users.

Maps have been used to expose geographical information for centuries. Geographical Information Systems (GIS) provide users with different layers of information related to geographical locations. Maps have been used as learning artifacts for years; they provide students with a good abstraction of the real world to locate information, as well as, to explore geographical zones.

Being the Cooperative Supported Collaborative Learning (CSCCL) the root of the Computer Supported Collaborative Work (CSCW), there are lots of eLearning applications that use the Groupware technology to improve the learning process of students.

The popularization of mobile technologies had a great impact on the learning process being the supporting tool of the informal learning [5].

The collaboration among students to carry out learning activities, such as those involved in the problem based learning, is becoming a key issue during the learning process.

Therefore, we propose the use of multi-touch surfaces as a tool for students to encourage the collaborative learning among them in informal learning scenarios [5].

The goal of this tool is the exploration of maps using a multi touch surface and a mobile device to retrieve information in a collaborative learning scenario.

This paper starts with a brief review of some work that is related to this proposal. Then, we expose the goals and the functionalities of the system to explore maps, as well as the system architecture. Finally, we expose conclusions and future work.

RELATED WORK

In this section, we expose the most relevant work related to the exploration of maps using mobile devices.

One of the most promising technologies that enabled this kind of applications is the RFID technology. One example of the application of this technology is exposed in [4] where the RFID technology was employed to provide metro users with information about transportation and city touristic points. This approach employed RFID tags to identify “hot-spots” in the map. Thus, when users approach a PDA equipped with a RFID reader to a tag identifying a location in the map, information related to the map location is displayed on the PDA screen.

Although this approach has some advantages, such as the reliability, and the low cost of the map deployment (RFID tags are really cheap); readers are not wide spread among mobile devices and they are expensive. Besides, the system does not allow users to introduce information related to a map location.

Information exchange among users is one of the most relevant characteristic of collaborative systems. The Interactive Learning Panels [8] combines the use of the RFID technology with didactic maps to improve the learning experience and collaboration between teachers and students. The system also employs RFID tags to identify “hot-spots”; however they are deployed as a matrix of tags. Thus, students are not limited to explore specifics points in the map, but the whole map. As result of this tag deployment student perceive a continuous flow of information from the map. This approach also provides students to introduce information related to a specific location in the map that can be retrieved by the professor.

However, the map resolution to extract or introduce information from or into the map using Interactive Learning Panels is coarse grained, and it is limited to the RFID tag size, and the RFID reader range. Besides, the view of the map is the same for all students, because it is a traditional map that is signaled with RFID tags on the back.

From the multi-touch perspective, an interesting approach is described in [9] where users are able to retrieve information at different regions of the map. The information to be displayed is selected through different menus that are showed into the shared surface. The main advantage of this approach regarding to the RFID-based approaches is the flexibility of displaying the information directly to the shared surface replacing the mobile device as the information retriever. Although this approach seems to be more practical and flexible, it has two major problems.

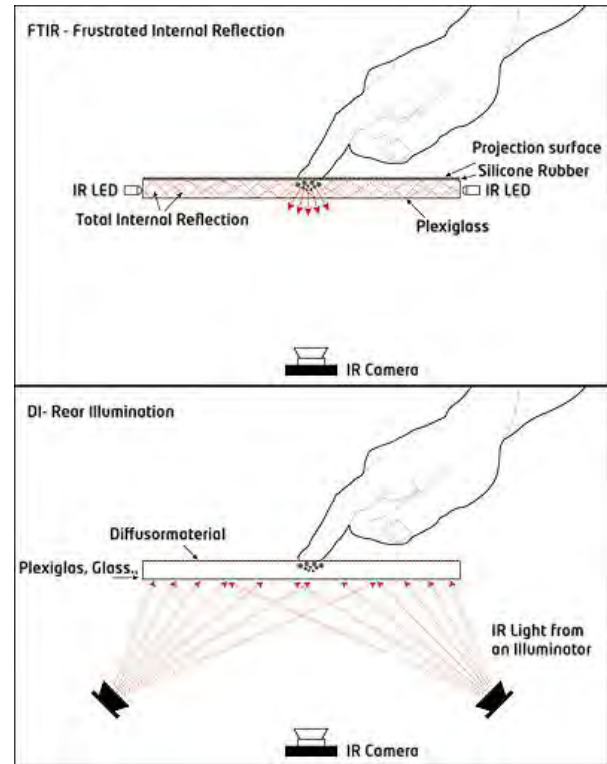


Figure 1. FTIR and DI technologies

The first one is related to the collaboration. The use of a shared display as the unique way to manipulate information deprives users from the ability to manipulate local information. For instance, if students have to achieve individual and group goals, individual goals should be kept for the student alone.

The second one is related to the Human-computer interaction (HCI) field focusing on the “information pollution” problem. The use of a single shared display by many users at the same time should minimize the use of the display for local information. For instance, the visualization of menus and options that are part of a particular user choice should be avoided.

Thus, our approach combines the advantages of the RFID-based approaches that employ personal devices, such as PDAs or mobile phones, to expose local information; with the flexibility of using a multi-touch surface to expose and manipulate shared information.

THE MULTI-TOUCH SURFACE

Multi-touch surfaces are being used as part of the household and the office furniture (i.e. walls, tables, mirrors, floors). This section is focused on the multi-touch table developed in our research lab.

The multi-touch table shows a graphical UI on the surface that can be controlled through gestures made by fingers or physical objects over the surface. This table is also capable of recognizing gestures from many users at the same time.

The two main technologies employed to build multi-touch tables are the Frustrated Total Internal Reflection (FTIR) and the Diffused Illumination (DI) [1].

The FTIR technology relies on infrared light beams that are emitted from the side of an acrylic panel. When a finger touches the acrylic surface the light is “frustrated”, causing the light to scatter downwards, where it is picked up by an infrared camera (see Figure 1).

The DI technology is based on the emission of infrared light that is emitted from the bottom of the table to the table surface. A diffuser is placed on the top or the bottom of the touch surface. When an object touches the surface, it reflects more light than the diffuser or the objects on the background. The extra light is then captured by an infrared camera. Depending on the diffuser, this method can also detect hover objects or objects placed over the surface (see Figure 1).

Although the DI technology is more unstable regarding to external light conditions, it allows the use of tags to identify tangible objects. In fact, this is the reason why we have adopted this approach to develop our multi-touch table.

The table is based on two synchronized independent systems: a) the visualization system and b) the control system. The visualization system projects the virtual objects that can be manipulated by users on the semi-transparent surface of the table. The control system captures real object reflections (i.e. fingers or objects on the surface) by the means of an infrared camera (see Figure 2). Images are processed using some filters provided by the OpenCV library to detect the position of the real objects.

Therefore, the combination of the control and the visualization systems results on the view and controller of the multi-touch table following the Model-View-ViewModel pattern (MVVM) [6].

THE COLLABORATIVE MAP EXPLORER

Our proposal presents a new learning artifact that allows students to explore, in collaborative way, different layers of information related to a concrete geographic location using coupled displays.

On the one hand, the teacher suggests tasks to students in order to work with different layers of views in the geographical locations of the map to get familiar with them. On the other hand, students perform the tasks proposed by the teacher using the Multi-touch Collaborative Map Explorer. To carry out the exploration, each student employs a PDA where he/she chooses the kind of

information to be exposed on map projected on the multi-touch surface.

When students log into the system, a floating window is associated to the student over the map. The window can be moved over the map by the student, simply by dragging his/her fingers to explore different locations on the map. Students can also zoom in/out the zone by peaching

In the Figure 3, we describe a scenario where two students, Maria and Luis, are using the Collaborative Map Explorer system. While Maria is exploring the political layer of the Extremadura community associated to her window; Luis situated his window over the Valencia community to explore the fauna information. Both students can move their associated windows to get information of other locations over the map.

THE SYSTEM ARCHITECTURE

The system is based on a client-server architecture using a Wi-Fi network as the communication medium.

The server application consists of two components: a) the communication facility that allows the communication among the clients; b) the visualization component that shows the reference map (background) and all student windows on the multi-touch surface. Besides, the server is responsible for the user authentication and authorization, too. The server application was developed using Microsoft Windows Presentation Foundation framework.

The client application allows students to connect with the server and select the information layer to be showed on the student windows over the map. The client application was developed using the Microsoft .NET Compact Framework technology 3.5.

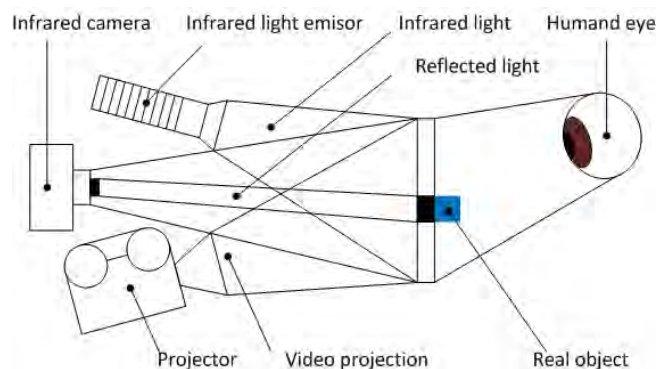


Figure 2. Hardware architecture



Figure 3. Specific scenario of Collaborative Map Explorer

CONCLUSION AND FUTURE WORK

In this paper, we have presented a system that allows students to explore different layers of information related to a concrete geographical location using coupled displays. The multi-touch table allows the collaborative learning among student.

The use of distributed user interfaces (DUIs) provides students with a way to work with both, local and shared information in the same space at the same time. While the local information is handled by the mobile device of the student, the multi-touch surface handles the shared information. Note that the information handled by the client application affects the shared information visible to all students (viewer state).

The Collaborative Map Explorer employs direct manipulation through student fingers making gestures such as dragging on the multi-touch surface to explore different locations or to zoom in/out the viewer zone.

Regarding the Interactive Learning Panels, presented in the related work section, the Collaborative Map Explorer improves two important aspects: the shared information is dynamic and no extra hardware is required in the PDA. Regarding the GlogalData system, the Collaborative Map Explorer improves the student collaborative work by manipulating the local and shared information in separated displays to avoid toolboxes on the shared screen.

On future versions of the system, we will include more actions that students can perform on Geographical Information Systems. We will also evaluate the Collaborative Map Explorer with real students.

ACKNOWLEDGMENTS

This research has been partially supported by the Spanish CDTI research project CENIT-2008-1019, the CICYT TIN2008-06596-C02-0 project and the regional projects with reference PPII10-0300-4174 and PII2C09-0185-1030.

REFERENCES

1. Chang, R., Wang, F., You, P. "A Survey on the Development of Multi-touch Technology," Wearable Computing Systems, Asia-Pacific Conference on, pp. 363-366, 2010 Asia-Pacific Conference on Wearable Computing Systems, 2010
2. Ellis, C. A., Gibbs, S. and Rein, G. L. Groupware: Some Issues and Experiences. Communications of ACM 34(1), 1991, 39-58.
3. Johansen, R., "Groupware: Computer support for business teams," New York: The Free Press, 1988.
4. Reilly, D., Rodgers, M., Argue, R., Nunes, M. and Inkpen, K. Marked-up maps: combining paper maps and electronic information resources. *Personal and Ubiquitous Computing* 10, 4 (2006), 215-226.
5. Scanlon, E., Jones, A. and Waycott, J. Mobile technologies: prospects for their use in learning in informal science settings. *Journal of Interactive Media in Education* (25), 1-17.
6. Sorensen, E., Mihailesc, M. I. Model-View-ViewModel (MVVM) Design Pattern using Windows Presentation Foundation (WPF) Technology. *MegaByte Journal*. 2010. ISSN: 1841-7361
7. Terrenghi, L., Quigley, A. and Dix, A. A taxonomy for and analysis of multi-person-display ecosystems. *Personal and Ubiquitous Computing* 13, (2009), 583-598.
8. Tesoriero, R., Fardoun, H. M., Gallud, J. A., Lozano, M. D. and Penichet, V. M. R. Interactive Learning Panels. *Human-Computer Interaction. Interacting in Various Application Domains*. LNCS 5613 (2009), 236-245.
9. Zadow, U., Daiber, F., Schöning, J., and Krüger, A. GlobalData: Multi-User Interaction with Geographic Information Systems on Interactive Surfaces. *ACM Interactive Tabletops and Surfaces 2010*. Saarbrücken, Germany. ISBN: 978-1-4503-0399-6.

Activity-Based Computing – Metaphors and Technologies for Distributed User Interfaces

Jakob Bardram, Afsaneh Doryab, Sofiane Gueddana
IT University of Copenhagen
Rued Langgaardsvej 7, DK-2300 Copenhagen, Denmark
{bardram, adoryab, sgue}@itu.dk

ABSTRACT

A recurrent challenge in DUI research is managing a task spanning multiple displays and devices. In this position paper, we propose to use the Activity-Based Computing (ABC) paradigm as an approach to form, manage, and use interactive workspaces. This approach applies ABC to distributed Multi-Display Environments (dMDE), i.e., environments with multiple devices and displays distributed across several spaces inside a large building. We describe ABC principles, current implementation and our ongoing work on activity-based dMDEs.

Author Keywords

distributed interfaces, healthcare, collaboration, context awareness

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation.

General Terms

Design, Human Factors.

INTRODUCTION

An emerging and significant research question within Human-Computer Interaction is related to interaction techniques and use of multiple interconnected displays and devices inside an interactive workspace – the so-called Multi-Display Environments (MDEs). A number of technologies, tools, systems, and environments are being investigated and evaluated. Research on ‘Smart Spaces’, such as iLand [9], Gaia [7], Interactive Workspaces [8], and IMPROMPTU [6] have provided initial insight into the underlying infrastructures and interaction technologies for such reactive environments, and research have shown how users may utilize new kinds of interaction techniques for using multiple co-located and shared displays and devices.

At the CHI 2006 Workshop on ‘Information Visualization and Interaction Techniques for Collaboration Across Mul-

iple Displays’ [10], which gathered most of the researchers addressing these issues, a recurrent theme was the challenge associated with managing a task spanning several displays and devices. It is a core challenge to manage large amount of devices, applications, files, documents, etc., which are associated with a specific task.

In this paper, we propose to use the Activity-Based Computing (ABC) paradigm as an approach to form, manage, and use interactive workspaces. In Activity-Based Computing, the user’s activities are modeled and managed by the technology. Modeling means that the technology has a persistent model of the type of activity, the intention, relevant resources, and participants of the activity. Managing means that the technology is able to store, distribute, adapt, and modify activity models according to their use and context. This approach applies ABC to what we call a ‘distributed Multi-Display Environment’ (dMDE), i.e. an MDE that is distributed across several physical spaces inside a large building [2].

The paper presents technologies for supporting activity-based dMDEs, where the activity of the user becomes explicit in fixed or mobile publicly available displays. This is used to adapt the public displays according to their context in a way that the most relevant activities are easy to access and use. The activity is then used to manage the complex set of applications, documents, files, and resources which are associated with performing a task. These technologies consist of two main parts; an underlying infrastructure for activity management, and a user interface software technology designed to be used on distributed public displays.

The goal of this research is to help users to use available wall-based and mobile public displays while moving in and out of places which have these devices available. Much of this work on supporting nomadic use of public displays is motivated in designing computer support for clinical work inside a hospital – work which is characterized as being highly nomadic, collaborative, and time-critical [5].

In this paper, we describe the metaphor offered by activity-based computing to organize distributed user interfaces at the scale of distributed multi-display environments. This position paper outlines the overall approach to activity-based computing for distributed multi-displays environments, and presents our current implementation and ongoing work. Our core research hypothesis is that the activity-based computing

approach helps manage the complexity of distributed user interfaces. This work takes its outset in the complex work environment of a hospital, but we have sound reasons to speculate that the activity-based computing approach for using public displays may be applicable in other work settings as well. We hope to be able to discuss this at the workshop.

CONCEPTUAL BACKGROUND

In this section, we describe distributed multi-Display environments, a concept built from observations of hospital work. We then present Activity-Based Computing and illustrate its principles.

Distributed Multi-Display Environments

This research is based on extensive observations of clinical work in large hospitals [1]. In many ways, the work of hospital clinicians is very different from that of information workers. First of all, work in hospitals is nomadic, i.e. clinicians constantly move around inside the hospital visiting different patients, departments, wards, conference rooms, and colleagues.

Secondly, clinical work is intensively collaborative; patient care happens in close collaboration between a team of highly specialized clinicians who constantly need to align, coordinate, and articulate their respective part of the overall activity of treating the patient. This collaboration happens remotely and co-located, and typically pivots around shared medical information available in medical records, medical charts, radiology images, etc.

Thirdly, publicly available ‘displays’ play a core role in the execution and coordination of work in hospitals. Examples include large whiteboards listing operations, patients, beds, and other critical resources; medical records; notebooks; and small notes. Some of these artifacts are fixed to the wall and public in nature, while others are carried around and are more personal.

Based on observations like these, it becomes relevant to investigate concepts and technologies that would allow clinicians to use multiple public and personal displays in their work. A distributed multi-display environment (dMDE) is an environment that supports multiple collaborating users through a technical and conceptual integration of heterogeneous personal and public devices, supporting physical distribution of the display environment all over a large building. Figure 1 illustrates a configuration of dMDE where displays are distributed across a hospital.

Activity-Based Computing

To form, manage, and use distributed user interface in distributed multi-display environments, we propose Activity-Based Computing (ABC) approach, defined around the following essential principles [3]:

Activity-Centered – A ‘Computational Activity’ collects, in a coherent set, a range of services and data needed to support a user carrying out some kind of (work) activity.

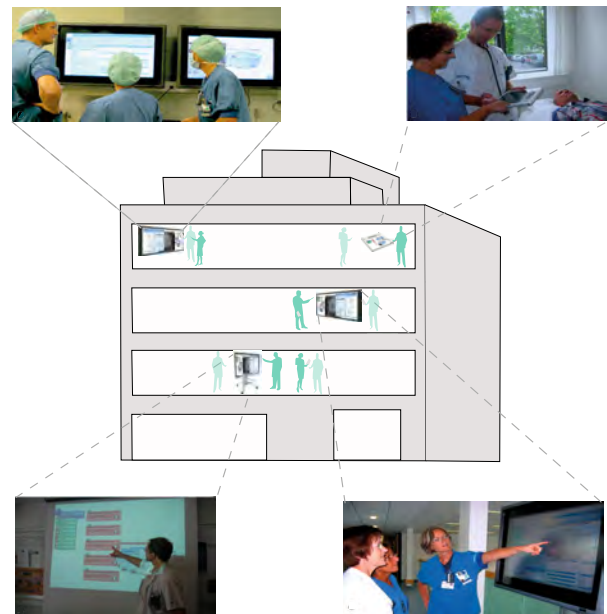


Figure 1. The figure shows a dMDE in a hospital.

Activity Suspend and Resume – A user participates in several activities and he or she can alternate between these by suspending one activity and resuming another. Resuming an activity will bring forth all the services and data which are part of the user’s activity.

Activity Roaming – An activity is stored in an infrastructure and can be distributed across a network. Hence, an activity can be suspended at one display and resumed on another in a different place.

Activity Adaptation – An activity adapts to the resources available on the device on which it is resumed. Such resources are e.g. the network bandwidth, CPU, or display on a given device.

Activity Sharing – An activity is shared among collaborating users. It has a list of participants who can access and manipulate the activity. Consequently, all participants of an activity can resume it and continue the work of another user and they can work concurrently on an activity.

Context-awareness – An activity is context-aware, i.e. it is able to adapt and adjust itself according to its usage context. Context-awareness can be used for adapting the user interface according to the user’s current work situation.

In the rest of this paper we will present the implementation of these principles and how they support distributed user interfaces in a distributed multi-display environment.

IMPLEMENTATION

Current implementation of ABC relies on three components: an activity-centered data model, a distributed storage infrastructure and a user interface. This implementation is designed for distributed multi-display environment and runs on various hardware devices distributed across a network, e.g. mo-



Figure 2. In the *activity view*, activities are represented by floating panels including graphic and text description. Related actions are represented by icons on that panel.

bile, desktop, table or wall-based display. The system can also be coupled with context sensors such as location trackers and keep track of physical locations of the users.

Activity-Centered

In ABC, activities are the center of the data model. They are composed of logical bundles, called *actions*. Each action contains a collection of *resources*, e.g. files, web pages, applications, medical files record, radiography, etc. The user interaction with the interface starts on the activity level, where activities and related actions are presented in *activity view* (Fig. 2). On this view, the user can see the status of activities and corresponding actions, e.g., resumable or done. Resources of an action can be displayed in *action view*, where the user can browse the resources of an action and start interacting with them (Fig. 3, left).

Activity Suspend and Resume

The user can start one or several activities and switch between them by resuming one activity and suspending another. When activities are resumed or suspended, the information about the state of activities is synchronized across the network. While resuming an action, the display switches to the *action view*, which shows all resources associated with this action. Each resource can be individually resumed or suspended. When the user suspends the action, the display switches back to the *activity view*.

Activity Roaming

The user can interact with an activity and suspend it on one display. When the activity is resumed on another device in a different place, all resources are restored. The principle of activity roaming is supported by storing activity data model in a distributed storage system. Current implementation employs a system called Aexo [4], that maintains a hierarchical mapping structure which holds all persistent data in the system including users, activities, (links to) resources, and contextual information such as location.

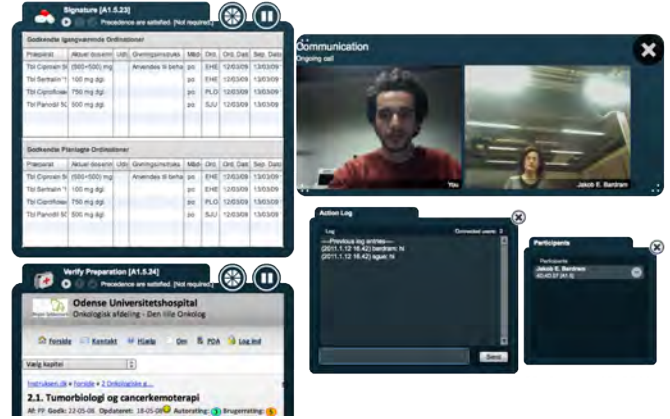


Figure 3. On the left, *Action view* presents the resources in panels encapsulating different type of interactive remote content, e.g. web pages, tables or graphs, etc. On the right, *Action view* collaborative windows include list of participants, chat window and video-conference window.

Activity Adaptation

Our current approach to adaptation is to design a generic technology that can run on any platforms in the same way. The user interface is designed for interaction on different displays with input specificities, e.g. mouse, touchpad or touch-based input device. This flexibility allows to interact with the activities in several alternative ways. For example, in order to move, suspend or resume the activities or resources on a touch display, the user can either use the mouse buttons or his fingers.

Activity Sharing

The ABC model offers a built-in support for collaboration, as activities are shared among participants. All participants of an activity can resume an activity and continue the work of another user or work concurrently. The system also supports activity sharing in desktop conferencing fashion : users can share the layout of the resources, start multi-site video-conference sessions, or post text notes (Fig. 3). Text notes are posted by participants in an activity through a chat window.

The layout of resources can also be shared among participants' displays, reinforcing WISIWYS¹ basis for collaboration : whenever a user moves or resizes a resource window, it is moved and resized in other users' displays as well. The interface also allows to establish audio and video links with remote participants and hold a multi-point video conference session. The video conference link is established depending on the location of users and the resumed activity.

Context-awareness

In ABC, context events, such as location of the users, are linked to a relevant activity. For example, the presence of the physician near the patient may trigger a suggestion to resume the computational activity associated with this patient's treatment which will bring up relevant documents and

¹Acronym for "What I See Is What You See", used for groupware that guarantee that users see the same thing at all times.

information for that patient on the available public display, e.g. the physician's portable device. Current implementation uses location trackers based on active RFID tags. The user's location is used for several purposes which include updating the list of participants in an activity and starting video-conference sessions between users resuming an activity.

ONGOING WORK

Our work on context-aware adaptation for DUIs has so far included considering the physical presence of participants and their locations. The more participants of the same activity are present in the same location, the more relevant the activity becomes. However, in many situations, the activities are related to the user's current task rather than the location. For example, in medical environments, the staff are usually engaged in several patient cases during a shift and there is a huge amount of documents and information resources associated with each patient case ranging from medicine charts, tens of x-rays, test results, nurse notes, etc. Automatically bringing all these up in front of the physician is clearly unnecessary just because he happens to be approaching a patient. Hence, we have extended the context of the system to include the user's physical tasks. By detecting and recognizing these tasks in the physical environment, the activities and associated resources can be adapted to current situation more precisely. It also enhances the awareness of the system in terms of information retrieval in distributed interfaces.

Currently, we are collecting data in order to train and build a learning model which can be used to detect and recognize the physical tasks of clinicians in hospitals. In distributed settings such as medical wards, sensing the location of tasks can be a major feature to reveal the activity if the place is set for a specific purpose, e.g., the medicine room. However, relying on location tracking might cause confusion in more general purpose places like hallways. Hence, more inputs, such as sensing tools and devices are required to detect the activities more correctly.

Unlike the activities in medical wards which are highly distributed into different places, some other clinical tasks are located in one place. For example, the whole surgical procedure takes place in the operating room. In this case, the activity recognizer should take into account other factors especially using tools and devices by clinicians.

REFERENCES

1. Bardram, J. E., and Bossen, C. Mobility Work - The Spatial Dimension of Collaboration at a Hospital. *Computer Supported Cooperative Work*. 14, 2 (2005), 131–160.
2. Bardram, J. E., Bunde-Pedersen, J., Doryab, A., and Sørensen, S. Clinical surfaces — activity-based computing for distributed multi-display environments in hospitals. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II*, INTERACT '09, Springer-Verlag (Berlin, Heidelberg, 2009), 704–717.
3. Bardram, J. E., Bunde-Pedersen, J., and Soegaard, M. Support for activity-based computing in a personal computing operating system. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, ACM Press (New York, NY, USA, 2006), 211–220.
4. Bardram, J. E., Bunde-pedersen, J., and Soegaard, M. Support for activity-based computing in a personal computing operating system. In *Proc. of SIGCHI*, Press (2006), 211–220.
5. Bardram, J. E., and Christensen, H. B. Pervasive computing support for hospitals: An overview of the activity-based computing project. *IEEE Pervasive Computing* 6, 1 (2007), 44–51.
6. Biehl, J. T., Baker, W. T., Bailey, B. P., Tan, D. S., Inkpen, K. M., and Czerwinski, M. Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ACM (New York, NY, USA, 2008), 939–948.
7. Hess, C. K., Román, M., and Campbell, R. H. Building applications for ubiquitous computing environments. In *Proceedings of the First International Conference on Pervasive Computing*, Pervasive '02, Springer-Verlag (London, UK, 2002), 16–29.
8. Johanson, B., Fox, A., and Winograd, T. The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing* 1, 2 (2002), 67–74.
9. Streitz, N. A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., and Steinmetz, R. i-land: an interactive landscape for creativity and innovation. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, ACM (New York, NY, USA, 1999), 120–127.
10. Terrenghi, L., May, R., Baudisch, P., MacKay, W., Paternò, F., Thomas, J., and Billingham, M. Information visualization and interaction techniques for collaboration across multiple displays. In *CHI '06 extended abstracts on Human factors in computing systems*, CHI '06, ACM (New York, NY, USA, 2006), 1643–1646.

Distributed interactive surfaces using tangible and virtual objects

Sophie Lepreux, Sébastien Kubicki,
Christophe Kolski

Université Lille Nord de France, F-59000 Lille,
France,
UVHC, LAMIH, F-59313, Valenciennes, France
CNRS, FRE 3304, F-59313 Valenciennes, France.
{firstname.name}@univ-valenciennes.fr

Jean Caelen

Multicom, Laboratoire d'Informatique de
Grenoble (LIG),
UMR 5217, BP53 F-38041 Grenoble cedex 9,
France.
{firstname.name}@imag.fr

ABSTRACT

After having outlined the uses of new technologies such as Smartphones, touchscreen tablets and laptops, in this paper we present the *TangiSense* interactive table, equipped with RFID technology tagged on tangible objects, as new paradigm of interaction for ambient intelligence. We propose two strategies for the distribution of interfaces integrated on the table in the framework of collective work. A case study centered on crisis management units, i.e. a collaborative situation, with multiple actors who are geographically separate, makes it possible to illustrate possible distributed uses and the *TangiSense*'s capacities. To finish, the article presents the directions under consideration for our future research.

Author Keywords

Interactive table, tangible and virtual objects, Distributed User Interfaces.

ACM Classification Keywords

H.5.3 Group and Organization Interfaces: Collaborative computing, C.2.4 Distributed Systems.

General Terms

Theory.

INTRODUCTION

New interactive surfaces such as touchscreen tablets are currently being studied a great deal. These surfaces allow interactions based on tactile technology. The principal uses of the tablets are: Internet access, consultation of books, and visualization of films or photographs. Smartphones, being smaller and thus more mobile, and also having access to the Internet albeit with less ease of navigation due to the size of the screen, are used when there is a need for mobility. As for laptop computers, they are increasingly

small (e.g. netbooks), but they remain the least mobile and thus are used mainly in fixed situations. They allow a wider range of activities than the tablets, such as the easy use of an editor for text documents, or running applications which necessitate more resources. We propose in this paper to increase another technology to this range of products: the Interactive Table (Figure 1).



Figure 1. The *TangiSense* interactive table with the Recognition and learning of colors application

Interactive tables have existed for a few years now [9]. They are similar to the touchscreen tablet, but larger, being more the size of a coffee table for example, around 1m square. These tables are mainly based on tactile or vision technology and allow the same uses as the tablets; they thus handle mainly virtual objects. In this article, we propose another type of interactive table which uses RFID technology in order to interact, not only with the fingers on virtual objects but directly with tangible objects RFID tagged. This table, called *TangiSense*, detects objects equipped with RFID tags when they are placed on it [4]. The use of RFID makes it possible firstly to detect objects present on the surface of the table, secondly to identify them thanks to one or more tags stuck underneath the object (because RFID tags are unique), and finally to store information directly in these objects or to superimpose them. It is thus possible to work with a set of objects on a table, to store data in these objects (for example their last position, or their possible owner) in order to subsequently be able to re-use them on another table at another moment

and with your own embedded information (example: the last state of a game of chess). The interaction is done completely by these objects and this can influence virtual objects (which cannot be directly manipulated). It provides different uses of the other technologies presented before because it uses the RFID characteristic as well as the advantage of tangible interaction. For example, this table was used in a school class with children to learn and recognize colors (Cf. Figure 1), in a museum to associate animals (tangible objects) to their environment (virtual), etc. From these four interactive technologies, we note that currently the systems are mainly centered on a one at a time type of usage. The users can use one system or another in turn but they rarely share the use over two or more systems. In this paper, we plan to propose architectures which make it possible to share an application between several users, and several platforms using several types of interaction. In this context, the user interface becomes distributed.

NEW TRENDS FOR DISTRIBUTED INTERACTIVE TABLES AND OTHER SURFACES & PROPOSITION

The arrival of mobile platforms such as the PDA, Smartphones and others has been the subject of many research projects. The objective is to facilitate the migration of applications when the context changes. In the field of Human-Computer Interfaces, the objective is similar, following the idea of the plasticity of the interfaces which was then adapted to distributed interfaces [1,3,5,7]. The user wanted to be able to move from one platform to another one without loss of coherence in the use of his/her application, without loss of data (e.g. he/she wanted to continue to deal with his/her e-mails or surf on Internet while being mobile). The Cameleon model became a framework for the modeling /transformation of the HCI [2]. Within this framework, the transformation is done according to the characteristics of a context, i.e. of a user, a platform and an environment. In our research, we aim to extend this work to the simultaneous use of several users (collaborative context), several platforms and consequently several environments [6]. In this article, we focus on the possible interactions with an interactive table and other supports. We are working here towards an approach which involves "taking the situation into account". The users wish to share information and interfaces with other users, not necessarily using the same platform, not having the same needs/constraints. These problems based only on the virtual have already been approached by Tandler [9]. Two important problems must be considered: (1) how are the objects used on a surface with tangible interaction perceived/handled by the other tactile/virtual platforms? (2) How and when can the interface be distributed between the supports?

Tangible objects and their representation

The answer to the first question depends on the objects and the associated platform. It is necessary first to define which objects have a sense on another platform. Objects used as

functionalities for instance do not need to be distributed. For example, an object which is used on *TangiSense* to zoom should not be used from a distance by another user on another platform. *A contrario*, an object representing the clouds is an element of information which could interest a distant user concerned by the weather. This object will be regarded as a shared object. According to its display format and its interaction modalities, the tangible object will not have the same representation. It is thus necessary to establish an equivalent for platform and modality for each tangible object. For example the object clouds will have a iconic representation which is coherent with the other applications in the meteorology domain and/or will be simply written on a platform with restricted display such as Smartphones or touchscreen tablets.

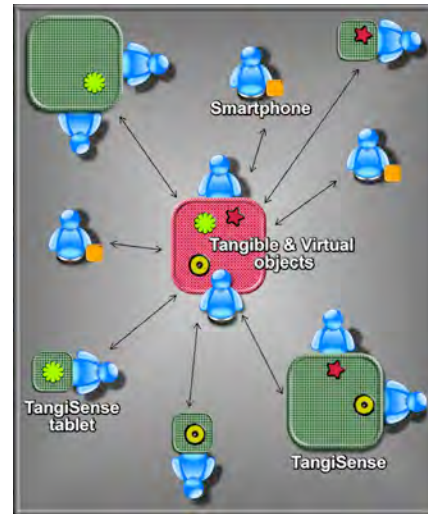


Figure 2. Centralized distribution of UI

Strategies of interface distribution

Concerning the second question regarding the distribution of the interface, we consider two possible strategies. Either (1) the interactive table is declared to be the *master* and the other devices are *slaves*. In this case, the table manages the information transfer according to the objectives of each platform and it centralizes all the information available in the distributed system (Cf. Figure 2). The interface distribution can be seen in the form of tree. The *master* surface is the root of the tree (in the figure, this surface is the interactive table). The other platforms correspond to the nodes or leaves; in the figure, children $(S_1)=\{S_2,S_3,S_4,S_5,S_6,S_7,S_8\}$. The relationship means that the child interface corresponds to a subset of the interface of the parent node. $\forall n, 2 \leq n \leq 8, S_n \subset S_1$. The transformations performed (tangible to virtual for example) are also specified in the relation by relation functions. The *master* table takes responsibility for choosing the adequate mode of representation to transmit to the target platform. For instance, the placement of an object on the master table which represents a choice is represented by a list on the smartphone concerned by this choice. The users use objects

on the *master* table in order to connect it to others platforms and select the UI to share. This strategy is useful when the UI is complete on one support with priority and if UI have to be distributed on other supports. The disadvantage of this strategy is that breakdowns are not tolerated.

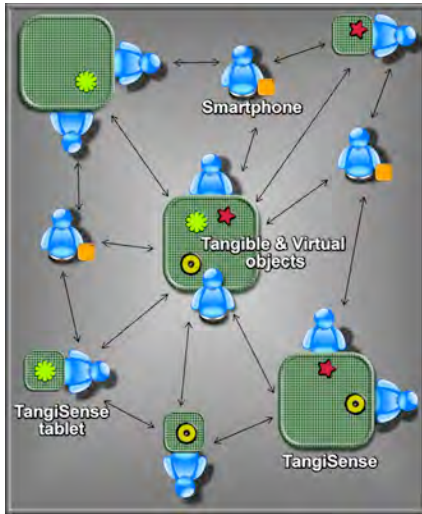


Figure 3. Network of Distributed UI

Or (2) all the platforms are autonomous and at the same decision level (Cf. Figure 3). The set forms a graph where n corresponds to the number of distributed interfaces (in figure 3, $n=9$). Here, a relation between two platforms means a distributed interface. There can be several functions of relation. Either the two parts of user interfaces (UI) are complementary: $UI=UI_1+UI_2$, or there are common parts to both interfaces: $UI=UI_1+UI_2$ and $|UI| < |UI_1|+|UI_2|$. As an interface can be linked to several others, it must compose the set of the concerned interfaces. For example if an interface is linked to 3 others following the functions f_1 , f_2 and f_3 , then $U=f_1(UI_1)+f_2(UI_2)+f_3(UI_3)$. The functions could be, for example, ensembles relations [8]. The interesting functions for the distribution and collaboration are complete duplication, partial duplication, the part extraction, etc. For example U can be the fusion of the complete duplication of U_1 , partial duplication of U_2 and the original (master) of U_3 . Each platform is responsible for being connected to the platform with which it interacts and the all of the exchanges are done in pairs. In this case, each platform must manage several UI distributed with several platform. When the user needs to move or to share a UI with another user, the platforms have to connect them and the UI is deployed according to the local context (Platform, User, Environment).

CASE STUDY

This case study presents a possible use of the table, of the tangible and/or virtual objects associated to it and of various devices used by several users in the case of a crisis management unit.

Crisis management unit

When a significant event such as a forest fire occurs, the people concerned are not together. Some are situated at a place where information is centralized; supervisors/decision makers are to be found among them. They collect information from other actors who are geographically separated on the ground, concerning elements such as like the state of the fire, its propagation velocity and the direct implications. The crisis unit makes decisions based on the collected information and must transmit them to the on-site teams. They are also in contact with other structures such as the police officers who must, according to the case, prohibit access to certain zones or warn/evacuate the potential disaster victims. The state of the system at one given moment with an example of use per device and actor is shown in Figure 4.

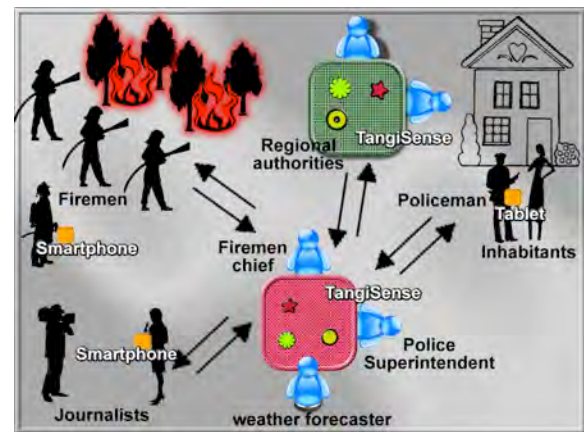


Figure 4. Crisis unit using TangiSense and other platforms

Scenario of the distribution operation

To illustrate the functioning of the system, an extract of scenario presenting the functioning of the IU distribution is proposed. To ask for a collaboration between two tables, a user U_1 interacting with the *TangiSense* table T_1 poses an object activating a “request for collaboration” on the table. : The message “name U_1 wishes to collaborate” is transmitted on the other platforms connected beforehand. U_2 answers by posing an object having for significance “me”. The equivalent of the application starts on T_2 . There is duplication of the interface, the tangible objects posed on T_1 are represented in a virtual way on T_2 . The tangible objects (and their associated functionalities) specific to U_2 and to the application are activated. In the case of this scenario, the whole of the interface is shared by duplication. It could be divided partially in this case U_1 should say which part (physical or logical) of the interface it wishes to divide. Two cases arise then, either U_1 accepts that its own objects are moved, in this case U_2 can handle the associated virtual object or U_2 uses another tangible object which it will have associated to the virtual object. The Table T_1 must then mean to U_1 which are the displacements or actions operated by U_2 so that U_2 handles its objects to keep the coherence, U_1 can also refuse the proposals of U_2

who is informed. When one of the two users wishes to stop the collaboration, he/she poses an object “I leave”. If the division is done by distribution, the interfaces can be restored on the table which remains active.

Figure 4 shows a centralized version of crisis management. It is the table which manages the interfaces and which, transmits the UI to the other platforms according to, for example, the scenario given previously. Indeed in this context, the whole of the interfaces is available on the interactive table which is master. The other platforms are regarded like child of this table and collect the interfaces that the master table authorizes to share. In this case, it is the master table which combines according to the need of interfaces. Given: an interface UI1 dedicated to the weather, UI2 dedicated to the cartography, UI3 and UI4 two interfaces dedicated to the firemen, UI5 tangible objects dedicated to the placement of firemen and UI6 tangible objects dedicated to the placement of police officers. An interface U on a master table is visible of Figure 5 while U' is the distributed part presented to the firemen.

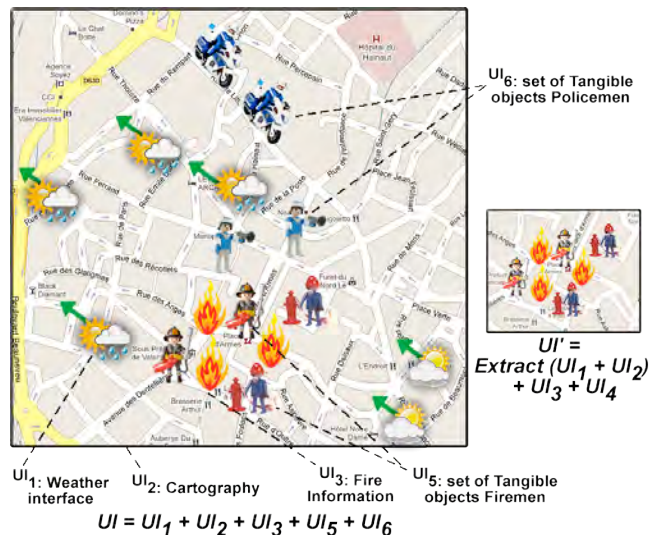


Figure 5. Examples of DUI

CONCLUSION & PROSPECTS

Starting from a context of distributed and collaborative interaction, we gave strategy areas of user interface distribution in the typical case of collaboration centered on an interactive table. A case study based on *TangiSense* and other surfaces made it possible to illustrate the strategies of distribution of the user interfaces. In order to distribute an interface given on two supports, it should be remembered that it can be seen as a set of interface elements (container / contained). It can thus be broken down and recomposed according to the context [8], the design and the evaluation of such new distributed interfaces open many prospects for research.

ACKNOWLEDGMENTS

This research work was partially financed by the “Ministère de l'Education Nationale, de la Recherche et de la Technologie”, the “région Nord Pas de Calais”, the “Centre National de la Recherche Scientifique”, the FEDER, CISIT, and especially the “Agence Nationale de la Recherche” (ANR TTT and IMAGIT projects).

REFERENCES

1. Bandelloni, R. and Paternò, F. Migratory user interfaces able to adapt to various interaction platforms. *Int. J. Human-Computer Studies* 60, 5-6 (2004), pp. 621-639.
2. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. *Interacting with Computer* 15, 3 (2003) 289-308
3. Demeure, A., Sottet, J-S., Calvary, G., Coutaz, J., Ganneau, V., and Vanderdonckt J. (2008) The 4C Reference Model for Distributed User Interfaces. *Proc. of 4th International Conference on Autonomic and Autonomous Systems ICAS 2008*. D. Greenwood, M. Grottkke, H. Lutfiyya, M. Popescu (eds.), IEEE Computer Society Press, Los Alamitos, Gosier, pp. 61-69.
4. Finkenzeller, K. (2003). *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, New York.
5. Grolaux, D., Van Roy, P., Vanderdonckt, J. (2004) *Migratable User Interfaces: Beyond Migratory User Interfaces*. In *Proc. of MOBIQUITOUS'04*. IEEE Computer Society Press, Los Alamitos, pp. 422-430.
6. Kubicki, S., Lepreux, S., Kolski, C., and Caelen, J. (2010). Towards New Human-Machine Systems in contexts involving interactive table and tangible objects. *11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, Valenciennes, France.
7. Larsson A., Ingmarsson M., Sun B., (2007) *A Development Platform for Distributed User Interfaces*. In *SEKE07*, Boston, U.S.A.
8. Lepreux S., Vanderdonckt J., Michotte B. (2006). Visual Design of User Interfaces by (De)composition. G. Doherty, A. Blandford, *Proceedings of 13th Int. Workshop on Design, Specification and Verification of Interactive Systems DSV-IS'2006* (Dublin, 26-28 July 2006), Springer, LNCS, Berlin, pp. 157-170.
9. Tandler, P. (2004) The BEACH Application Model and Software Framework for Synchronous Collaboration in Ubiquitous Computing Environments. *Journal of Systems & Software (JSS) Special issue on Ubiquitous Computing*, 69(3):267-296.

Dual Displays: towards an interaction model and associated design guidelines.

Nima Kaviani, Matthias Finke, Rodger Lea, Sidney Fels

Media And Graphics Interdisciplinary Centre (MAGIC)

University of British Columbia, Vancouver, Canada

{nkaviani@cs, {martinf, rodgerl, ssfels}@ece}.ubc.ca

ABSTRACT

There has been significant research interest over recent years in the use of public digital displays and in particular their capabilities to offer both interactivity and personalized content. A promising approach in interacting with large public displays has been the use of ubiquitous cell phones which not only offer a means to interact with displays, but increasingly offer a small, but high quality screen to complement the larger public display. A dual screen approach suggests a number of intriguing possibilities including a potential solution to the problem of managing conflicts when using a shared screen in a public setting. However, until we have formulated a well-founded interaction model for such dual displays, we believe it will be hard for the community to develop coherent guidelines for system and application developers. Our position is that we can build on top of Norman's seven stage model of interaction, and that such a formalism would help designers better cope with the requirements of a dual display design.

Author Keywords

Interactive large displays, small devices, distributed user interfaces, user study.

ACM Classification Keywords

H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION

Large public displays are becoming increasingly prevalent, rapidly replacing conventional paper-based methods of presenting information to the public.

Extending interactive large displays (LD) with small devices (SD) such as PDAs or smart phones has been discussed in earlier research efforts [3, 6]. The main idea behind this approach, which is also known as the *Dual Display* approach [2], is to execute a user interface across

LD and SD to take advantage of input and output capabilities of both device types at the same time. Dix and Sas [3] argue that such an approach could help designers to solve GUI design issues due to multi-user interaction with large public displays.

Current research work with large public displays rests on the assumption that interaction feedback and user requested information (output data) can be presented on LD, SD, or a combination of both. Furthermore, it is assumed that coupling LD with SD during interaction helps to reduce the load of information presentation on the LD and increases users' ability to manage content on large displays, mainly because of users' inherent experience in using their phones. What seems to be missing from the current research work is identifying differences in design requirements for interactive and non-interactive widgets depending on whether they are placed on LD or SD. There is lack of clear guidelines on how users respond to placement of elements in a user interface on LD or SD.

This paper builds on our previous research [6, 7] into the use of dual displays and explores how well users comprehend the nature of interaction in a dual display environment where both input and output interface widgets are distributed across LD and SD. We take Norman's model for seven stages of interaction and formally discuss it within the context of a dual display design where interactive or non-interactive widgets can equally live on the LD or on the SD. We hope the results of providing this formalism, coupled with undergoing experimental analysis, can lead to design guidelines that would allow system designers better decide about placement of user interface widgets when it comes to application design within a dual display setting.

RELATED WORK

Jin et al. [9] introduce an approach in which the combination of a handheld device and a large public display is used to share and manage the information. Content can be shown on both LD and SD based on users' choice. The authors however have not validated their approach through any formal user study. Nestler et al. [5] have evaluated collaborative interaction concepts, for a game application, conducted between touch-based tabletops serving as large displays and small devices (e.g., PDAs). Their results from the user studies show that small devices enabled users to

remotely collaborate with others performing tasks on the tabletop. Carter et al. [4] envision a scenario in which users can annotate content shown on a large display in order to encourage collaboration and discussion. The results of their user study show that SD could serve as a medium to present information to the users; enable users to modify the content; and preserve a copy when leaving the interaction scene.

As shown in the brief review above, while there have been a number of investigations into the use of dual displays, there exist limited number of research works looking into combining interactive and non-interactive widgets when designing a dual display application. To the best of our knowledge, the examination of input and output capabilities of SD and LD for designing interactive user interface, which forms the core of our research, is largely unexplored.

DISTRIBUTED USER INTERFACE DESIGN

Our primary research question is to verify if users benefit from executing an application across large displays and small devices taking advantage of input and output capabilities of both devices, i.e. LD and SD. In other words we would like to understand if and how splitting interface entities (user interface widgets) across LD and SD affects user task performance when interacting with applications designed for large public displays.

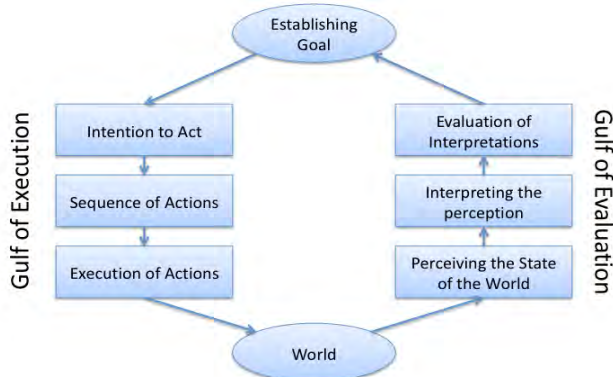


Figure 1: Norman's seven stage model

We apply Norman's seven stage model [8] to explore our research question in more detail. The model categorizes users' mental activities into *Execution* and *Evaluation* stages in order to describe an entire *Cycle of Interaction*. During Execution, user goals are translated into an *Intention*, which will be converted into a *Sequence of Actions* that has to be *Executed*. During Evaluation, users *Perceive* the new system state based on the execution and *Interpret* it. Based on a subsequent *Evaluation* they determine whether they achieve their goals.

This model allows us to separately explore the input and the output of a (distributed) user interface and how users respond to it during the execution and evaluation stages. According to Norman, different mental activities are

associated with the input and output and so widgets allowing users to enter input might exhibit different design constraints compared to output widgets. Hence, we denote widgets associated with system input as *interactive-widgets* and those associated with system output as *non-interactive widgets*. Non-interactive widgets are in general used to present system states to users. However, interactive widgets accept user input to initiate system state changes.

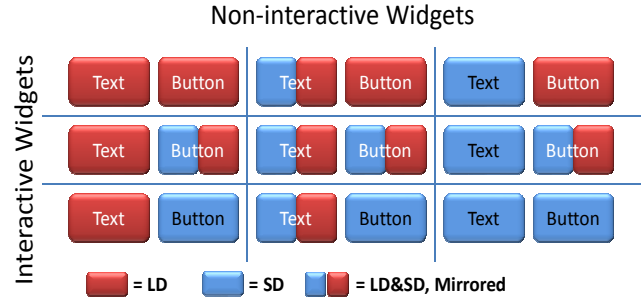


Figure 2: Placement complexity of paired Text and Button widgets across a large display (LD) and small device (SD)

There is considerable complexity concerning the placement of non-interactive and interactive widgets when designing a distributed user interface across LD and SD. Imagine a simple interactive application containing only a *Button* widget that triggers a *Text* widget to print "Hello World". The button widget can be placed either on the LD, the SD or both. The same is true for the text widget. Hence, for this very simple case we can already come up with 9 possible settings (see Figure 2). Each setting might provide different solutions for interface issues a designer has to solve.

Execution Path

When creating a distributed user interface using a smart phone as the primary input gateway, a designer has to make the additional decision of where to place the interactive widgets enabling users to execute a sequence of actions (see Figure 3) and so to enter their input.

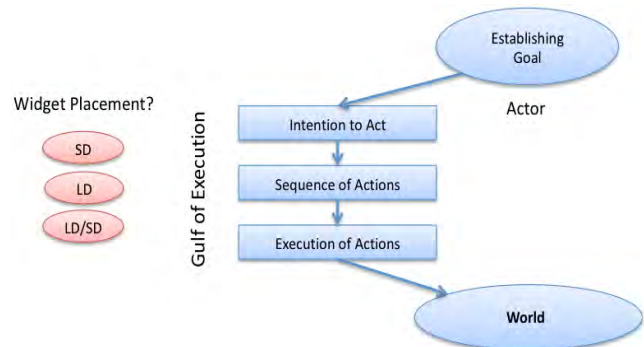


Figure 3: Interactive Widget Placement

Considering multi-step interactions that involve more than one interactive widget, we can come up with four basic

design strategies: i) *LD mode*: With this design strategy, users use their smart phone to directly interact with widgets placed on the large public display. Although simple, in a multi-user scenario; problems can arise where more than one user want to access the same interactive widget. Furthermore, a sequence of actions might require a user to interact with several widgets over a longer time period which demands for space or 'real estate' on the LD; ii) *SD mode*: Moving interactive widgets down to the SD will free up real estate on the LD. Placing the interactive widgets on the display of a smart phone will also help ameliorate the multi-user access problem. Conversely this approach has the added complexity of using both the primary LD and the secondary SD where the smaller display imposes limitations in widgets placement compared to available real estate on the LD. iii) *LD-SD mode*: Here some interactive widgets are placed on the LD and some on the SD. For example, a menu bar widget placed on the LD with associated menu widgets placed on the SD when activated by the users on the LD. The advantage here is that every user interaction has a defined entry point, which is in this case the LD. On the other hand, an interactive widget arrangement that is partly across LD and SD may cause user confusion; *Mirrored mode*: This design strategy introduces redundancy into the distributed user interface by placing the same widgets on both LD and SD. Hence, the audience can clearly see what the user enters. Of course, we neither solve real estate nor multi-user access problems, but users can make their own choice of selecting the display they want to interact with, without the risk of getting confused.

Evaluation Path

Similar to placement decisions regarding interactive widgets, a designer of a distributed user interface has to make the choice of where to place non-interactive widgets that present a new system state as a result of user interactions.

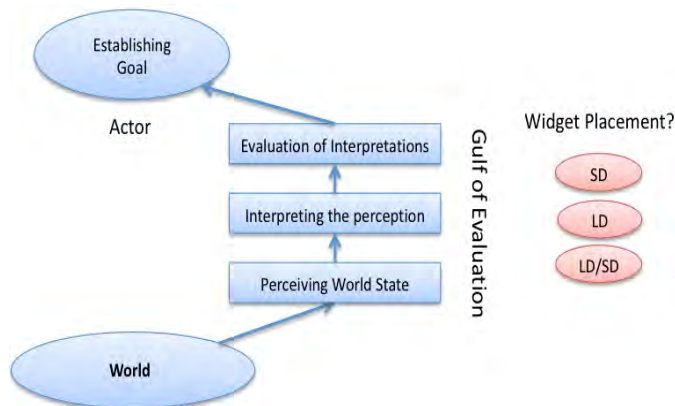


Figure 4: Non-interactive Widget Placement

Since non-interactive widgets are meant to present content while interactive widgets are mainly to receive user inputs, the placement decision a designer has to make in case of

non-interactive content would be based on the gains and the loss in each design decision as described below: i) *LD mode*: This is the classic design strategy used by the majority of today's interactive large public display installations. From the design point of view the LD mode is a challenging task. A designer has to find the right balance between available real estate on the LD and the quantity and the quality of content presented on the LD to properly serve the audience; ii) *SD mode*: Forcing users to rely solely on a SD defeats the purpose of large public display installations, which are designed to be enjoyable for an entire audience. However, in some circumstances, e.g. content that requires privacy or security considerations, utilizing this mode may be useful for implementing certain aspects of a large display application; iii) *LD-SD mode*: This mode combines the advantages of LD and SD. Content that might be interesting to the entire audience can be presented using non-interactive widgets on the LD. On the other hand, content specific to a requesting user, and of less interest and importance for the rest of the audience, can be placed on the SD; iv) *Mirrored Mode*: In this mode the very same content is presented through same non-interactive widgets placed on the LD and SD. Here users can make their own choice of which display they use to view content.

As listed above, the overall design strategies for non-interactive widgets compared to interactive-widgets differ in the way they serve and apply to distributed user interface concepts/issues.

EARLY EXPERIMENTS

To help validate our design strategies we have developed a number of initial applications that can be run in both single screen and dual screen modes. To explore the execution gulf of Norman's model, we developed two games namely the *Polar Defense Game* [7] and the *Eyeballing Game*. The design of Polar Defense relies on coarse grained interaction with the display where changes in the representational states of the GUI happen in the form of coarse and noticeable feedback received as a matter of cursor transmissions between blocks of a grid-like user interface. In contrast, the Eyeballing game focuses on fine grained interaction where changes in the representational state of the application are much smoother and at the level of pixels making borders for geometrical objects. By covering coarse-grained and fine-grained interaction patterns, we aimed at covering the two extremes on the spectrum of possible interaction styles using a mobile device and a large display. To explore the evaluation gulf of Norman's model, we developed an interactive directory application providing information about points of interest (POIs) on a map, by showing information about venues of interest in a city either on the LD or on the SD.

To this point, we have conducted some informal user studies through which we asked our participants to interact with each of the above applications both in LD and LD/SD

modes. In the LD mode, all feedback to the users appears on the LD, and SD is only used as an input medium; whereas in the LD/SD mode, part of the output also occurs on the SD. We provided our participants with post experiment questionnaires, asking about their interaction experience with each of the above applications in a dual display setting. In the case of exploring the execution gulf in Norman's model, initial results indicate that participants preferred more to interact in the LD/SD mode. Furthermore, our participants thought they made fewer mistakes when interacting in the LD/SD mode. They also found the interaction with the application more natural when doing it on SD. However, they didn't think they were faster in the LD/SD mode.

As for assessing Norman's gulf of evaluation, our participants indicated that they would like to get detailed information on their phones and in particular read and scroll text on the SD rather than on the large display. However, we were not able to validate whether participants found interaction on SD easier than LD or if they preferred perceiving non-text data (e.g., images) on SD over LD.

DISCUSSION & CONCLUSION

To date, there is a lack of understanding of how users respond to placement of elements of a user interface on large public displays and mobile phones. Toward establishing some design guidelines for such dual display applications, we would like to understand the effect of user performance when user interface widgets are split across these displays. Building on top of Norman's seven stage model of interaction, we have identified four different ways that interactive and non-interactive widgets can be distributed across the mobile and large displays. To offer designers the flexibility of choosing among these four possible widgets placements, we have to first understand how placement affects user performance when executing a sequence of actions. That is, user satisfaction should stay the same regardless of the placement setting they choose to interact with the large display using the mobile phone.

We believe that the use of a formal model for associating design decisions with a dual display setting with users' stages of interaction and perception of a system's behavior can lead to better design decisions within this context. The goal is to identify the key design decisions that, when applied to a dual display setting, lead to the best experience in terms of performance and satisfaction when performing a combined LD/SD interaction.

Toward addressing these open research questions, we have built several LD/SD applications and are now conducting several experiments. Preliminary results from our initial

user studies demonstrate that dual display user interface design can leverage the use of SD to provide better user experience and information perception for the users.

Based on these early results, we are optimistic that we can provide some value design guidelines that will lead to increased user satisfaction when designing applications for large public displays that interact with mobile phones.

REFERENCES

1. Akio Shinohara and Junji Tomita and Tamio Kihara and Shinya Nakajima and Katsuhiko Ogawa. A Huge Screen Interactive Public Media System: Mirai-Tube. HCI (2), 4551: 936-945, 2007.
2. Alan. Dix. Small meets Large: research issues for using personal devices to interact with public displays. (unpublished) internal discussion paper, Lancaster University, January 2005.
3. A. Dix and C. Sas. Public displays and private devices: A design space analysis. WSh. on Designing and evaluating mobile phone-based interaction with public displays. CHI2008, Florence, 5th April 2008.
4. Carter, S., Churchill, E., Denoue, L., Helfman, J., and Nelson, L. 2004. Digital graffiti: public annotation of multimedia content. In CHI '04 Extended Abstracts on Human Factors in Computing Systems (Vienna, Austria, April 24 - 29, 2004). CHI '04. 1207-1210.
5. Simon Nestler, Florian Echtler, Andreas Dippon, Gudrun Klinker, Collaborative problem solving on mobile hand-held devices and stationary multi-touch interfaces. In PPD 08 WSh. on designing multi-touch interaction techniques for coupled public and private displays, 2008.
6. Nima Kaviani, Matthias Finke, Rodger Lea. Encouraging Crowd Interaction with Large Displays using Handheld Devices. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI 2009). (April 4-9, Boston, USA).
7. Matthias Finke, Anthony Tang, Rock Leung, Michael Blackstock. Lessons Learned: Game Design for Large Public Displays. In Proceedings of the International Conference on Digital Interactive Media in Entertainment and Arts (DIMEA 2008).
8. Norman, D., "Psychology of Everyday Action". The Design of Everyday Things. New York: Basic Book, 1988. 45-46
9. Chunming Jin, Shin Takahashi, Jiro Tanaka: Interaction Between Small Size Device and Large Screen in Public Space. KES (3) 2006. pp. 197-204.

Distributed User Interface for a Multi-Tablet Active Reading System

Nicholas Chen¹, Francois Guimbretiere² and Abigail Sellen³

¹Dept. of Computer Science
University of Maryland
3452 A.V. Williams Building
College Park, MD 20742
nchen@cs.umd.edu

²Dept. of Information Science
Cornell University
301 College Ave.
Ithaca, NY 14850
francois@cs.cornell.edu

³Microsoft Research Cambridge
7 J. J. Thomson Avenue
Cambridge CB3 0FB
asellen@microsoft.com

ABSTRACT

Ethnographers have shown that paper supports active reading activities because it distributes content across several independent displays. Using ethnographical reports, observing potential users, and the deployment of a technology probe, we explored how a set of interconnected tablets might provide many of the same functionalities. From these findings, we show how a multi-tablet reading system, one example of a distributed user interface (DUI), can be used to address problems associated with a multi-tablet active reading environment.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms:

Keywords: Tablets, distributed user interfaces, active reading

INTRODUCTION

Active reading [2] is the term that describes the style of reading where readers actively interact with the content they are reading in order to reflect on and better comprehend the material. Ways readers engage with the content include cross-referencing related materials, annotating, note taking, along with other reading activities. Active reading behavior is frequently observed in the reading tasks of knowledge workers like students [6] and professionals [12].

Paper tends to support active reading well, as evidenced by the large proportion of reading performed on paper. Ethnographers have long believed that one of the key reasons paper supports active reading is due to the fact it distributes content across several lightweight, independent surfaces (i.e. sheets of paper) [1]. However, paper, offering a static display, lacks the ability to provide important digital functionality, including the ability to store, access, search and share large amounts of digital content quickly and cheaply. Conversely, traditional computers, which offer digital functionality, are difficult to use for active reading. One of the main reasons for this is their form factor, which hinders users' ability to navigate and write effectively.

Tablet-style reading devices are a potential alternative. Advances in interaction such as responsive multi-touch interfaces, as well as reductions and weight in size have generated a great deal of new interest about tablet devices. In particular, the use of several thin and light tablet devices



Figure 1. Sample multi-tablet setup with four tablets and a laptop computer running the remote interaction application. Note that the visualization of the tablets in the application matches the state of the tablets.

together could convey many of the capabilities users have when using several sheets of paper simultaneously, such as the ability to compare documents [1], and read while writing [9]. In fact, the use of several interchangeable electronic “pads” for document work formed a major component in Weiser’s vision of how ubiquitous computing would shape the office of the future [14].

One of the major obstacles of a multi-tablet reading system is that existing interfaces for tablet devices are designed for use in isolation. Morris et al., who performed a study comparing various technologies for active reading, reported that users found that the complete isolation of the three TabletPCs in the study hindered digital operations [8]. Since the multi-tablet reading environment involves several interfaces split across one or more devices, it is a prime candidate for a distributed user interface (DUI) [13].

Here, we describe a DUI to support active reading on an inter-connected set of tablet devices. While our work is primarily focused on supporting reading activities, it provides broader examples of interaction techniques for DUIs to accomplish goals like unifying the devices participating in a distribution, and for handling the context changes in the system. Furthermore, we believe that the ways we have employed a DUI highlight areas beyond the realm of electronic reading systems where DUI systems may prove uniquely advantageous.

SYSTEM GOALS AND DESCRIPTION

Our reading DUI is designed to help users manage the complexities associated with reading across several tablets. In this section, we describe the major challenges that users in a multi-tablet environment face, and the features that our tablet management system provides to help alleviate those problems.

The challenges and features are motivated by requirements identified in prior ethnographic studies [1,9,12] and from a 2.5-month technology probe we conducted with 13 undergraduate English students in the Spring 2010 semester. For the probe, we supplied students first with one tablet and about 1.5-months into the deployment handed out a second tablet. Students performed a significant portion of their course readings on the devices. During the probe, we conducted bi-weekly interviews with students to apprise ourselves of the status of the deployment.

General Philosophy

We started with the core idea that each tablet should provide a full-page view of content to minimize the amount of navigation needed to view documents that were already formatted for a Letter/A4 presentation. Based on the size of the screens on our devices (9.7-inch diagonal), our tablets assumed the role of each displaying a single page at a time.

In effect, one way to see the function of our reading DUI is to think of it as a special type of window manager for *physical* windows in the form of tablet devices. The tablet setup, which compartmentalizes information into separate, but dynamic, spaces, is one that is familiar to users of modern GUIs. Windows are dynamic regions, largely independent of one another, that allow users to have multiple pieces of information viewable simultaneously. However, in order to facilitate activities that cross the bounds of windows and to organize or manage the windows, a window manager is necessary.

Unifying Tablet Content

One of the more problematic aspects of isolated tablets is that each tablet acts as an island of information. This was the chief complaint users had about the multi-tablet condition in the Morris et al. study [8]. We provide two ways to help users unify the data in these separate spaces into a single domain.

Shared Datastore and Distributed Clipboard

To preclude users from having to mentally version documents across devices, our system is set up as a single distributed data domain. As such, our system treats each display as a view into a shared data store. Changes like ink annotations are automatically propagated and synchronized to all of a user's tablet devices. The system imposes no restrictions about viewing the same page simultaneously on several devices.

In addition to sharing a document database, devices also share a common distributed clipboard [7]. The clipboard can contain arbitrary data including text and bitmaps.

Inter-Device Command System

To provide digital functionality beyond basic viewing and inking, our system provides a generic command issuance system that operates across devices. The command system aims to provide a direct manipulation-based approach for data exchange between devices, like Pick-and-Drop [10], albeit with support for additional operations. Our specific implementation operates on principles outlined in Scriboli [5], a flexible pen-based command system that uses gesture-based selections and marking menus. To use the command system, the user must first press and hold the command button to enter command mode. Following that, the user can select items using lasso and line gestures. Once the desired items are selected, users indicate the end of the selection phase by drawing a pigtail delimiter [5]. Users can extend the pigtail and invoke a command using a menu selection from a multi-stroke marking menu.

The design of the command system has the advantage of not distinguishing between commands that occur on a single device and those that cross device boundaries. For instance, the paste command following a copy command can be issued on the same device, or on a different device.

Spanning UIs Across Tablets

The reading DUI needs to balance reading tasks that require independent user interfaces (many UI-to-many device distribution), like comparing two documents, with those that benefit from multiple devices each displaying a portion of a single interface (one UI-to-many device distribution), like showing a document overview alongside page detail to aid navigation.

Similar to previous work [3], we support UI spanning using hardware links, where physically attaching devices forms a linked UI. Although this works well for more permanent relationships between devices, it can be too heavy weight for temporary associations. To address this problem, the user can specify two UIs to link together on a temporary basis by entering command mode using the command button on one device, while issuing the command on a different device. When doing this, the system performs the command as usual on the device where the command is issued, but treats the device on which the command button is depressed as the *de facto* partner. We call this style of implicitly specifying device linkage a **Conduit** interaction. Conduit reinforces the idea of a lightweight and temporary link by only being active in a user-maintained mode [11].



Figure 2. Conduit interaction for linked SFT. Left: With the command button pressed on the left device (red circle), tapping on a thumbnail on the right device opens the corresponding page on the left device. Right: Final result of the operation

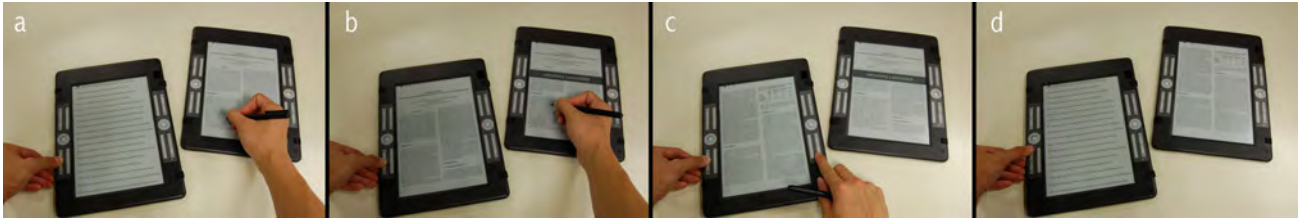


Figure 3. Remote Access sequence. (a) First, the user issues the remote access command on the remote device. (b) The local device (device where the command button is depressed) is now a proxy to the remote device. During remote access, the remote device displays a message indicating it is being remotely operated. (c) Users can navigate and otherwise interact with the local proxy of the remote device. (d) At the end of command mode, the local device reverts back to its original content while the remote device updates to match the state of its proxy.

For example, Conduit can be used to link the Space-Filling Thumbnail (SFT) [4] function of our system across devices (Figure 2). In linked SFT, when one device is showing the thumbnails of a document, the user can enter command mode on a second device. Selecting a page on the SFT view of the first device causes the second device to display the full-screen view of that page. In this way, the user can combine the two UIs to see both an overview of the document along with page detail.

Providing Alternatives to Physical Interaction

Although tablets convey some of the physicality of paper, they are still considerably thicker and heavier than a sheet of paper. Since people frequently read with devices in hand, there are many situations where users have to put a tablet down and then shuffle tablets around to work with a device that is far away, which can be cumbersome.

To provide users with an alternative to physically moving devices around, we take advantage of the distributed nature of the UI to implement a remote access feature that enables users to temporarily use a device as any other (Figure 3). When this function is activated (via the Conduit mechanism described above) the state of the remote device is transferred to the closer device and the user can interact with the closer device as if it was the remote device (Figure 3 b,c). During this time, the remote device displays a message notifying the user that it is being remotely controlled. When the user releases the command button, which ends the command, the nearby device reverts to its original state and changes are reflected to the remote device.

Adapting to Environmental Changes

Various requirements can dictate the computing resources a user has available. In our study, some users mentioned that having two devices was helpful but did not outweigh the convenience of carrying only one device around, while others felt the opposite. Furthermore, we expect users to have more tablets available in locations like at home or in the office than while travelling. Therefore, we designed our reading system to handle the fact that devices can enter or leave the distribution at any time.

The Session Manager (Figure 4) component of the system stores the state of devices in a distribution at any given moment. When devices leave, the session manager allows the content on that device to be recalled and distributed among the remaining devices. Conversely, when the user

returns to an environment with more devices available, the user can re-create and resume a prior setup that relied on a larger number of devices. This way, the system gracefully handles changing numbers of resources.

Working Across Heterogeneous Devices

Although our system is primarily for tablet management, studies have shown that traditional laptop or desktop configurations are better for certain reading activities [8]. We realized that using tablets alongside traditional computers would require awkward context-switches between using a mouse and keyboard to using a stylus. We use the remote access functionality to remedy the situation by letting the user control the tablet devices directly from a PC. The user can use the computer to call up a visualization of the tablets in the environment (Figure 1). From this visualization, the user can select a particular tablet and then control the tablet using a keyboard and mouse as if the tablet was a window on the computer desktop.

DISCUSSION

Dynamism in DUIs

One of the strengths a UI distributed across multiple devices has over a monolithic system is the ability to flexibly add or remove computing resources as circumstances dictate. Consequently, there are scenarios where DUIs will need to support and embrace a large degree of dynamism.

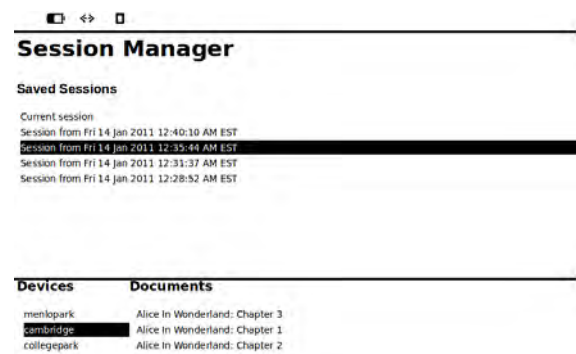


Figure 4. Screenshot of the Session Manager interface. There are 4 saved sessions in the history. In the selected saved session, the device **cambridge** was being used to view three documents.

Active reading tasks are a particularly good example since reading activities rapidly evolve and can occur in a variety of environments. As tasks shift, the optimal interface may change as well. The way our system handles a high degree of dynamism in device configurations and task requirements is through a mix of implicit and explicit UI mechanisms to reconfigure the distribution.

The implicit mechanisms are characterized by the *lack* of an interface for performing reconfiguration. Rather, these mechanisms rely on the fact that operations are identical regardless of what devices are available in the environment (as in the case of the command issuing mechanism), that the reconfiguration can piggy-back on natural manipulations on the tablets (like hardware links) or that the process can be made transparent to the user (such as the fact that the user can simply start using an extra tablet as needed). We believe that implicit mechanisms are preferable because they allow the user to proceed with the task without the overhead and interruption of an explicit UI interaction.

For other reconfiguration operations, explicit mechanisms are necessary. However, to keep the system agile, we make sure that these explicit mechanisms are optional, and can be delayed (as in the case of using the session manager history). We also place an emphasis on explicit operations that offer the option of temporary reconfiguration, from which users can easily revert to a prior mode if desired.

With these reconfiguration capabilities, the interface can adapt to user needs with a minimum amount of overhead.

Beneficial Implications for DUIs

By splitting a user interface into distributable components, DUIs make possible some unique functionality. Here, we identify two of the more interesting implications that arise from incorporating a DUI into our system.

Distribution Enables Redirectable Interfaces

In a DUI construction, interfaces can be redirected and shipped around electronically so that the user can interact using a proxy device. In our system, this has been useful for minimizing physical interactions when these interactions are undesirable and to allow users to effect changes without incurring the costs of device switching.

Distribution Helps Interfaces Scale In Both Directions

The devices available in a given environment are determined by a combination of factors that include the available infrastructure, concerns about portability and users' personal preferences. By allowing UIs to be broken down into distributable pieces, the UI can better accommodate both device-rich and device-starved situations. Since a DUI-based system allows users to easily multiplex UI elements across devices in space or time, the system can take advantage of additional devices, while offering fallbacks (like cycling through UIs) for device-starved environments.

CONCLUSION

In this paper, we present a tablet management system that demonstrates solutions for several requirements of active reading. Our system provides a set of novel features for

unifying a distributed interface, adjusting the distribution and handling changes to the distribution. Although the interactions we introduce have been tailored to an active reading environment, we believe they are illustrative of challenges and opportunities for DUIs in general.

ACKNOWLEDGEMENTS

Funding for the project comes in from NSF Grants IIS-0812196 and IIS-0936105, and a gift from Microsoft Research. Chen is supported by a Google Ph.D. Fellowship.

REFERENCES

1. Adler, A., Gujar, A., Harrison, B.L., O'Hara, K., and Sellen, A. A diary study of work-related reading: design implications for digital reading devices. *Proceedings of CHI '98*, ACM Press/Addison-Wesley Publishing Co. (1998), 241-248.
2. Adler, M.J. and van Doren, C. *How to Read a Book*. Simon and Schuster, New York, NY, 1972.
3. Chen, N., Guimbretiere, F., Dixon, M., Lewis, C., and Agrawala, M. Navigation techniques for dual-display e-book readers. *Proceedings of CHI '08*, ACM (2008), 1779-1788.
4. Cockburn, A., Gutwin, C., and Alexander, J. Faster document navigation with space-filling thumbnails. *Proceedings of CHI '06*, ACM (2006), 1-10.
5. Hinckley, K., Baudisch, P., Ramos, G., and Guimbretiere, F. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. *Proceedings of CHI '05*. ACM (2005), 451-460.
6. Marshall, C.C. and Ruotolo, C. Reading-in-the-small: a study of reading on small form factor devices. *Proceedings of JCDL '02*. ACM (2002), 56-64.
7. Miller, R.C. and Myers, B.A. Synchronizing clipboards of multiple computers. *Proceedings of UIST '99*, ACM (1999), 65-66.
8. Morris, M.R., Brush, A.J.B., and Meyers, B.R. Reading Revisted: Evaluating the Usability of Digital Display Surfaces for Active Reading Tasks. *Proceedings of TABLETOP '07*, (2007), 79-86.
9. O'Hara, K. and Sellen, A. A comparison of reading paper and on-line documents. *Proceedings of CHI '97*, ACM (1997), 335-342.
10. Rekimoto, J. Pick-and-drop: a direct manipulation technique for multiple computer environments. *Proceedings of UIST '97*, ACM (1997), 31-39.
11. Sellen, A.J., Kurtenbach, G.E., and Buxton, W.A.S. The Prevention of Mode Errors Through Sensory Feedback. *Human Computer Interaction* 7, 2 (1992), 141-164.
12. Sellen, A.J. and Harper, R.H.R. *The Myth of the Paperless Office*. MIT Press, 2003.
13. Vandervelpen, C. and Coninx, K. Towards model-based design support for distributed user interfaces. *Proceedings of NordiCHI '04*, ACM (2004), 61-70.
14. Weiser, M. The computer for the 21st century. In *Human-computer interaction: toward the year 2000*. Morgan Kaufmann Publishers Inc., 1995, 933-940.

Supporting Distributed Decision Making Using Secure Distributed User Interfaces

Thomas Barth, Thomas Fielenbach, Mohamed Bourimi, Dogan Kesdogan
Information Systems Institute - IT Security Group
University of Siegen
Hölderlinstr. 3, 57076, Siegen, Germany
{barth;fielenbach;bourimi;kesdogan@fb5.uni-siegen.de}

Pedro G. Villanueva
ISE Research group
Computing Systems Department
University of Castilla-La Mancha
Av. España S/N. Campus Universitario de
Albacete. (02071) Albacete, Spain
Pedro.gonzalez@uclm.es

ABSTRACT

In disaster situations people with different roles are cooperating to solve emerging problems. Cooperative systems providing enhanced interaction facilities in the user interface (e.g. direct manipulation techniques) could substantially support decision making especially for geographically distributed cooperating teams. Thereby, sensitive information has to be shared in a common workspace requiring different handling procedures according to the different roles involved in the process. In this paper, we propose the use of a common multilaterally secure distributed user interface to support the decision making in geographically distributed groups of professionals. The system combines a collaborative multi-pointer system with an anonymous credential security system to provide users with an easy way to share and access information in a secure way ensuring the privacy.

Author Keywords

Distributed user interfaces, security and privacy, HCI, decision making, multilateral security, mission-critical situations.

ACM Classification Keywords

H.5.2 User Interfaces: Graphical user interfaces (GUI), K.6.M. Miscellaneous: Security.

General Terms

See list of the limited ACM 16 terms in the instructions, see <http://www.sheridanprinting.com/sigchi/generalterms.htm>.

INTRODUCTION

When catastrophes and crises happen, various people with different roles and privileges according to their specific expertise and function are working together within the process of analyzing, evaluating and deriving counteractive

measures. Typically, they (e.g. police, firefighting forces, supply-chain partners, invited journalists to inform the public etc.) contribute their individual, domain-specific knowledge from different geographical locations when solving emerging problems. Since such situations need fast reactions, cooperative/collaborative systems are often used. Furthermore, handling these kinds of decision-making situations results in very knowledge-intensive ad-hoc processes that cannot be fully planned and automated a priori. Supporting the decision makers during these processes typically means to provide them with problem-specific information from all related domains in an efficient and easy-to-use way. Hence, these systems provide shared environments/workspaces equipped with different communication, cooperation and coordination means to ease the collaboration. Thereby, enhanced interaction facilities built-in in those environments for the manipulation of shared artifacts (e.g. used documents) could substantially support fast and efficient decision making in these knowledge-intensive situations, especially for geographically distributed cooperating teams. Therefore sensitive information belonging to the participating organizations/individuals have to be shared in a common workspace/environment, which requires different handling according to the different roles in order to assure respecting security and privacy requirements of all partners.

In this paper, we propose utilizing a multilaterally secure and privacy preserving system supporting distributed user interfaces in a real-life supply-chain scenario in the ReSCUeIT project [6]. The scenario targets the optimization of the product recall process in food supply chains in order to secure the consumer against risks and threats posed by food. We show how to extend a distributed, secure and process-oriented supply-chain software infrastructure integrating the WallShare [7] collaborative system with an anonymous credential system such as Idemix. This can facilitate decision making by allowing for fine-grained secure access rights for the shared artifacts in the common shared workspace created in a particular situation to exchange and process the required information.

The reminder of this paper is structured as follows; first we

describe the scenario and derive the requirements of the system. Then, we present related work followed by our approach. Finally, we conclude by reporting on ongoing work and envisaged future extensions.

SCENARIO AND REQUIREMENTS ANALYSIS

The ReSCUe IT project [6] is a German-French research and development project focusing on increasing the safety of the civil population in presence of threats by intended or unintended risks introduced to the food supply chain (e.g. contamination or decay during production or transportation). The project integrates partners from academia, food production, retail, and logistics in order to assure the consideration of requirements from all different stakeholders cooperatively involved in the food supply chain.

Even though close collaboration of partners along the supply chain is certainly inevitable and common practice for decades, they still have their individual requirements on the security and privacy of their mission-critical data, e.g. from quality management, logistics, pricing etc. Also in the presence of substantial threats to the supply chain as a whole, they cannot afford to make all relevant data available to their supply chain partners when preserving or reconstituting the integrity of the supply-chain, their processes or the products distributed along the supply-chain.

Hence, one essential goal of the project is to provide a secure and robust software platform covering the whole lifecycle of (business) processes along the supply chain: From modeling of processes and the involved threats and risks over any asset involved in the supply chain up to the actual execution of business processes and their recovery in the case of disturbance. One central process when handling disturbances of the supply chain is the recalling of products identified as a risk to the end consumer. The specific risk must be determined and can range from simply warning the consumer and asking them to return a specified product to the shop to starting a process to withdraw all products in all shops in a very short timeframe from two hours in case of a very severe, maybe even lethal threat to the health of consumers. Obviously, making this decision is extremely demanding, responsible, time-critical and depends on a plethora of diverse information. The expert commission involved in this decision-making situation largely depends on having all relevant information at hand immediately and being supported during the discussion of the information independent from their actual geographical location. As already mentioned, this collaboration is subject to multi-lateral security requirements of all participating persons/organizations. Summarizing, these requirements must be met by an adequate software environment useful in the given situation: security and privacy respecting, process-oriented, enabling efficient document sharing

according to a user/role model.

From the architectural point of view, this kind of secure document sharing environment is envisaged to be a part of the overall ReSCUeIT platform. This platform is currently designed and implemented as an open and secure, process- and service-oriented infrastructure, which is the basis for integrating all relevant applications and data sources along the supply chain. Foremost challenge is satisfying multi-lateral security requirements and simultaneously providing all necessary information from all supply chain partners in the case of crisis to recover from this crisis. Functionality provided by the platform to reach these goals is also available to the document sharing facility outlined in this contribution such that this facility can be seamlessly integrated into the platform utilizing the same security- and privacy-preserving functionality as all the other components without the need to implement it explicitly.

Hence, the idea is to use a single common pool for sharing all needed information in form of shared artifacts that could be accessed and manipulated according to the different access rights assigned to a giving role by the pool administrator. Such proceeding was proposed based on brainstorming with different people from the security and HCI as well as CSCW research fields. Using a flat shared workspace ensures no information is ignored, each incoming information is perceived by all involved people, and available information can be processed in parallel which could enhance and improve decision making (e.g. parallel brainstorming or annotating artifacts).

People in the same physical room see the same interface (virtual pool) at the wall but have different interfaces in their mobile devices according to their access rights (based on the different roles). Furthermore, supporting involved people users' mobile devices eases the interaction with the system since no additional hardware is needed.

RELATED WORK

With respect to enhanced interaction in computer systems, user interface technology has advanced from command line interfaces to the established and approved use of direct manipulation techniques or WIMP (windows, icons, menus and pointing) [5]. A good overview of enhancing interaction in collaborative systems by using such techniques in general can be found in [4]. There, the adaptability and usability of a collaborative system named CURE has been particularly improved by supporting direct manipulation techniques for navigation as well as tailoring (e.g. drag and drop support for access rights assignment).

CURE's new functionality for the so-called visual tailoring and navigation has been complemented by new forms of visualizing synchronous awareness information and supporting communication. However, this work highlights crucial impact of navigation in collaborative systems which was recognized e.g. in the BSCW system [1], too.

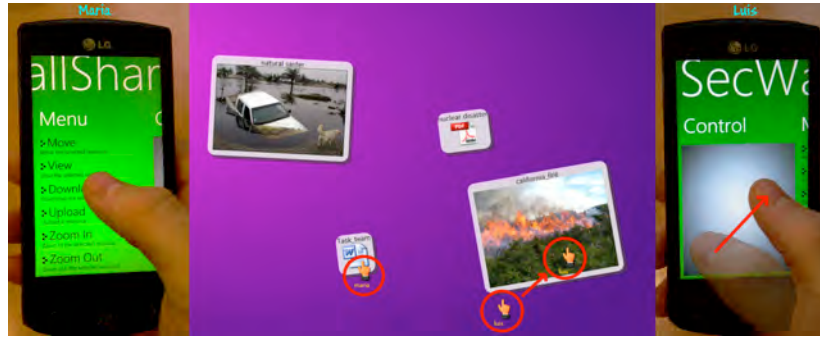


Figure 1: Controlling the shared surface

Related to fine-grained access rights assignment, the work of [2] shows the potential of using proof-based credential systems like Idemix to enhance the usability of privacy-preserving social interaction in collaborative settings such as transparently performing authorization in the background e.g. without any user intervention at the level of the user interface.

The novelty of the presented work here consists of (1) providing a collaborative system supporting distributed user interfaces (DUIs) for multi-user parallel interaction (i.e. multi-pointer) by using mobile devices of the involved people and (2) by avoiding navigation problems (common shared pool) as well as (3) fine-grained access rights for DUIs according to the roles in the background. To our best knowledge we are not informed about such a system which at least is used for enhancing decision making in mission-critical situations as required in the context of ReSCUe IT project and as we present in the following.

APPROACH

The Idemix Enabled WallShare DUI

WallShare [7] is a client-server system that allows users to collaborate in decision making meetings by the means of a DUI. WallShare provides a multi-pointer and collaborative system that enables users to interact with a shared surface (a big screen, or a projection on the wall) using their mobile devices. Users are represented in the shared surface with cursors that can be moved by the means of dragging gestures performed on the touch screens of their mobile devices. This effect is shown on the right mobile phone in Figure 1.

The shared surface is used to present resources to users. Users are able to download these resources from the shared surface to the mobile device when they locate their cursor over the resource they want to download, and then clicking the download menu item in the mobile device. The menu is shown on the left mobile phone in Figure 1. Users are also able to upload resources to the shared surface just by clicking the upload menu item on the mobile device and selecting the resource they want to upload.

In order to support the decision making process at different physical locations we have extended the WallShare system.

Therefore, the distribution of the user interface takes place at two levels: at the same physical space where the participants of a meeting interact with the shared surface using their mobile devices at the same room, and at different physical spaces where many collaborative meetings are carried out at the same time in different physical spaces (buildings, emergency rooms, etc.).

Different views of the information are provided to users according to the physical location of the meetings. The view of the information to be displayed in the shared surface at the police department is not the same as the view of the information to be displayed at the National Government, or the Journal Building.

Besides, another dimension to be taken into account is the way the information is manipulated because it depends on the role each participant is playing in the meeting (the governor, fire department chief, fire fighters, policemen, etc.). To manipulate information that depends on the user role, we use the mobile device that provides us a private and personalized view and control over the shared information. Thus, our proposal to extend the WallShare system provides different views of the information according to the location of the group of professionals that are using it; besides it allows different ways of manipulating this information according to the role each participant plays in the meeting (see Figure 2).

For role-related and fine-grained shared artifacts access, Idemix represents a cutting-edge anonymous credential system, which enables to perform anonymous authentication between users and/or service providers and as well supports accountability of transactions [3]. An Idemix credential is obtained from an issuing authority, attesting attributes to the users such as access rights and allows for various protocols and mechanisms cited in standard literature (i.e. property proofs, usage limitation, revocation of credentials or verifiable encryption). During the issuance protocol the user and a certificate authority (CA) interactively create a credential.

This credential is signed by the issuing CA with its private key, so it can easily be verified using the issuer's public key. It also contains the user's pseudonym to bind the user's master secret to the credential.

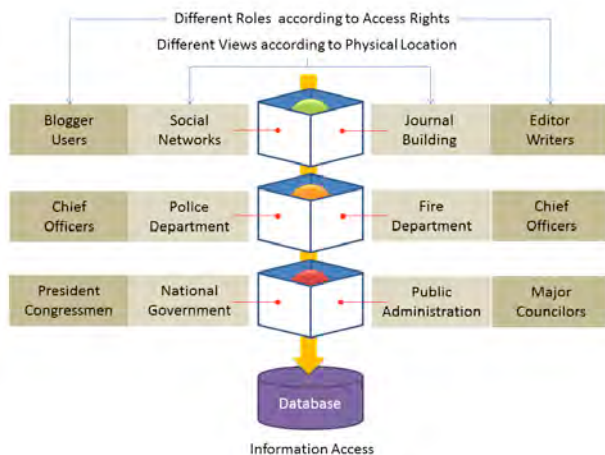


Figure 2: Conceptual design for a system supporting secure distributed user interfaces

In contrast to other privacy enhancing technologies, which send (pseudonym) certificates to a given verifier, Idemix based solutions only send proofs (such as "employee of retailer" or "working in the logistics industry") and allows for performing authentication and authorization transparently in the background as shown in [2].

CONCLUSIONS AND FUTURE WORK

This paper presents a system offering a shared workspace that enables experts (i.e. policemen, firefighters, doctors, engineers, etc.) to call simultaneous meetings at different geographical locations (i.e. police department, fire department, hospitals, universities, etc.), to support the decision making process in mission-critical situations (natural disasters, accidents, food supply chain problems, etc.) ensuring multi-lateral security and privacy.

The system provides users with different views of the information according to the groups of experts that are part of the meeting. Besides, the system functionality offered to the users varies according to the role they play in the meeting.

The combination of an anonymous credential security system with a multi-pointer collaborative system provides users with easy way to share the access of information in secure way ensuring the privacy.

Consequently, different groups of experts are able to make decisions based on different views of the same situation simultaneously. To accomplish this goal, experts of different fields are able to exchange information at different physical locations in real time just by manipulating a shared surface where they can easily manipulate the resources on

it.

As future work, we are developing adaptability techniques to customize view according to the group of experts in the meeting, as well as to customize the access rights for each role in the meeting.

The SecWallShare client is actually developed for Microsoft Windows Phone 7 platform: however, we are actually working in the development of the client for other platforms, such as iPhone, Symbian, Android, etc.

ACKNOWLEDGMENTS

Parts of the work presented here is funded within the joint project ReSCUe IT supported by the German Federal Ministry of Education and Research (BMBF) and the French L'Agence nationale de la recherche (ANR) under grant #13N10964. This research has also been partially supported by the Spanish CDTI research project CENIT-2008-1019, the CICYT TIN2008-06596-C02-0 project and the regional projects with reference PPII10-0300-4174 and PII2C09-0185-1030.

REFERENCES

1. Appelt, W. and Mambrey, P.: Experiences with the BSCW shared workspace system as the backbone of a virtual learning environment for students. In Proc. of ED-MEDIA99, 1999.
2. Bourimi, M., Heupel, M., Kesdogan, D. and Fielenbach, T.: Enhancing Usability of Privacy-Respecting Authentication and Authorization in Mobile Social Settings by Using Idemix. Technical Report. 2011. http://www.uni-siegen.de/fb5/itsec/publikationen/dime_technical_report_idemix.pdf.
3. Camenisch, J. and Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system. In CCS '02: Proceedings of the 9th ACM conference on Computer and communications security, pp. 21–30, New York, NY, USA, 2002.
4. Lukosch, S. and Bourimi, M.: Towards an Enhanced Adaptability and Usability of Web-Based Collaborative Systems. Int. J. Cooperative Inf. Syst. 17(4): 467-494, 2008.
5. Maybury, M. and Wahlster, W.: Readings in Intelligent User Interfaces (Interactive Technologies). Morgan Kaufmann; 1st edition, 1998.
6. ReSCUeIT: Robust and secure supply-chain supporting IT. <http://www.sichere-warenketten.de/>
7. Villanueva, P. G., Tesoriero, R. and Gallud, J.A.: Multi-pointer and Collaborative System for Mobile Devices. Proc. of Mobile HCI'10, pp. 435- 438, ACM, 2010

Co-Interactive Table: a New Facility Based on Distributed User Interfaces to Improve Collaborative Meetings

Elena de la Guía

Computer Science Research Institute
University of Castilla-La Mancha
Albacete, Spain
elenage84@gmail.com

María Lozano, Víctor M. R. Penichet

Computer Systems Department
University of Castilla-La Mancha
Albacete, Spain
{maria.lozano, victor.penichet}@uclm.es

ABSTRACT

Co-Interactive Table is an interactive and collaborative system based on an intelligent table that in conjunction with mobile devices and shared spaces provide collaborative tasks with a simple and intuitive gesture, thanks to the combination of technologies such as RFID, Wi-Fi and Web Services. The system implements the concept of Distributed User Interface, providing the ability to distribute the interfaces at the same time and space across multiple monitors, screens and platforms. In this way it facilitates information sharing and solves problems related to the limitations of the screens of mobile devices.

Author Keywords

Interactive table, Distributed User Interfaces, collaboration, RFID, mobile devices, interactive meetings

ACM Classification Keywords

. H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General Terms

Design, Human Factors, Experimentation.

INTRODUCTION

Group work is a fundamental human activity and collaborative environments provide good tools to facilitate this process. According to [7] the concept of collaborative applications or groupware is defined as software systems that are designed to facilitate common tasks to a group of users. From this concept arose the discipline that is responsible for guiding the analysis, design and development of collaborative systems, notably in 1988 when Grief and Cashman coined the term Computer Supported Cooperative Work (CSCW) [3].

Meetings are a good exercise enhancing integration and communication among users. In addition, it is highly effective to increase users' creativity, thus yielding better

and more satisfactory results. However, this task is very complex and it needs a lot of information and technological tools to support it correctly.

Nowadays it is evident that user interfaces (UIs) present a significant barrier to the collaborative work. Physical and cognitive barriers suggest that the windows, icons, menus and pointing (WIMP) UIs, so prevalent in today's computing technology, are not the most appropriate for the user.

The intelligent interaction with objects can help to improve cognitive processes in an intuitive way for the user. Technology from the perspective of distributed cognition cannot be seen as a substitute for cognitive processes, but as a complementary element which can help to improve attention, perception, memory or thought adapted to different contexts and goals [5][6].

To take advantage of distributed cognition the system uses distributed user interfaces (DUI). The modeling and design of DUI aims to facilitate the complexity to handle user interaction with heterogeneous devices in mobile environments and in a dynamic way. The development of new devices that support the proposed DUI models provide users with an interface divided into different devices and introduce new possibilities for interaction and collaboration [1,2,4,10].

There are interesting solutions that implement interactive system using mobile devices and RFID as we can see in the literature [8], [9], but these systems do not support face-to-face collaboration, which is an important factor to take into account in meetings rooms.

This proposal aims to address the weaknesses that arise in work meetings. To really help users in these scenarios it is necessary to find an interaction style that requires virtually no attention from the user. This way the user focuses just on the topic of discussion and not in how to use the system. We aim to provide a system that is easy to use and very intuitive. This system has been implemented as an interactive table based on RFID technology, which allows the participation of different users in a simple way through natural gestures.

CO-INTERACTIVE TABLE

In this section we describe how the co-interactive table is used to facilitate collaborative tasks in a meeting room.

Description system

The interactive table is composed of different panels. The size of each panel is 210x297 mm, the same as a din-A4 paper. Its interface shows the operations that a user may perform in the meeting. These operations are graphically represented by attractive and intuitive metaphors. RFID tags that offer the functionality to the panels are hidden under the external interface

Each user has a panel and a mobile device with an RFID reader. Each mobile device has the application *Co-IT* running in it. This application shows the information needed to use the interactive table. All the devices are connected by a Wi-Fi access point to the server that stores the important data, files and methods necessary for the operation of the system. In addition, the room has a projector connected to a PC that shows the users connected in each moment and the post-it notes that they have sent during the meeting through their panels. It can also support remote meetings where users are distributed in different geographical locations connected to the same server.

As an example of use, Figure 2 shows a concrete scenario in which four people participate in a meeting using the Co-interactive table. Each participant has a panel with all the provided functionality depicted on it and a PDA with an integrated RFID reader.

The interaction mode

The interaction mode between the user and the system is very intuitive. The system can recognize and offer the service required by the user with a gesture as simple and natural as bringing the device mobile near the interactive table, as shown in Figure 1. It also ensures that the user has complete control of the application without focusing on its usage; this way the user can enjoy all the functionality that offers the system without worrying about how to use it.

The mechanism used with the interactive table is always the same, the user selects the operation to perform, if this operation is a collaborative one, then he/she chooses another user to interact with and immediately the system responds and the result is displayed on the user's mobile screen.

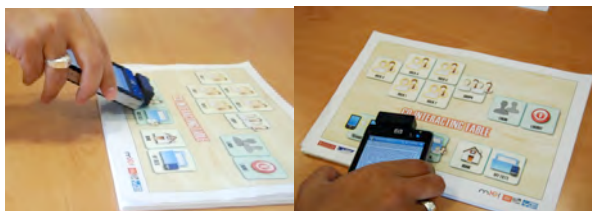


Figure 1: Natural gesture performed to execute a task by means of the interactive panel. This interaction style has been called “approach&remove”.



Figure 2: Meeting room that incorporates the “Co-Interactive Table system”.

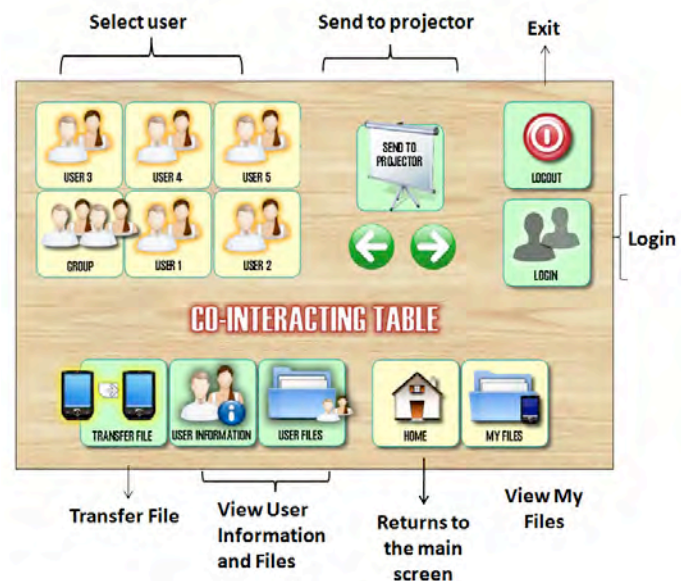


Figure 3. Interactive panel user interface showing the functions offered by the intelligent objects.

The interactive panel (See figure 3) shows visual metaphors that represent a specific operation. The available functions are the following:

- Log in: It suggests the user to log in. At this moment the system panel is associated with the user
- Transfer File: This function allows users to share files with the participants in the meetings, this way you can share and save information easily in a few seconds. Depending on the metaphor selected, the file can be shared with a particular user or with all the users in the meeting
- View User Information and Files: This metaphor shows the users' academic and professional information. This function facilitates communication in meetings where people do not know each other previously. In addition they can see the files uploaded by other users.
- Returns to the main screen: This function returns to the main screen.

- View my files: This metaphor shows the files received during the meeting
- Select user: Selecting a user is always necessary to carry out a collaborative activity. In our case it will be necessary to transfer and view files and information from another user.
- Exit: The user logs out. This function removes the association between the panel and the user.
- Send to projector: The user can send his/her interface to projector and the rest of user can see, download and use it.



Figure 4. Co-Interactive Table implementation of distributed user interfaces.

Distributed User Interfaces and Collaboration

As mentioned in the previous section, the use of DUIs in mobile environments is a very interesting point. The main objective of a DUI is to facilitate the tasks that mobile users want to make optimal configuration providing a resource for interaction in an interconnected manner. In this case the resources are interconnected interaction screens of mobile devices and screen sharing.

Mobile devices usually have a quite reduced screen. Co-Interactive Table extends the interface of mobile devices. Users can work with their mobile device very easily through the user interface offered by the panel and the shared area that shows information sent by users.

Figure 4 shows the Co-Interactive Table's vision of the distributed user interface paradigm.

The migration of a user interface (UI) is the action of transferring a UI from a device to another one, for example from a desktop computer to a handheld device. A UI is said *migratable* if it has the migration ability. We use the interactive panel to migrate the interfaces. The steps are as follows: the user interacts with the panel approaching the mobile device to the task he wants to run. Then the system clone the interface device in the projector, so all users can interact with it. The user has the possibility of sending presentations, notes and important information for the meeting to the shared workspace.

Figure 5 shows the migration from a handheld device to the projector when the user brings the mobile device near the metaphor "Send to projector".

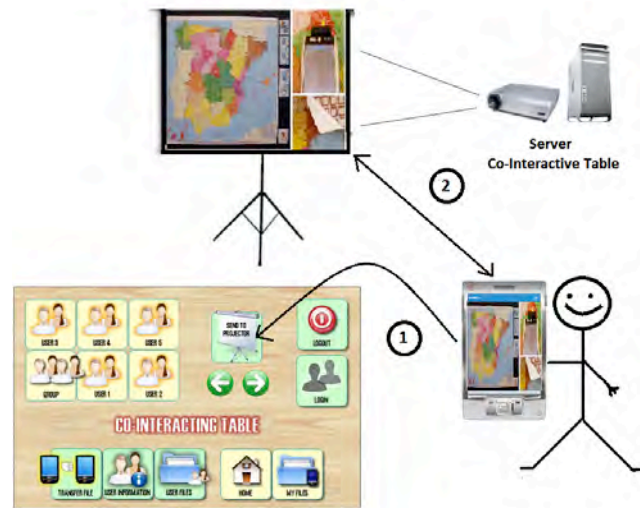


Figure 5. The user in (1) has a presentation open in his/her mobile device. The user brings near the device to the metaphor called "Send to project". Instantly the projector shows the same presentation that the user has in his/her mobile devices. In this way the interface mobile device is distributed in the projector (2).

System Architecture

The system is a client-server system (see Figure 6). The client system runs on the user mobile device. It is connected to the server application through a wireless network and it is communicated with the interactive table via RFID. When the RFID reader in the mobile device is approached to the chosen metaphor, the RFID tag is excited by electromagnetic waves sent by the RFID reader, then, the component controller, explained below, sends the identifier to the server. The server maps this information in the database and executes the steps necessary to return the information to the mobile device. The applications running in the server are shown in the projector. In addition, the system includes a webcam to communicate via conference with other users and share files and information remotely.

Solution Benefits

The main advantages of the system are the following:

- Implementation of new intuitive interfaces for any mobile device which allows a simple interaction. This also ensures that the learning process is more effective, providing motivation for the user to use the system.
- Very cheap to deploy. Mobile devices will incorporate RFID technology in the short term and passive RFID tags are very inexpensive, offering the panel at a very economical and affordable cost.

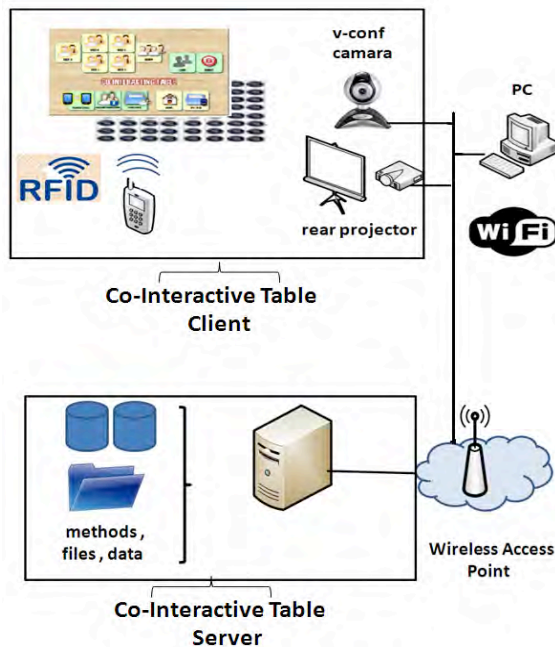


Figure 6. System Architecture

- Scalable. It provides the possibility to extend the functions without losing quality. To include new functionality we just need to add new RFID tags to the panel.
- Greater flexibility in the system dynamic content. The application can update the information quickly and effectively, thanks to the existing interconnection network.
- Improving the collaborative tasks, essential in any activity that requires group work. In this case the main objective is to improve communication in meetings and facilitate the sharing of files among different users.
- Implementation of simple functions with natural gestures. It also ensures that the user has complete control of the application without focusing on the usage of the system, just doing simple natural gestures such as bringing near the mobile device to the metaphors prepared for that.
- Panel easy to use and move anywhere. Thanks to its small size, the user can carry the same panel to be used in different meetings. This panel provides the advantage of being used anywhere, without losing functionality.
- The system offers the possibility to perform remote meetings.

CONCLUSIONS AND FUTURE WORKS

This article describes the implementation of Co-Interactive Table, a collaborative application designed to facilitate meetings where users can share, through natural and intuitive gestures, files and information. To implement the

system, we have used several technologies, including mobile devices and RFID. In addition we make use of distributed user interfaces in order to distribute cognition easily in different devices, monitors and platforms.

ACKNOWLEDGMENTS

This research has been partially supported by the Spanish CDTI research project CENIT-2008-1019, the CICYT TIN2008-06596-C02-0 project and the regional projects with reference PAI06-0093-8836 and PII2C09-0185-1030. We also would like to thank M^a Dolores Monedero for her collaboration in this project.

REFERENCES

1. Bharat, K.A., Cardelli, L.: *Migratory Applications Distributed User Interfaces*. In: Proc. of ACM Conf. on User Interface Software Technology UIST'95. ACM Press (1995) 133–142.
2. Coutaz, J., Balme, L., Lachenal, C., Barralon, N.: *Software Infrastructure for Distributed Migratable User Interfaces*. UbiHCISys 2003 Workshop on UbiComp 2003.
3. Crowe, M.K. "Cooperative work with multimedia", Springer-Verlag, 1994
4. González, P., Gallud, J.A., Tesoriero, R. WallShare: A Collaborative Multi-pointer System for Portable Devices. PPD10: Workshop on coupled display visual interfaces. May 25, 2010: Rome, Italy.
5. Harnad, S., (2005) Distributed processes, distributed cognizes, and collaborative cognition, *Pragmatics&cognition*, 13, 501-514.
6. Hollan, J., Hutchins, Kirsch, E. D.. *Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research.*, ACM Special Interest Group on Computer-Human Interaction, Vol. 7, No. 2, June 2000, 179.
7. Peter y Trudy Johnson-Lenz, *Consider the groupware: Design and Group Process Impacts on Communication in the Electronic Medium*, 1981
8. Tesoriero, R., Tébar, R., Gallud, J. A., Penichet, V. M. R., Lozano, M.: *Interactive EcoPanels: Paneles Ecológicos Interactivos basados en RFID*. Proceedings of the IX Congreso Internacional de Interacción Persona-Ordenador Interacción 2008. ISBN: 978-84-691-3871-7; pp 155-165.
9. Tesoriero, R.; Gallud, J. A.; Lozano, M. D.; Penichet, V. M. R.: *A Location-aware System using RFID and Mobile Devices for Art Museums*. Fourth International Conference on Autonomic and Autonomous Systems. IEEE/CS Press. IARIA-ICAS 2008. Páginas 76 - 81. 2008.
10. Vandervelpen, C., Coninx, K.: *Towards model-based design support for distributed user interfaces*. In Proceedings of the third Nordic Conference on Human-Computer Interaction, pages 61–70. ACM Press, 2004.

Improving Ubiquitous Environments through Collaborative Features

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
{maria.lozano, victor.penichet}@uclm.es

ABSTRACT

Ubiquitous environments are a main objective for new technologies. In this paper, we propose to improve ubiquitous environments through collaborative features: collaboration, cooperation, coordination, communication, information sharing, awareness, time, and space. The proposed model will allow users to work in an environment where collaborative applications will adapt its behavior depending on the environment state: user tasks, used and free resources, who is working, user location, etc. Applications need to incorporate a distributed interface in order to allow users to run the same application in different devices and in different situations.

Our proposal could be applied in real-life scenarios as Residential Care Homes. The integrative proposal can help by reducing human errors and optimizing tasks and information management. The application will give them the appropriate functionality according to the tasks they should perform and the environment state. Hence, employees will be able to focus their attention only on their tasks.

Author Keywords

Collaboration, Ubiquity, context-awareness, awareness, context, healthcare.

ACM Classification Keywords

H5.3. Group and Organization Interfaces.

General Terms

Design, Management.

INTRODUCTION

Currently, computer users can make use of a wide range of systems. One main reason is the evolution of technology, which continuously tries to be automatically adapted to the

user needs. Another reason is the huge number of developers involved in the use of technology.

During the last years, the need of collaboration has been an exploited topic. Complex objectives usually need to join efforts, which implies reaching faster solutions and less individual effort. Another exploited topic has been the use of technologies anywhere, anytime: ubiquitous computing. Additionally, the last topic implies that applications will automatically adapt their behavior to the user needs and environment conditions.

In this paper, we present an integrative proposal where ubiquitous environments are improved through collaborative features. We want to approach the possibility of enjoying actual technologies anywhere during the use of systems with applications which adapt their functionality to the users needs and allows them to collaborate. Current technology allows the use of different types of devices (PDA, iPhone, Mobile Phone PC or laptop), so that they can use the same application in any device. This is possible by using a well-defined distributed interface.

The rest of the paper is organized as follows: Our proposal and a brief explanation of its foundations are presented in Section 2. Section 3 presents a case study to provide a better understanding. Finally, conclusions and future work are presented in Section 4.

COLLABORATION AND UBIQUITY INTEGRATION

Collaborative Environments

Human-Computer Interaction (HCI) is a research field concerning the interaction of the users with the computer applications they use. Its aim is to improve such interaction by developing new user interfaces, which can make software systems more natural to the final user. The evolution of the working style of humans and the improvements in computer technologies, in terms of communication and interaction mechanisms, have deeply altered the classical concept of HCI towards a new HCHI (Human-Computer-Human Interaction) concept due to the necessity of working in collaboration with other people. These changes have produced what it is known as Computer-Supported Cooperative Work (CSCW) [3, 4, 5, 6, 8].

In order to get high quality groupware applications, special features regarding CSCW should be taken into account from the requirements gathering stage. Such special features constitute the CSCW basis. (1) First, *cooperation* and *collaboration* whose differences are explained in some works such as [5], where Jonathan Grudin discusses those differences. It is assumed on the one hand, that small groups of users, who share key objectives, cooperate. And on the other hand, big organizations, where objectives and goals, usually differ or even come into conflict, collaborate. (2) *Coordination* implies harmonizing media, efforts, and so forth, to perform a common task. (3) *Communication* could be understood just as it is defined in the Wikipedia, as another collaborative Web phenomenon: “the process of exchanging information usually via a common system of symbols” [11]. (4) *Information sharing* [9, 10] occurs when two or more actors carry out cooperative or collaborative group tasks, because there is a common access to the system resources in the group process. (5) *Time* [7] refers to synchronous or asynchronous tasks. (6) *Space* [6] specifies where the users of the groupware system are physically located. The human-computer-human interaction could take place in the same place or users can work “together” in the distance. (7) Finally, *awareness* will be approached in the next subsection.

Ubiquitous Environments

Nowadays, the development of applications is growing at a fast pace due to the possibilities offered by the evolution of technology. This evolution has turned ubiquitous environments (*where it is possible to access to information anytime, anywhere and any circumstance*) into reality.

Context-aware applications are one of the most important and useful applications in ubiquitous environments. Users can pay attention only to their tasks as they do not need to know completely how to operate the application and the device. Users need an application which is able to run automatically by itself. Applications may provide users with the correct information depending on the context, adapting its behavior to the environment.

Context-aware applications combine two important concepts: *context* and *awareness*. Context is “*any information that can be used to characterize the situation of entities (people, places or objects), which are considered as outstanding for the application-user interaction including both*” [1]. It implies that applications may not ignore the conditions around them. Awareness is “*knowing what is going on*” [2]. Therefore, users need the information that will help them know what is happening around them, i.e. actions performed by other users, used resources or who is working.

Integrative Proposal using Distributed User Interfaces

We have detected an important gap that involves collaborative and ubiquitous environments. Ubiquitous

environments have usually centered their efforts to consider only one user who interacts with his/her environment. Thus, the user tasks are mainly individual, minimizing the possibility to collaborate.

Our proposal implies the application of collaborative features to ubiquitous environments. Thereby, users will be able to use collaborative systems in ubiquitous environments. Applications will adapt their functionality or features depending on the context. Users will continue depending on the technology but with a lower degree of dependence. Also, the user experience will be more natural and comfortable: (1) The system will show appropriate functionality and features according to the user environment and it won’t require any special skill. (2) Users will be able to collaborate achieving objectives. Collaborative possibilities will depend on the environment conditions. (3) Automatic functionality adaptation and possibilities of collaboration will be useful in an important number of real environments: hospitals, real care homes, universities, etc.

Collaborative ubiquitous environments entail important implications, which are complex when working with only one user who interacts with the system. Then, the system complexity increases when it has a set of users who also collaborate. The main reason is that users need to be aware of what is happening in their environment. Therefore, collaborative applications have to take into account what is happening around them as well as any information about the rest of the users.

The integrative proposal needs to include a distributed interface in order to allow users to use different devices in the environment. Users have to be able to access to the needed information in the environment from any possible device. Hence, the distributed interface will adapt its appearance based on the device: PDAs, iPhones, Mobile Phones, PCs or Laptops.

The next section describes the integrative proposal using a specific case study. The case study will clarify new benefits of the environment and its features through a possible real case.

A CASE STUDY: RESIDENTIAL CARE HOME

We have selected a Residential Care Home (RCH) as the place to apply the proposed environment. In RCHs, employees may have difficulties to do sensitive tasks under pressure. The proposed system can guide users by reminding each step of the current protocol. The main reasons why we selected RCHs are the following:

- Employees must control a set of task protocols. Each protocol needs a high level of collaboration

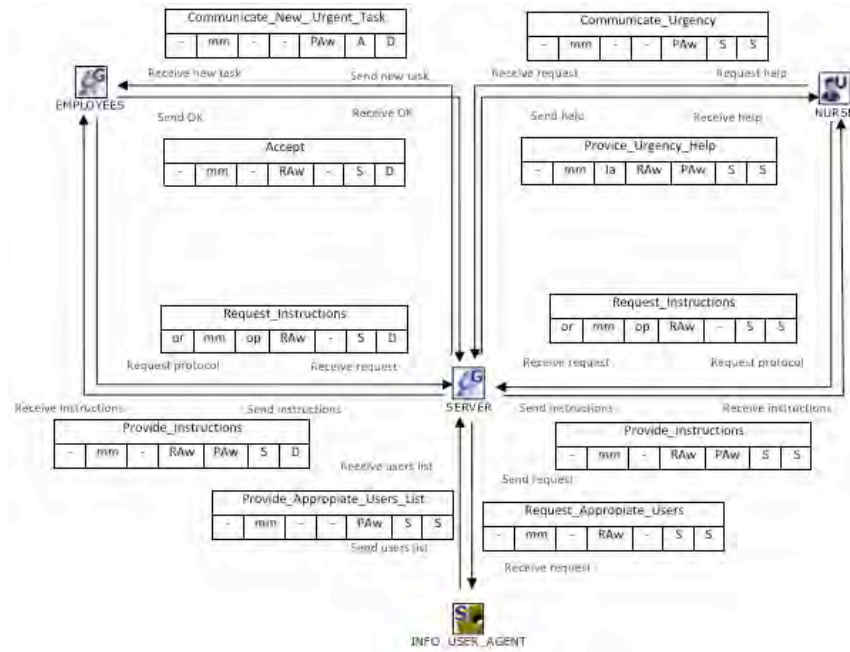


Figure 1. Case study Co-Interactive Diagram

among employees. Therefore, RCHs require a management system with collaborative features.

- Ubiquitous environments facilitate RCH management. Employees will be able to access any required information or functionality at any time and anywhere. In their devices, they will always have functionality available depending on the tasks to do and their location. Additionally, the devices have collaborative functions available depending on the environment of the colleagues.
- The system can reduce the number of human errors. For example, the system will remind important tasks.

Distributed Ubiquitous Context-Aware Interface

Users will be able to use several devices depending on their location. The devices available will be PDAs, iPhones, Mobile Phones, PCs or Laptops. Employees will use the same application in different devices. Each application may adapt its appearance to the device in use. Therefore, a distributed interface is needed in order to offer the correct visual state when an application identifies the device where it will be executed. Thus, two aspects have to be considered: platform and information.

Different platforms appear when users use different types of devices. Each platform implies its own development considerations, so the application interface needs a well-defined distribution with different appearances based on the possible platforms. In this way, our proposal includes the necessity of distributed interfaces because users will use the correct and more useful device depending on the environment (location and task to be done).

Information is an important attribute because employees always need a specific range of data. That information allows users to carry out their tasks. Then, each interface distribution will always show the same type of information with the most similar appearance, maintaining the main platform restrictions.

Example Scenario

The description of an example scenario is the best way to see how our integrative proposal can help the management of a RCH.

A nurse, named Mary, is checking a room where there are some residents. Suddenly, a resident falls on the floor. Mary selects the urgency functionality in its mobile device application. The application offers Mary two options: (1) To ask for advice to the appropriate colleague. (2) To ask for advice to a specific colleague. Mary selects the first option because she is so nervous. The application sends the request to the server. The server searches the appropriate list of employees through user information agents and a sensors system. The agents will give the server information about users near Mary whose tasks can wait. The sensors read identification devices (e.g. when a user starts or finishes his/her working day) and send compiled information to the Server. All the information allows to detect who is in the system and where. The system sends a message to the helper employee when Mary selects him/her from the list offered by the application. A nursing assistant, named John, will be the helper. John receives the message with the information about the new task. John uses the map offered by the application to go to the location of the urgency. That information is essential when a RCH has huge dimensions.

The described scenario shows how our integrative proposal can be applied in real life. The scenario shows collaborative aspects: a nurse uses the system to ask for help because a resident has fallen down; a nursing assistant comes to help in an emergency task. Additionally, the scenario shows ubiquitous aspects: the system knows who is at work, their current tasks and locations at any time. Therefore, the system can offer users the appropriate functionality in each moment based on their environment state.

Figure 1 shows the modeling of the interaction among the users in the described scenario. It exhibits a co-interaction diagram (CD), which comes from the process model TOUCHE [12]. We have selected TOUCHE because it offers the elements that we need for our collaborative ubiquitous environment. Each CD rectangle is a group task characterize by its name and CSCW features: coordination, comunication, cooperation or collaboration, request awareness, provide awareness, task to be performed in the same or different places and finally, the task can be syncronous or asynchronous. The original CD has been modified including awareness in the group task definition. The main reason is that awareness is obtaining a remarkable importance in the studied environments. The new version allows indicating that a task needs awareness (Request Awareness, RAW) or provides awareness (Provide Awareness, PAW).

Now, a group task of the diagram above will be described: *Communicate_Urgency*. It is a group task that describes that a nurse (represented as a user) communicates her situation (resident has fallen and needs help) to the system (represented as a group). Specifically, the task provides awareness (PAW) because it gives information about the urgency, that is, about what is happening. The rest of the task among the CD works in a similar way.

CONCLUSIONS AND FUTURE WORK

In this paper, an integrative proposal improving ubiquitous environments through collaborative features has been presented. The proposal consists of an environment where users can use applications that adapt its functionality depending on the environment conditions; also, users can collaborate with others in order to complete their tasks. The application needs to use a distributed interface in order to be adapted to any type of device used by the employees.

A case study is presented to illustrate where the proposal can be applied in a real scenario, a RCH. We chose a RCH because: (1) It is an environment where users need to follow collaborative protocols. (2) Ubiquitous environments facilitate the management of the RCH. (3) The system can reduce human errors.

In the future, we are considering to work on the creation of a prototype in a RCH. Also, we want to carry out different experiences with the prototype to check how it facilitates the user tasks.

ACKNOWLEDGMENTS

This work has been partially supported by the Spanish CDTI research project CENIT-2008-1019, the CICYT TIN2008-06596-C02-0 project and the regional projects with reference PAI06-0093-8836 and PII2C09-0185-1030.

REFERENCES

1. Dey, A., Abowd, G. D., & Salber, D. (2001): A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16, 97-166.
2. Endsley, M. (1995): Towards a Theory of Situation Awareness in Dinamic Systems. *Human Factors*, 37(1), 32-64.
3. Greenberg, S., 1989. The 1988 conference on computer-supported cooperative work: Trip report. *ACM SIGCHI Bulletin*, 21(1), pp. 49-55, July.
4. Greif, I., 1988. *Computer-Supported Cooperative Work: A Book of Readings*. Morgan Kaufmann, San Mateo CA.
5. Grudin, J., 1994. CSCW: History and Focus. *University of California. IEEE Computer*, 27, 5, 19-26.
6. Horn, D. B., Finholt, T. A., Birnholtz, J. P., Motwani, D., Jayaraman, S., 2004. Six degrees of Jonathan Grudin: a social network analysis of the evolution and impact of CSCW research. *ACM conference on Computer supported cooperative work*, pp 582 – 591.
7. Johansen, R., 1988. *Groupware: Computer support for business teams*. New York: The Free Press.
8. Johnson-Lenz, P. and Johnson-Lenz, T., 1981. Consider the Groupware: Design and Group Process Impacts on Communication in the Electronic Medium. In Hiltz, S. and Kerr, E. (Ed.), *New Jersey Institute of Technology*.
9. Poltrock, S. and Grudin, J., 1999. CSCW, groupware and workflow: experiences, state of art, and future trends. In *CHI '99 Extended Abstracts on Human Factors in Computing Systems* (Pittsburgh, Pennsylvania). *CHI '99*. ACM Press, New York, NY, 120-121.
10. Poltrock, S. and Grudin, J., 2005. Computer Supported Cooperative Work and Groupware (CSCW). In *Interact 2005*. Rome, Italy.
11. R. Penichet, V. M., Lozano, M. D., Gallud, J. A., Tesoriero, R.: Requirement-based Approach for Groupware Environments Design. *Journal of Systems and Software (JSS)*. ISSN: 0164-1212. DOI: 10.1016/j.jss.2010.03.029.
12. R. Penichet, V. M., Lozano, M. D., Gallud, J. A., Tesoriero, R. (2009): User Interface Analysis for Groupware Applications in the TOUCHE Process Model. *International Journal Advances in Engineering Software (ADES)*, ISSN: 0965-9978, pp. 1212-1222

Drag&Share: A Shared Workspace for Synchronous Collaboration

Félix Albertos Marco
Victor M.R. Penichet

Instituto de Investigación en informática de
Albacete (I3A)
Universidad de Castilla-La Mancha
Felix.Albertos,Victor.Penichet@uclm.es

José A. Gallud

Miguel Hernández University
Avda. de la Universidad s/n,
(03202) Elche (Alicante), Spain
jgallud@umh.es

ABSTRACT

In this paper we present a new web application (Drag&Share) to share resources among different computers using the distributed user interfaces paradigm to easily share information for synchronous collaboration. It provides a shared workspace as an extension for the local system into the network, allowing users to communicate and to share documents in real time. The application works in an internet browser, without any plug-in. It does not need to be installed on the user's system, and it is platform independent.

Author Keywords

Shared workspace, synchronous, collaboration, websockets, distributed user interface, real time

INTRODUCTION

We need to share information when performing collaborative tasks. Verbal communication is another important mean used for coordinating and providing extra information about the work we do. Real time communication mechanisms are necessities in order to improve communication, so users can solve the problems in the most effective possible way.

Simple tasks such as sending and receiving documents between multiple users may become problematic. Problems may come by various factors, including network infrastructure, devices and platforms involved in the process and the experience and user knowledge. In some cases it is necessary to install specific applications or plug-ins in the internet browser. May be these complements are not available for the platform used. Sometimes, even been available the necessary complements, users may not have the necessary privileges to complete the. Another problem may arise is the user's lack of experience in the operation of the program or lack of the required features in the program.

There are many commercial applications and research proposals for users to perform the aforementioned tasks, e.g. Windows Live Mesh 2011 [1] or Dropbox [2]. Both of them allow users to synchronize documents between computers. Mesh does not allow users to communicate in real time and do not give information about the tasks

performed by the other users. It is recommended for asynchronous document sharing. Mesh requires the installation of proprietary software, available on a limited number of platforms. In a similar way, Dropbox let the users synchronize folders and share documents across multiple computers, but users also are required to install the application. Dropbox does not show information about what users are doing with shared documents. Even the popular Skype [3] allows the coordination of tasks and allows users to send files and to communicate in real time. However there is no information about the actions performed by the other users with the documents.

Some research proposals such as CE4Web [4] or WallShare [5] provide users with synchronous collaboration to perform a specific task. WallShare provides a multi-pointer system allowing users to share documents in a synchronous environment. However, most applications need to be installed, or need a plug-in in the navigator, or are limited to specific platforms.

To solve the underlying problems in carrying out collaborative tasks, work habits lead users to use means that slow down the process of sharing documents. The use of physical media, such as USB sticks, complicates the process and it is not feasible when the number of users increases. Sending emails is not the ideal solution because it requires knowledge of the addresses of those involved and the forwarding of documents to users who join to carry out the task once the mail has been sent. Our proposal provides real-time communication, improving the communication process and giving persistence to documents being accessed by users who connect later.

Taking advantage of the latest trends and technologies in software development, we use the web as an application platform. This allows us to develop the entire system based on Web technologies.

The rest of the paper is organized as follows. First we briefly describe some trends on user interface design. Next sections describe the proposed solution and the distributed user interfaces. Then the architecture used in developing the system is shown. Finally, benefits of the solution and conclusions and future work.

TREND ON USER INTERFACES DESIGN

Nowadays, the main proposal regarding design of user interfaces is UsiXML [6] which is based on the Cameleon Project [7]. Interactive applications with different types of interaction techniques, modalities of use, and computing platforms can be described in a way that preserves the design independently from peculiar characteristics of physical computing platform [8].

On a similar way, software applications are expected to be used on different platforms. Users look for user interfaces which are familiar, intuitive, and easy to use. Designers should develop for the user (user-center design) looking for increasing the user experience and the usability of the system.

Following such trends, the approach presented in this paper shows a tool where users can directly interact among them. No extensive knowledge is needed to use it effectively: appropriate metaphors, known elements and common file browser actions are the base. The tool is platform independent since may work on any Internet browser. It is not necessary to install an additional plug-in or any software, so it may be used in a familiar way for the user.

SYSTEM DESCRIPTION

The proposal presented in this paper, allows real time communication among users. It is an extension of the desktop through a real-time shared workspace [11, 12] where users can share all kinds of documents. It also keeps them informed in a continuous way about the actions undertaken by participants in the system.

User just needs a web browser to run the application and connect to the system's URL. Then, the user provides his username. This username will identify the user into the whole system, both the Multi-pointer system and the chat.

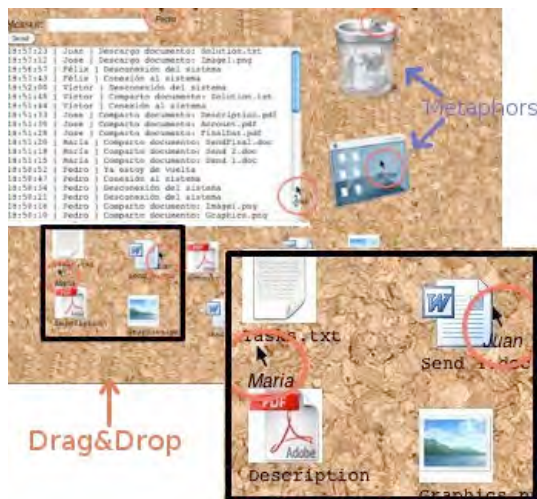


Figure 1. Shared workspace with multi-pointer system

Once the user has connected, he can view the shared workspace (figure 1). All the users have the same vision of the shared workspace. To disconnect from the system the

user only need to close the internet browser. Through this event, the system handles the disconnection, eliminates Multi-pointers from the shared workspace, and notifies the system.

Shared workspace

In the shared workspace are represented all the elements that conforms the system. Within this area users can interact to each other and with the shared documents. Shared documents are represented by icons. Depending on the document type represented, the object takes the appropriate icon. Under the object is the document name.

Users are represented in the system through the multipointer management system.

Multipointer management

Pointer objects represent users in the system. Consists of the mouse cursor, below which is the user's name. Telepointer object's movement is showing to all users in the system, seem the action performed with it. Pointers allow users to Drag&drop documents.

Drag&drop inside the application

The manipulation of the shared documents is performed in the same way the user manipulate the elements in a classical GUI. In systems with no mouse, the user just has to use the host method to manipulate objects. The movement of a document allows users to move it on the shared workspace, changing its position. All users in the system show the movement performed over the documents.

Drag & Drop between the desktop and the application

The Drag & Drop area is the total area of the browser's window. This window can take any size, without affecting system's performance or element's position in the area. To share a document, the user simply drags it from any window in his device, or from the desktop, to this area. This action is performed making an identical gesture as would be done if the user has to move the document into his system.

Once the user releases the document within the shared Drag & Drop area, the system places document in the exact location where the user drops the document. It is allowed to drag into the area groups of documents to share them. The implementation of the action takes place in the same way as in the host system, selecting as many documents as the user wants, and dragging them into the Drag & Drop area.

Documents have persistence in the system, so users can interact with it regardless of when the documents were shared. The position of the documents will be consistent at all times to all users. When a user log in, the system displays the document in the same position as it was left, by him or by other user.

Visual metaphors

Metaphors represent actions the user can do with documents. To perform the action represented by the

metaphor the user simply drags the selected document in it. At the moment the user places the object over the metaphor, it will show the action it performs. If the user wants to perform the action, only have to release the document over the metaphor, performing the appropriate action.

Resources downloading

Desktop metaphor allows the user to view a document, only dragging the document to this metaphor. Once the user drop the document in the metaphor, it will open the document. Then, the document will be place back in the location from which the user started the drag gesture. The document also can be opened by double clicking on it. Users who use IOS platforms, like iPhone and iPad, may open the document by a simultaneous double touch over the object.

Resources deleting

Delete metaphor deletes a document from the system. Once the action is performed, the system prompts the user if he is sure to delete the document. Keep in mind that the deletion process is not reversible. If the user agrees to delete the document, it disappears from the shared workspace. If the user does not agree to delete de document, it goes back to the location in the shared workspace from which then user started the drag gesture.

Immediate feedback

Users are informed immediately about the actions performed by others users through multi-pointer system and chat utilities. This chat is visible on demand, hiding when it is not necessary. All participants will receive the messages send. This chat does not show only users messages. It also shows information related with the system such as users login and logout, documents removing and documents visualization.

Chat information includes the date and the user performing the action, in order to keep track of who is involved in the system and what actions it has performed. Chat messages are stored in a log, so users can see the information related with other sessions.

IMPLEMENTING DISTRIBUTED USER INTERFACES IN DRAG&SHARE

Through our proposal is possible to share a workspace with other users. Users are allowed to see the interaction process from another point of view (figure 2). To achieve this, the system includes two additional views. Both works in real time, like the main window. They show the same information for different purposes, in different ways. The first of them, allows users to view the shared workspace, without the possibility to interact with the system. It can be used as a reference for users or non-users of the system. This view can be projected on a common screen. The second view provides a historic view of events in the system. The user can see chat conversations, actions taken over documents and records of the users login and logout. It provides a simple way to analyze historical information.

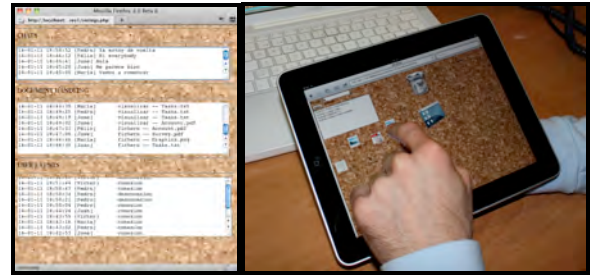


Figure 2. Distributed interfaces. Two different views.

SYSTEM ARCHITECTURE

Our proposal consists of a web application, comprising the following components:

- Web server.
- Data base server.
- Websockets server.
- Client.

Servers can be hosted on different machines. Users connect to the system through the URL that gives access to the page served by the web server. No special configuration is required to run the application on the server. The database gives persistence to documents, and chat information.

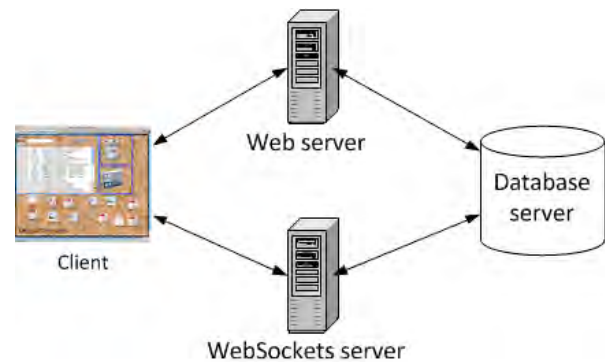


Figure 3. System Architecture

Communication between clients is managed through the websockets [9] server. Leveraging this technology, standard in HTML5 specification, the network traffic is reduced. This is due the technology used by the websockets, push instead polling. With polling technologies, client request information from the server in order to check for any changes on the system. However, push technologies allow clients to subscribe to the server. Since then the server send messages to the clients when there are any changes on the system.

The client is the internet browser. It displays the page containing the shared workspace, and communicates with the web server and the websockets server.

The system architecture allows multiple users to interact concurrently in the system, delete documents or use the chat

system. The interface always shows a consistent view with the state of the system.

BENEFITS OF THE SOLUTION

Our solution uses web technology, which does not require installing any software on the client device, or the use of plug-ins in the user's browser. It is built with standards-based technologies, such as HTML 5. These technologies can be used in a wide variety of devices.

These standards are backed by consortia and companies that support and promote their use. The application works on all types of desktops, laptops, iPads, iPhones and any device with an Internet browser in accordance with the standards of the Web [9]. User just needs a Web browser to run the application.

Using shared workspace facilitates real time collaboration among users. With the included chat, users from a wide range of platforms can communicate in real time. The multipointer system shows the username below the user local cursor in the Drag & Drop area. This feature help users when the cursor is not available, as in the case of some touch screen devices, like iPhones and iPads.

Due to the number of platforms that can use our proposal, the addition of a chat, allow users to communicate in real time through the shared workspace. This feature is important in devices restricted by size or architecture. Those restrictions may disallow some users to use an independent chat application. Therefore, is necessary to include this facility in the shared workspace.

Consistency with known applications makes very simple to share documents, without requiring a particular training. The user does not have to separate the action of sharing documents with other users with the use of the device itself.

In daily work, users realize an improvement in task such as sharing documents between users, making collaborative task, doing meetings. It also improves the transfer of documents between computers by the same user.

CONCLUSIONS AND FUTURE WORK

This proposal improves the process of document sharing and communication between users in collaborative tasks. Mainly, because the actions to be taken to share information are consistent with the actions the user uses in his host system.

Real-time interaction allows users be informed of the others participant's behavior, facilitating cooperative tasks.

The system is fully functional, but during it development and use we have detected that it could be improved by incorporating some features. Establishing a workspaces management, will allow users to access to other shared workspaces letting users to organize the information for easy processing and classification. Also, a user and group management will allow users to manage access to certain

shared workspaces to individual users or groups. It would work like a directory system.

ACKNOWLEDGEMENTS

This research has been partially supported by the Spanish CDTI research project CENIT-2008-1019, the CICYT TIN2008-06596-C02-0 project and the regional projects with reference PPII10-0300-4174 and PII2C09-0185-1030.

We thank CHI, PDC and CSCW volunteers, and all publications support and staff, who wrote and provided helpful comments on previous versions of this document. Some of the references cited in this paper are included for illustrative purposes only.

REFERENCES

1. Windows Live Mesh 2011.
<http://explore.live.com/windows-live-mesh>.
2. Dropbox.
<http://www.dropbox.com/>.
3. Skype.
<http://www.skype.com/>.
4. Víctor M.R. Penichet, María D. Lozano, J.A. Gallud, R. Tesoriero. CE4WEB: Una Herramienta CASE Colaborativa para el Modelado de Aplicaciones con UML.
5. Villanueva, P. G., Tesoriero, R. and Gallud, J.A.: Multi-pointer and Collaborative System for Mobile Devices. Proc. of Mobile HCI'10, pp. 435- 438, ACM, 2010.
6. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: USIXML: A language supporting multi-path development of user interfaces. In: Bastide, R., Palanque, P., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 200–220. Springer, Heidelberg (2005).
7. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A unifying reference framework for multi-target user interfaces. *Interacting with Computers* 15(3), 289–308 (2003).
8. UsiXML, USer Interface eXtensible Markup Language: <http://www.usixml.org>.
9. HTML5
<http://www.w3.org/TR/html5>.
10. Websockets
<http://dev.w3.org/html5/websockets/>.
11. Ellis, C., S. Gibbs, et al: Groupware: Some Issues and Experiences. *Communications of ACM* 34(1): 39-58. 1991.
12. Gutwin, C., and Greenberg, S. (2002) A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Computer Supported Cooperative Work*, 11(3-4), 411-446, Special Issue on Awareness in CSCW, Kluwer Academic Press

Distributing User Interfaces with MARIA

Marco Manca, Fabio Paternò
CNR-ISTI, HIIS Laboratory
Via Moruzzi 1, 56124 Pisa, Italy
fabio.paterno@isti.cnr.it

ABSTRACT

The increasing interest for distributed user interfaces needs languages that better support their specification in multi-device environments. In this paper we discuss an approach to specifying this type of user interfaces, and how it is applied to an extension of the MARIA language aiming to provide flexible support for the description of distributed user interfaces.

Author Keywords

Distributed User Interfaces, Model-based User Interface Design, User Interface Description Languages.

ACM Classification Keywords

H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms

Design

INTRODUCTION

One of the main technological trends is the steadily increasing number of devices per person. This has an impact on the user interface languages and technologies because it will be more and more common to interact with an application through multiple devices. A typical applications scenario is using a graphical or a vocal mobile device to interact with a large screen

This trend has stimulated increasing interest in Distributed User Interfaces (DUIs), which have been defined [2] as those interfaces whose different parts can be distributed in time and space on different monitors, screens, and computing platform, depending on several parameters expressing the context of use, such as the user, the computing platform, and the physical environment in which the user is carrying out her interactive task.

Model-based approaches [3] have often been considered in order to manage the increasing complexity derived from managing user interfaces in multi-device environments,

since each device has specific interaction resources and implementation languages to execute such user interfaces.

An interesting approach has been proposed [6] through the use of a XML-based language called HLUID (High Level UI Description), in which the user interface has a hierarchical structure and the leaves of the tree are Abstract Interactor Object (AIOs) describing high-level interactors. During the rendering process the AIOs are mapped onto Concrete Interaction Object (CIOs) associated with the current platform. In addition, they introduce a split concept for the groupings through an attribute that, when it is set to true, allows the distribution of the user interface elements without losing its logical structure.

In our case we propose a different solution, still using a model-based approach. One difference is that we support the specification at the concrete level because at this level it is easier to generate the corresponding implementations and there is a better understanding of the actual effects that can be obtained.

In the next section we describe the design concepts that characterise the proposed solution for specifying user interfaces distributed across various devices with MARIA. Our approach is not limited to graphical user interfaces but can address also devices supporting other modalities, such as voice. Then, we also provide some application examples. Lastly, some conclusions are drawn along with indications for future work.

THE PROPOSED SOLUTION

The proposed solution has been introduced for the MARIA language [5], which provides both an abstract and a set of concrete languages. The concrete languages refer to different platforms, characterised by their interaction modalities (graphical desktop, graphical mobile with touch, vocal, graphical and vocal combination). Thus, a distributed concrete user interface should be able to indicate how the user interface elements are distributed among devices, which can belong to such various platforms.

One important point to consider is at what level of granularity the distribution can occur. We have identified four possible levels:

- Presentation, which means the set of elements that characterise the possible perceivable information at a given time (e.g. a page in a Web application);

- Composition of elements, groups of elements inside a presentation that are logically related among them;
- Interactor element, single user interface elements;
- Interaction phase, we have identified three general phases that characterise interactions: prompt, input, and feedback, and it is possible to distribute the user interface even across such sub-components of one interactor. For example, entering a text through a mobile device and receiving feedback both on the mobile and a large screen device.

The next aspects that needs to be addressed is to define how distribution can occur. For this purpose we have considered the CARE properties [1], which were introduced for describing multimodal user interfaces, and have already been applied for this purpose to the MARIA language [4]. They are a simple but expressive set of properties:

- Complementarity, indicates that the user interface elements that we are considering are distributed across multiple devices;
- Assignment, indicates that the user interface elements considered are assigned to one given device;
- Redundancy, indicates that the user interface elements considered are replicated in multiple devices;
- Equivalence, indicates that the user interface elements considered can be supported by either one device or another.

The basic idea is to associate CARE properties to the various granularity levels considered. There are a few cases in which one property is not meaningful. In particular, the equivalence property mainly refers to any user input, and allows the designer to indicate that the user can choose among multiple devices for entering such input. However, the equivalence property is not meaningful for prompt or feedback phases of interaction, since such parts are controlled by the system according to its implementation and there is no possible choice for the user.

In this way we can flexibly specify distributions, even at a very detailed level. For example we can take one interactor to enter some text and specify that it should be distributed across three devices: a mobile, a vocal, and a graphical PC with a large screen in such a way that the prompt is complementary, the input is equivalent, and the feedback is redundant. This means that at the beginning the three devices will provide different prompts, which can address various aspects of the text to enter, then the user can freely choose the device to enter the requested text, and then all the three devices will present the feedback of the actual text entered by the user.

In the application of the CARE properties, we consider a user interface specification as a hierarchical structure: the user interface is composed of presentations, presentations contain single interactor elements, and compositions of such elements. In particular MARIA has various operators for composition of interactors:

The different types of interactor compositions are:

- *Grouping*: a generic group of interactor elements.
- *Relation*: when two or more interactors or compositions are related to each other.
- *Composite Description*: represents a group aimed to present contents through a mixture of *Description* and *Navigator* elements.
- *Repeater*, which is used to repeat the content according to data retrieved from a generic data source

When the CARE properties are applied to a high element in the structure then then they are automatically inherited by all its children. Thus, for example if we indicate that a presentation is redundant across two devices (one desktop and one mobile), it means that there will be two versions of such presentation, one for each device. Consequently, when from one point of the user interface there is an access to such presentation the two versions are activated at the same time. Another example is a grouping associated with a complementarity property. This means that part of the interactors in it are assigned to one device and part to another one. The corresponding device of each included interactor is indicated at the grouping level, and does not need to be specified again inside the interactor definitions also in this case.

This mechanism is an efficient way to associate flexibly devices to various user interface components without having to duplicate such device allocation in all the interface element definition.

SOME EXAMPLE APPLICATION

If we consider a composition operator, such as the grouping, then the specification of the involved interactors varies depending on the CARE property. In the case of *redundancy*, it is sufficient to list the devices in which all the grouped interactor should be supported. For example:

```
<grouping>
<output care_value="redundancy">
  <bind>
    <device id="paterno" platform="desktop"/>
    <device id="sisti" platform="vocal"/>
    <device id="iphone_lab" platform="mobile"/>
    <device id="manca" platform="multimodal"/>
  </bind>
</output>
</grouping>
```

The case of *complementarity* is more complex. There are various possibilities. In one case we can indicate for each interactor to what device is associated, and such interactors are not distributed in terms of their sub-components (prompt, input and feedback). Thus, in this case at the grouping level we indicate the interactor/device allocation and nothing is specified at the interactor level regarding CARE properties. This also happens in the case that there is a complementarity (thus the interactors are distributed across various devices) in which there is some redundancy (which means that in this distribution it can occur that some interactor is duplicated on multiple devices). Below you can see an example in which there is complementarity at the grouping level. Thus, one interactor (description_video) is associated with one desktop device and another interactor (description_text) is associated with three devices (one vocal, one mobile and one multimodal). Then, the corresponding concrete descriptions are provided. It is possible to note that in the case of the concrete multimodal description, CARE properties are used to indicate the allocation of the sub-components to the various modalities (in this case graphical and vocal) available on the multimodal device. Since in the example we consider an only-output object, only this aspect is assigned to one of the CARE properties (since in this case the distinction in prompt, input, and feedback does not apply given that it is only for interactive elements).

```
<grouping>
  <output care_value="complementarity">
    <bind>
      <interactor interactor_id="description_video"/>
      <device id="paterno" platform="desktop"/>
    </bind>
    <bind>
      <interactor interactor_id="description_text"/>
      <device id="sisti" platform="vocal"/>
      <device id="iphone_lab" platform="mobile"/>
      <device id="manca" platform="multimodal"/>
    </bind>
  </output>

  <description id="description_video">
    <description_desktop>
      <video src="video.flv" alt="alternative_text"/>
    </description_desktop>
  </description>

  <description id="description_text">
    <!-- REDUNDANT DISTRIBUTION -->
    <description_mobile>
      <text><string>Text </string></text>
    </description_mobile>
    <description_vocal>
      <speech><content>Text </content></speech>
    </description_vocal>
    <description_multimodal output="redudancy">
      <!-- [graphical part] -->
      <text><string>Text</string></text>
      <!-- [vocal part] -->
      <speech><content>Text</content></speech>
    </description_multimodal>
  </description>
</grouping>
```

An even more complex case of distribution is the case in which the interactors in a grouping composition are distributed according to the complementarity property, and then some of such interactors are further internally distributed according to the interaction phase. In this case part of the distribution is specified at the grouping level and part at the level of the single interactors. Below an example of this case, which distributes one interactor single_choice_id to both a desktop and a multimodal device and one interactor vocal_text to one vocal device. When we move to the specification of the single_choice_id interactor, we can see that it is distributed in such a way that the input is assigned to the desktop system while the prompt and the feedback are complementary across the desktop and the multimodal device. Then, the specification of the multimodal interactor must be consistent and using the CARE properties to distribute across modalities only those aspects that have been assigned to the multimodal device in the specification of the distribution across devices. Thus, in this example only the prompt and feedback can be distributed across the modalities of the device under consideration since the input part was distributed only to the desktop device.

```
<grouping>
  <output care_value="complementarity">
    <bind>
      <interactor interactor_id="single_choice_id"/>
      <device id="paterno" platform="desktop"/>
      <devices id="manca" platform="multimodal"/>
    </bind>
    <bind>
      <interactor interactor_id="vocal_text"/>
      <device id="sisti" platform="vocal"/>
    </bind>
  </output>

  <single_choice id="single_choice_id">
    <input care_value="assignment">
      <devices id="paterno" platform="desktop"/>
    </input>
    <prompt care_value="complementarity">
      <devices id="paterno" platform="desktop"/>
      <devices id="manca" platform="multimodal"/>
    </prompt>
    <feedback care_value="complementarity">
      <devices id="paterno" platform="desktop"/>
      <devices id="manca" platform="multimodal"/>
    </feedback>
    <choice_element value="ch_1"/>
    <choice_element value="ch_2"/>
    <desktop_single>
      <drop_down_list>
        <label label_position="left">
          <text_width="200px"><string>Label</string></text>
        </label>
      </drop_down_list>
    </desktop_single>
    <multimodal_single>
      <prompt="complementarity" feedback="redundancy">
        <drop_down_list>
          <label label_position="left"><text_width="200px">
            <string>Label</string></text></label></drop_down_list>
          <single_vocal_selection>
            <!-- PROMPT -->
          </single_vocal_selection>
        </drop_down_list>
      </prompt>
    </multimodal_single>
  </single_choice>
</grouping>
```

```

<!-- FEEDBACK -->
<vocal_feedback><content>You_choose...</content>
</vocal_feedback>
</single_vocal_selection>
</multimodal_single>

</single_choice>
<description id="vocal_text">
  <output care_value="assignment">
    <devices id="sisti" platform="vocal"/>
  </output>
</description_vocal>

  <speech><content>Vocal text to speech </content>
  </speech>
</description_vocal>
</description>

```

A similar approach can be applied also when the other composition operators of the MARIA language are considered (relation, repeater and composite description). In addition, such approach can be considered also when distributing interactors or composition of interactors at the presentation level.

EXAMPLE APPLICATION

In this section we show an example application considering a museum application in which a user plays an association game on a mobile device and at some point decides to migrate to a distributed user interface across a mobile and a large screen (see Figure 1).

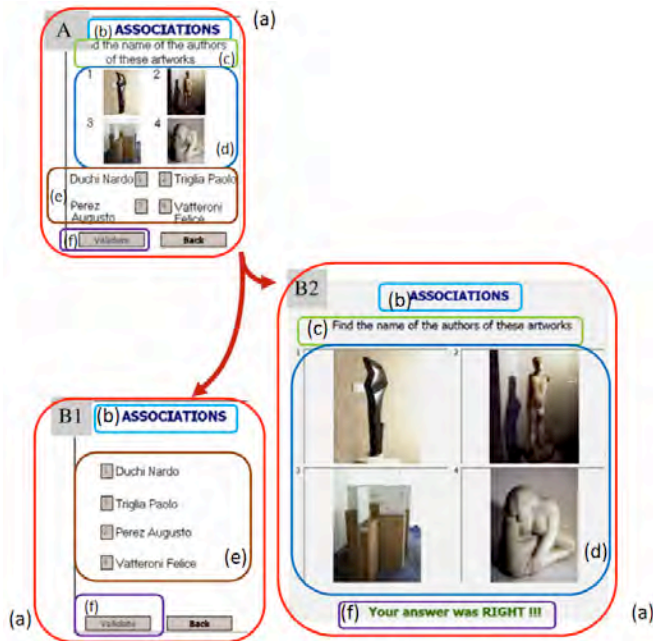


Figure 1: The example of distributed user interface.

The original grouping of elements, (a) in the figure, is distributed in a complementary way on the B1 (the mobile) and B2 (the large screen) devices. The description (text) interact (b) is distributed in a redundant manner, while the description (text) (c) is assigned only to the B2 device, along with the images (d). On the other hand, there are four text edit interactors that are assigned only to the B1 device.

Lastly, there is an activator interactor (validate button) whose input and prompt are assigned to the B1 device while the feedback (the message “Your answer was RIGHT !!!”) is assigned to the B2 device.

CONCLUSIONS AND FUTURE WORK

We have presented the method that we are using to develop the possibility of specifying distributed user interfaces with MARIA. Our approach allows designers to specify distribution in a flexible way, at various levels, not only at the level of interaction elements as it happens in other approaches. This is based on a logical framework, which distinguishes both the granularity levels for distribution and the properties indicating how such distribution should be applied to the considered elements. Such framework allows us to have a better understanding of the design space for distributed user interfaces, and exploit it in the design of the new features for the MARIA language.

Future work will be dedicated to supporting authoring of distributed user interfaces extending the MARIAE environment for this purpose.

ACKNOWLEDGMENTS

This work is supported by the EU ICT STREP SERENOA Project and the EU ARTEMIS SMARCOS Project.

REFERENCES

1. Coutaz J., Nigay L., Salber D., Blandford A., May J., Young R., 1995. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: the CARE Properties. Proceedings INTERACT 1995, pp.115-120.
2. Demeure, A., Sottet J.S., Calvary G., Coutaz J., Ganneau V., and Vanderdonckt J.. The 4C Reference Model for Distributed User Interfaces. Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems 2008, pp. 61-69.
3. Fonseca J.M.C. (ed.), W3C Model-Based UI XG Final Report, May 2010, available at <http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/>
4. M. Manca, F. Paternò, Supporting Multimodality in Service-oriented Model-based Development Environments, Proceedings HCSE 2010, 3rd Conference on Human-Centred Software Engineering, pp.135-148, LNCS 6409 Springer, Reykjavik, October 2010.
5. Paternò F., Santoro C., Spano L.D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Trans. Comput.-Hum. Interact. 16(4): (2009).
6. Vandervelpen C., Conix K., Towards Model-Based Design Support for Distributed User Interfaces, NordiCHI 2004: 61-70, 2004.

A Formal View of Distributed User Interfaces

J. J. López-Espin, J. A. Gallud, E. Lazcorreta, A. Peñalver, F. Botella
Operations Research Center University Institute, Miguel Hernández University of Elche
Avda de la Universidad s/n, 03202 Elche, Spain
{fjlopez,jgallud,enrique,a.penalver,federico}@umh.es

ABSTRACT

The traditional concept of user interface is changing as new interaction possibilities are becoming real in the form of surprising devices supporting amazing interaction mechanisms. In this sense, the increasing use of different displays managed by several users to accomplish a common task, adds the term "distributed" to the traditional concept of user interfaces. This paper offers a formal view of distributed user interfaces (DUI) as a mean to understand better their essentials and to establish the bases for formally proving properties as correctness and coherency. The proposal has been applied to a case study.

Author Keywords

Formal models, distributed user interfaces, formal specification.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: Miscellaneous

General Terms

Formal specification, Distributed User Interfaces.

INTRODUCTION

Some years ago interaction means were limited to the use of mouse, keyboards and displays. Nowadays, users can interact with computers in many different ways, as for example by using gestures or view.

The use of formal models for user interface design can provide a number of benefits. It can help to ensure consistency across designs for multiple platforms, prove properties such as reachability and completeness [1]. Previous efforts dedicated to specify user interfaces (UI) [2] must be revisited and redefined in order to consider this new interaction environment provided by distributed user interfaces.

The paper is organized as follows. First we briefly describe the field of Distributed User Interfaces. Next section summarizes the set of key definitions about DUIs. Then the

formal specification is proposed. In the next paragraph, the proposed notation is applied to specify distributed user interfaces in different real examples. Finally, we provide conclusions and directions for future work.

DISTRIBUTED USER INTERFACES

A distributed user interface is an interface where all or some elements of the interface can be distributed across multiple displays or multiple platforms. In this way, a group of users can collaborate by means of DUI to accomplish several tasks. A single user can move his interaction activity from one platform to another by means of an application that supports DUIs.

According to [4], DUI concerns the repartition of one or many elements from one or many user interfaces in order to support one or many users to carry out one or many tasks on one or many domains in one or many contexts of use, each context of use consisting of users, platforms, and environments. Vanderdonckt explains that where UI distribution is supported, UI federation is needed. UI federation supports the concentration of UI elements from distributed sources.

The former definition of DUI is the starting point of the proposal described in this paper. The DUI adds the term "distributed" to the concept of user interface.

Before presenting our proposal, the definition of the user interface must be revisited to give it a broader view. That is justified by the fact that Graphical User Interfaces (GUI) have lost the traditional prominent position in the field of interaction devices.

DUI: DEFINITION OF TERMS

The definition of Distributed User Interfaces (DUI) is based on the definition of User Interface (UI). In this section we are revisiting each term included in the former definition of DUI: user interface elements, UI, user, task, domain, contexts of use. Some authors have proposed a new definition of UI concept by using the term Human User Interface (HUI) [3] to underline the fact that the interface concept is becoming an element that is "nearer" the user than the computer. In the traditional definition, the interface is "nearer" the computer or part of the computer indeed. In this sense, a UI is a set of elements that allows users to interact with computers. These elements can be categorized into input data elements, output data elements and control elements. This definition of UI supports all kind of

technologies and interaction mechanisms. The task can be defined as the set of actions the user performs to accomplish an objective. A DUI system is an application or a set of applications that makes use of DUIs, since these applications share the user interface. A DUI system can be implemented by means of several kinds of devices, hardware and software platforms. So, for the purposes of this paper, there is no need to maintain the differentiation between device and platform.

ESSENTIAL PROPERTIES OF DUIs

In this section a set of essential DUI properties is presented. We mean by essential the features that should be present in a DUI system to be considered as such. Taking into account the former definition of DUI, we define as essential properties the next: portability, decomposability (and composability), simultaneity and continuity.

The next paragraphs are devoted to describe each property.

Portability: This property means that the UI as a whole or elements of the UI can be transferred between platforms and devices by means of easy user actions. For example, a user is running a graphic editor in his desktop computer and with a simple action, decides to transfer the color palette panel (UI element) to another platform (a portable device).

Decomposability: A DUI system is decomposable if given a UI composed by a number of elements, one or more elements of this UI can be executed independently as UI without losing their functionality. For example, an application calculator can be decomposed in two UI elements, the display and the numeric keyboard. This property can be used together with Portability in order to allow, for instance, the keyboard being executed in a smartphone, while the display can be showed in a public display. These two UI elements can be also joined in a unique UI (composability).

Simultaneity: A DUI system is said simultaneous if different UI elements of the same DUI system can be managed in the same instant of time on different platforms. For example, two or more users can be using the same DUI system, each one interacting with one of the different platforms at the same time. This does not imply that all DUI systems are multiuser as we see later.

Continuity: A DUI system is said continuous if an element of the DUI system can be transferred to another platform of the same DUI system maintaining the state. For example, a user can be on a call in his mobile phone walking on the street and when he arrives at home, he can transfer the call to the TV without interrupting the call.

Figure 1 shows the relationships among essential and derived properties of DUIs.

Other properties can be derived from the former ones:

Multiplatform: The Portability and Simultaneity properties imply the DUI system is making use of more than one

platform (or device). This property can be considered essential but derived.

Multimonitor: This property suggests that a DUI system is implemented by using more than one monitor (or display). Although this can be considered the usual case, it is not a mandatory property as we can implement DUI with only one display or monitor. This is why we do not distinguish between display and monitor.

Multiuser: The Simultaneity property can suggest that more than one user is interacting with the DUI system. Once again, this can be considered the usual case although it is not mandatory because we can implement simultaneous UI in a DUI system without multiple users.

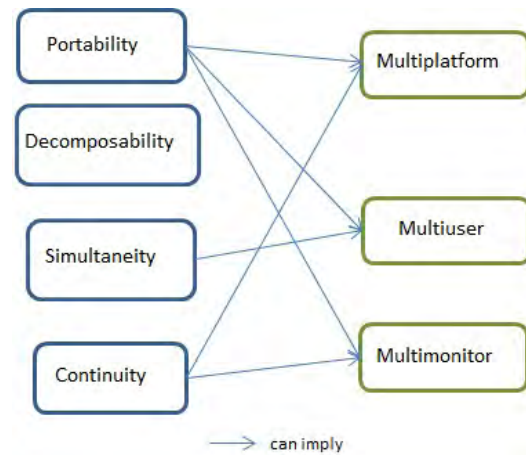


Figure 1. Relations among properties

We can define additional not essentials but desirable properties of DUI systems as are consistency and flexibility.

Consistency: A DUI system is said consistent if different UI elements of the same DUI system are managed in the same way. For example, a group of users can be interacting with a DUI system by using two touch screens of two different platforms. The action “object selection” should be supported on both platforms (it is not required to be performed in the same way).

Flexibility: A DUI system supports flexibility if users can perform the same action in different ways supported by the different platforms of the DUI system. For example the action “delete the selected object” can be performed in many different ways on the same DUI system, depending on the platform used.

Efficiency: Users should be able to perform common tasks on DUIs with a reasonable cost.

In addition to the previous properties, it would be desirable to guarantee some of the traditional quality criteria applied to user interfaces, as it can be *usability* and *accessibility*.

FORMAL SPECIFICATION OF DUIS

This section is devoted to present our proposal for specifying formally the concept of Distributed User Interface taking as starting point the definition given in section 1. In order to define a DUI, we first need to define other important concepts as interaction element, functionality, user interface and platform.

Definition 1: Interaction Element

An Interaction Element $e \in E$ is defined as an element which allows an user u to carry out an interaction through a platform p (denoted by $u \sim^e p$).

The notation allows us to define the kind of interaction among three possible (input, output and control). The element can be an input data element $u \sim_{-}^e p$ or an element that displays data $u \sim_{+}^e p$ or a control or command element $u \sim_c^e p$. In this paper we are using the generic $\sim^e$ that includes the different kind of interaction flows.

Definition 2: Functionality

Functionality has two viewpoints. If we consider an interaction element we can think about its functionality as the pure interaction between a user and a computer, while considering a set of interaction elements we can think in its functionality

Two interaction elements e and e' have the same functionality if both can perform the same interaction between a user and a computer. We denote it with $e =^F e'$. In this sense a GUI button has the same functionality as a gestural movement if the computer receives the same order when one of them are used by the user. Similarly, a warning label has the same functionality as an audio alert if the user receives the same information when the computer runs it. as the common goal connected to a user's task.

On the other hand, a subset of elements $E_0 \subset E$ is a subset with Functionality if $\forall e \in E_0$, e participates in a common goal or supports a user's task in E_0 (denoted by $e \in^F E_0$).

Definition 3: User Interface

An User Interface $i \in UI$ is a set of interaction elements such as $i = \{e \in E \mid e \in^F i\}$. It is possible to realize that $UI \in P(E)$, that is to say UI belongs to E 's parts.

Once the concepts of interaction elements and functionality are laid down, a user interface (UI) is defined as a set of interaction elements that are included in the same use context for allowing users to carry out a given computer task.

Definition 4: Platform

Let P be denoted by the set of platforms $p \in P$ within which an interaction element $e \in E$ exists (denoted by $\sim^e p$), in the sense that the interaction element can be implemented, supported or executed on that platform. This definition also includes the existence of a framework that supports the interaction element.

We will say that a $i \in UI$ is supported on a platform $p \in P$ if $\forall e \in i$ then $u \sim^e p$ being u a user. We will use the notation $u \sim^i p$.

In addition we will say that a $i \in UI$ is supported on a set of platforms $P_0 \subset P$ if $\forall e \in i, \exists p \in P_0 \mid u \sim^e p$ being u a user. We will use the notation $u \sim^i P_0$.

Definition 5: Distributed User Interface

A Distributed User Interface $di \in DUI$ is defined as

$$\begin{aligned} di &= \bigcup_{k=1}^N i_k = \\ &= \bigcup_{k=1}^N \{e_{kj} \in E \mid e_{kj} \in i_k, j = 1 \dots N_k, e \in^F di\} \\ &\quad \text{exists } n_p > 1 \text{ platforms } \{p_s \in P \mid s = 1 \dots n_p\} \text{ and} \\ &\quad di = \bigcup_{s=1}^{n_p} \{e_{sj} \in E \mid u \sim^{e_{sj}} p_s, j = 1 \dots n_{p_s}, e_{sj} \in^F di\} \\ &\quad \text{for a user } u \in U. \end{aligned}$$

So, a Distributed User Interface is the collection of interaction elements that forms a set of User Interfaces, that is to say, a set of elements with a common functionality. These elements are distributed over a set of platforms without losing the common functionality given by the user's tasks. It is also possible to consider a group of users instead of a single user.

Using this new notation it is possible to rewrite different situations according to previous similar works:

- We can express the user interaction using traditional UIs as $u \sim^i p$, being $i \in UI$.
- The user interaction by means of DUIs can be expressed as $u \sim^{di} p$ being $di \in DUI$.
- Finally, we can express the interaction of multiple users on multiple platforms by means of DUIs as $\{u \mid u \in U\} \sim^{di} \{p \mid p \in P\}$

DEFINING THE ESSENTIAL PROPERTIES

Once the formal definition of DUI concept has been presented, the essential properties exposed above can be formalized in the same way.

Portability

According to the previous textual definition of portability, we can say that a user interface $i \in UI$ is portable if there exists $E_0 = \{e \in E \mid e \in i\}$ and a group of users u_1 and $u_2 \in U$ such as $u_1 \sim^{E_0} p$ and $u_2 \sim^{E_0} p'$ being $p, p' \in P$. We can also consider $u_1 = u_2$.

Decomposability

A distributed user interaction $di \in DUI$ is decomposable if exists a subset E_0 of di such as

$$E_0 = \{e \in di \mid e \in^F E_0\}$$

Simultaneity

A distributed user interaction $di \in DUI$ accomplishes the simultaneity property in $p_0, p_1 \dots p_n \in P$ with $n > 1$ and a group of users $u_k \in U$ if $di = \bigcup_{j=1}^N i_j$ with $i_j \in UI$, and $u_k \sim^{i_j} p_s$ in the same temporal instant, with $j = 1 \dots N$ and $s = 1 \dots n$ and $k = 1 \dots n$.

Continuity

A user interaction $i \in UI$ is continuous in $p_0, p_1 \in P$ if $\forall e \in i \mid u \sim^e p_0, u \sim^e p_1$ maintaining the state of i .

AN EXAMPLE OF APPLICATION

In this section we use the exposed notation to specify distributed user interfaces used in different real examples.

The first example is the one exposed in [4] (section 2.6), which describes how to distribute GUIs as the more studied UIs due to the capacity of different platforms to display common interaction GUI elements. The decomposition of a dialog is a good example of the essential property of Decomposability. As the author describes, a dialog can be split if it contains blocks of elements with their own functionality, which in our notation could be expressed as $i_k \in^F di, k = 1, 2, 3$. That is to say, the three different dialogs $i_k \in UI$ compose the $di \in DUI$, as a set of user interfaces that have the common functionality.

A second example could be WallShare [5]. WallShare provides a shared zone that is displayed by a projector over a wall. Users can collaborate through the shared zone using their own mobile devices, such as mobile phones, PDAs, laptops and so on. In WallShare we have, for example, two users with their own platform (mobile device), $u_1 \sim^{i_1} p_1$ and $u_2 \sim^{i_2} p_2$ and the platform responsible of the shared zone, p_3, i_3 . The DUI in this case would be $di = \{i_1, i_2, i_3\}$ with this two kind of interactions, $u_1 \sim^{di} \{p_1, p_3\}$ and $u_2 \sim^{di} \{p_2, p_3\}$. Regarding to the essential properties, we have the following situations:

- *Portability*. Considering the users' pointer control as part of the i_3 user interface, since this interface is distributed on platforms p_1 and p_2 , we say that WallShare accomplishes this property.
- *Decomposability*. The user interface is divided into a common display elements and several distributed control interfaces. Each i_k maintains the general functionality.
- *Simultaneity*. A group of users $u_k \in U, k \geq 1$, can work at the same time in different platforms.
- *Continuity*. WallShare also accomplishes this property since the state of the common display is updated and visible for all users in real time.

We consider a third hypothetical example related to the zoom control of an application, where we can use different interaction elements: a slider, a text input, an up-down buttons, two buttons, a keyboard (Ctrl+-, Ctrl++) and so on. We can transfer two buttons to another platform (i_2, p_2) from the main platform (i_1, p_1) for managing the zoom (the

common goal). Our DUI could be expressed as $di = \{i_1, i_2\}$ and we can discuss how it could accomplish the four essential properties:

- *Portability*. The main application can receive the keystroke of every button and trigger the tasks Zoom-Out and Zoom-In.
- *Decomposability*. In this case we have split the original UI into two UIs that form our DUI di with the same original functionality.
- *Simultaneity*. This DUI can be used simultaneously by two users in order to manage the common zoom.
- *Continuity*. The i_2 does not maintain a state (it consists in two buttons) but affects the overall state. In this case, we say that the DUI should maintain the state of the overall zoom in the different controls.

CONCLUSION

This paper presents a new notation to describe formally the Distributed User Interface (DUI) concept. This formal notation has allowed us to determine the essential properties of DUIs as decomposability, portability, simultaneity and continuity. This notation has been proved powerful in defining previous definitions or characterizations of distributed user interfaces. Future work will include the revision of the quality in use (usability and accessibility) concept in order to evaluate the quality of Distributed User Interfaces.

REFERENCES

1. Bowen, J., and Reeves, S. Using formal models to design user interfaces: a case study. In Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it - Volume 1, BCS-HCI '07, British Computer Society (Swinton, UK, UK, 2007), 159–166.
2. Chi, U. Formal specification of user interfaces: A comparison and evaluation of four axiomatic approaches. IEEE Transactions on Software Engineering 11 (1985), 671–685.
3. Gallud, J. A., Villanueva, P. G., Tesoriero, R., Sebastian, G., Molina, S., and Navarrete, A. Gesture-based interaction: Concept map and application scenarios. In Proceedings of The 3rd International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services. CENTRIC 2010, IEEE (2010), 28–33.
4. Vanderdonckt, J. Distributed user interfaces: How to distribute user interface elements across users, platforms, and environments. In Proc of the International Conference Interaccion 2010 (2010).
5. Villanueva, P. G., Tesoriero, R., and Gallud, J. A. Multi-pointer and collaborative system for mobile devices. MobileHCI '10, ACM (New York, NY, USA, 2010), 435–438.

Distributed User Interfaces in Ambient Intelligent Environments: A Tale of Three Studies

Pavan Dadlani, Jorge Peregrin Emparanza

Philips Research
High Tech Campus 34, Eindhoven 5656AE, The Netherlands
pavan.dadlani@philips.com

Panos Markopoulos

Eindhoven University of Technology
P.O. Box 513, Den Dolech 2
Eindhoven, 5600 MB, the Netherlands
p.markopoulos@tue.nl

ABSTRACT

In this paper we describe the research conducted at Philips Research regarding the use of Distributed User Interfaces (DUIs) in the field of Ambient Intelligence (AmI). We first introduce the emerging area of AmI, followed by describing three case studies where user interfaces or ambient displays are distributed blending in the user's environment. In such environment, technology is hidden capturing contextual information aimed at different applications, each displaying information tailored to the user. We end the paper with lessons learned from these projects on DUIs in ambient intelligent environments.

Author Keywords

Distributed user interfaces, ambient displays, ambient intelligence, social connectedness.

ACM Classification Keywords

H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION

Ambient Intelligence (AmI) refers to electronic systems that are sensitive and responsive to the presence of people [1]. It is a paradigm envisioned to materialize in the next few years [2], when the home is expected to become populated by large number of 'smart' devices interconnected via an invisible web of networked services [1]. In several ambient intelligent environments, it is required for having mechanisms for users to interact with such AmI systems for control, awareness, etc. Furthermore, users are required to interact with interfaces that blend in their environment and even accessible in several locations. This research portrays the distributed user interfaces in three different ambient intelligent environments aiming at enhancing social connectedness between two or more remote parties.

The first is related to DUIs in ambient assisted living supporting aging in place and social connectedness between elderly and their adult children. The second one entails a DUI supporting ambient telephony (a hands-free telephony based on capturing the user's presence around the home). Lastly, we present the family bonding devices aiming at enhancing social presence and connectedness between empty nesters and their children living apart from each other.

In all three cases, the technologies used are based on advanced sensing mechanisms either by distributed sensors recording raw sensor firings, or audio signals from distributed microphones, and/or video signals from distributed cameras. Sensing in any of these modalities is followed by a reasoning engine for context determination.

DUI FOR AMBIENT ASSISTED LIVING

In the Ambient Assisted Living domain, intelligent environments are created with the aim of capturing and understanding contextual information of an elderly's person and make inferences regarding their wellbeing. These systems are often referred to as awareness systems. Several examples in the field have demonstrated the value of such systems in bringing elderly and their children closer [5,6,7]. However, the design of these earlier systems did not have a focus on the aesthetic aspects of the interaction and the appearance of the displays, despite that their importance was acknowledged by researchers and confirmed in their surveys of user opinions. Furthermore, there is still lack of empirical evidence of such systems and an understanding of the needs of elderly and their children.

Focus groups and interviews with caregivers and seniors, showed that rather than detailed trivia of daily activities, care giving children wish to obtain high-level overviews about their parents' wellbeing, but only if a pattern has changed. They wish to be informed about gradual changes in lifestyle that may not be easy to notice during visits. Although they value the possibility to see historical data, they dislike the idea of getting a daily status report or a 'story' with detailed schedules and activities, as they felt that this would be too intrusive. Among the various possibilities to present awareness information, participants favored interactive digital photo frames.



Figure 1. Distributed photo frames with light auras indicating high level changes in the elderly's wellbeing, and intelligent photo frame jewelry to exchange affective messages.

Design of the Awareness System

Based on the insights from the user study we implemented and iteratively evaluated a system presenting information about the elder regarding their presence at home, sleep patterns, weight variations, mobility, and bathroom visits. The system includes an interactive photo frame which emits colored light 'auras' presenting at-a-glance information regarding the elder's wellbeing (Figure 1). If the elder is at home, a blue aura appears on the child's photo frame, which disappears if the elder leaves. To provide awareness of changes in wellbeing a yellow or orange aura is formed, depending on the severity of the change. When users interact with the photo frame by touching the screen, charts are displayed showing long-term trends of wellbeing. Wellbeing parameters are detected using distributed hidden sensors around the home. For example, sleep and weight are captured using load sensors placed under each leg of the bed.

During the initial studies conducted in people's home, there was a need for more symmetrical flow of information and having a similar user interface in the elder's home giving information regarding their children. There was also a need to share more affective information regarding each other and more positive news (instead of only negative alerts

regarding the elder's wellbeing). Thus, we improved the system by providing the same user interface (an augmented photo frame with light auras to the elder and being able to share emotions via tangible playful objects. See Figure 1).

There is certain wellbeing information that elderly's children prefer to have formal care providers receive and make a better interpretation of the elder's status. Furthermore, there are emergency situations (e.g. a fall in the bathroom or leaving home at night) that the system can also detect, but that is preferred to involve a care provider for immediate action. Thus, we included a channel of information sent to the care provider in the form of mobile messages and a web-portal. Figure 2 gives an overview of the system with tailored user interfaces for the different parties involved.

Two trials of two weeks each and an extensive trial of six months confirm earlier findings regarding the potential of this class of applications. The user studies revealed that children appreciated being involved in the elderly's wellbeing. Privacy did not seem to be an issue since elderly parents are willing to share such information with their children and in fact were pleased to receive phone calls from them when light auras were shown on their photo frame. It was clear that intertwining an implicit interface (colored light auras on the frame) with an explicit form of communication further fostered reassurance and connectedness.

This study showed the need to transcend momentary awareness of commotion, or mundane activities like prior work [5,6,7] and to focus on long-term trends in their elderly parents' wellbeing. To further motivate the adoption of such a system and to enhance the connectedness between elderly and their children, our study suggests coupling awareness information with a bi-directional

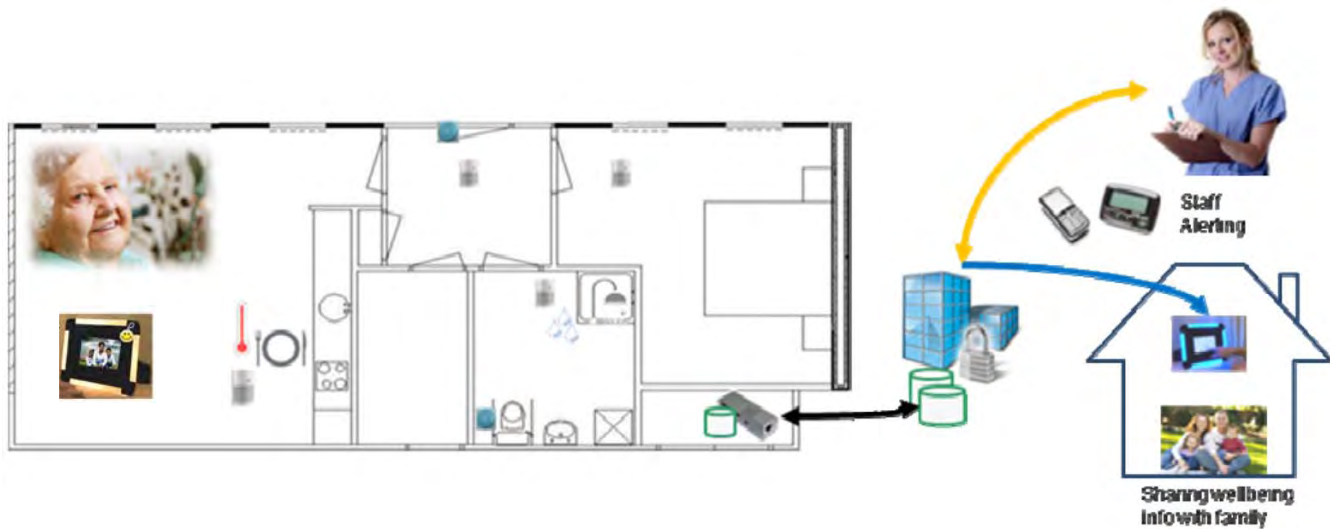


Figure 1. Overview of the ambient assisted living system with tailored user interfaces for each party. The sensors in the elderly's home capture contextual information to assess their wellbeing and inform family and care providers.

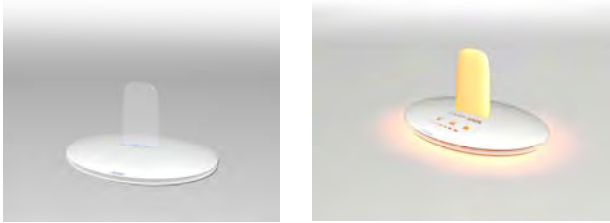


Figure 3. A rendering a unit in standby and conversation mode.

communication; an abstract symbolic mapping of emotions to colored auras was found appropriate in this design as it helped trigger direct contacts.

DUI FOR AMBIENT TELEPHONY

Ambient telephony is based on the concept of blending communication into the home environment [8]. The system consists of a set of loudspeakers and microphones around the house, which detect the user's presence, and renders the call only in the area of the house where the person is, and thus, allowing free mobility. This allows for completely hands-free conversations, without the boundaries of a device or a physical location. It aims to elicit the feeling of a visit instead of just a phone call and to lower the awareness of mediation in the communication.

In order to understand how people would like to use an ambient telephony system, an exploratory study was carried out with 10 participants. A basic set-up of ambient telephony built into the ceiling of a home-like laboratory was used for the experiment. Participants recruited had to have a contact with whom they have a caring and affective relation at a distant location and also be familiar with voice over IP (VOIP) communication systems. In this initial study, participants commented very positively on ambient telephony considering it more comfortable and convenient than cordless, especially regarding its hands-free and location-free use. Some participants commented about feeling their contact closer, as if they were in the same room. Participants noticed the sound sources (speakers on the ceiling) and in many cases showed intimacy behaviors (like nodding and staring at them). On the other hand, there were negative remarks about *"talking to the ceiling"*, the *"unnatural"* position of the sound, not having anything to look at, and the low quality of the sound. These could be annoying, especially when multitasking.

A user study revealed certain requirements for the user interaction of ambient telephony. For example, the sound should come from face height with sources at the center and side locations, and should have a meaningful representation of the remote contact to match both visual and audible stimuli in an affective manner allowing silent presence. Furthermore, the distributed units should be visually unobtrusive and aesthetically pleasant and be flexible to adapt to different spatial needs. These led to the new

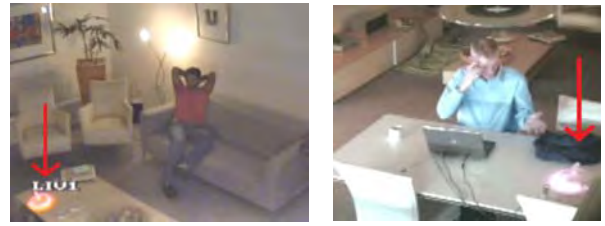


Figure 4. User test of the ambient telephony system: (left) a user laying back during the call and (right) a user addressing the rendering unit while performing a task.

ambient telephony design based on distributed units with ambient lighting as shown in Figure 3. It is based on a flat small object with a translucent upper part which glows in the color the user assigned to the contact and will be 'invisible' when it is off. When the user walk around the house the unit closest to the user will take over the call and light up, thus the call follows the user around the house.

A user test in a home-like environment was conducted with the ambient telephony system. Users were asked to conduct a certain number of tasks around the house while making a phone call to their loved one. Figure 4 shows sample shots of the user test. Users appreciated the distributed nature of the phone call and the 'follow-me' effect. Aesthetics showed to be particularly important to elicit feelings of social presence. Neutral not-phone-like rendering units are appreciated and suitable solutions for ambient telephony, helping to lower the feeling of mediation and thus enhance social presence. Furthermore, simple touch contact oriented intelligent interfaces are preferred over buttons. The fewer the necessary steps to perform an action (like setting a call) the lower the feeling of mediation is assumed to be.

DUI FOR FAMILY BONDING

The third case study at Philips Research was the development of new Social Presence Technologies (SoPresenT) to capture and convey contextual information, with the aim of exploring mechanisms that will strengthen the caring relationship between two remote parties. Typical context-aware systems entail distributed sensors strategically placed in an environment and a dedicated or separate user interface blending in the environment for target users. The SoPresenT system however, combines both sensing and the user interface in one embodiment; thus, having one dedicated device (e.g. placed in a living room) that will capture awareness information and render it on similar remote distributed devices).

The device is equipped with a microphone and a camera. The working principle is based on audio and video scene analysis algorithms that capture presence and activity of a user, together with computational intelligence for interpreting contextual data. Figure 5 shows the SoPresenT device in two separate locations, each rendering presence information about the remote party: as soon as the one user



Figure 5. An example of the SoPresenT devices distributed remotely conveying presence and motion information of the remote party.

moves within the living room, the device captures the angle with respect to the device and renders a moving glow of light on the remote device, such that the glow of light represents the movement patterns of a remote party. Furthermore, users can further interact with the SoPresenT device by either tapping it or caressing it. These actions are captured by the microphone and using audio analysis it is possible to distinguish between these actions. By caressing the device the user triggers the remote device to glow indicating the person is thinking about the remote party. By tapping the device the user requests information (patterns of light) of the remote party regarding how active they have been around their device. Furthermore, when one taps their device requesting information of the remote party, the remote party's device can glow indicating that their information has been requested.

Several user studies are under way to research different affective mechanisms with the SoPresenT user interfaces and understand what works and what doesn't when aiming at enhancing social connectedness between remote parties, taking into account privacy and control aspects.

CONCLUSION

In ambient intelligent environments, distributed user interfaces play an important role to fulfill the requirements of typical AmI systems. We have explored three case studies if using DUIs for different AmI applications. There are several lessons to draw regarding DUIs in AmI environments. Firstly, we have seen that AmI strives to

blend technology within the environment and grasping the attention of the user in a calm and unobtrusive way (e.g. light auras). Secondly, we have explored the need to have DUIs combine both implicit and explicit forms of interaction. The added value of this dual nature of the interface allows to easily moving from high-level awareness information that can inform users in their periphery, towards a more focused interface requiring their full attention. Lastly, we have seen distributed interfaces both geographically (ambient assisted living and family bonding) and locally (ambient telephony). The geographical distribution typically entails exchange of information or rendering information of the remote party.

Further research will further explore different modalities of DUIs and combination of such modalities to provide awareness information in ambient environments.

ACKNOWLEDGMENTS

The authors would like to thank Alexander Sinitsyn and Simon Abernethy for the implementation of the ambient assisted living project; and colleagues from the Philips Research Laboratories for their help and support in the projects.

REFERENCES

1. E. Aarts, H. Harwig and M. Schuurmans, Ambient intelligence, in: *The Invisible Future*, J. Denning, ed., McGraw-Hill, New York, 2001, pp. 235–250.
2. E. Aarts and S. Marzano, eds, *The New Everyday: Visions of Ambient Intelligence*, 010 Publishing, Rotterdam, 2003.
3. Dadlani, P., Sinistyn, A., Fontijn, W. and Markopoulos, P. Aurama: caregiver awareness for living independently with an augmented picture frame display. In *Artificial Intelligence & Society*, Springer (2010).
4. J. Peregrin Emparanza, P. Dadlani, B. de Ruyter, A. Harma: Ambient Telephony: Designing a Communication System for Enhancing Social Presence in Home Mediated Communication. In *Proceedings of ASCII 2009*, IEEE.
5. S. Consolvo, P. Roessler and B. Shelton, The CareNet display: Lessons learned from an in home evaluation of an ambient display, in: *Proceedings of Ubicomp*, 2004.
6. G. Metaxas, B. Metin, J. Schneider, P. Markopoulos and B. de Ruyter, Diarist: Aging in place with semantically enriched narratives, in: *Proceedings of Interact*, 2007.
7. J. Rowan and E. Mynatt, digital family portrait field trial: Support for aging in place, in: *Proceedings of CHI 2005*, ACM Press, 2005, pp. 521–530.
8. A. Härmä (2007). Ambient telephony: scenarios and research challenges, *Proc. INTERSPEECH 2007*, Antwerp, Belgium, 2007.

Visually Augmented Interfaces for Improving Awareness in Mobile Collaboration

Barrett Ens

University of Manitoba
Winnipeg, MB, Canada
bens@cs.umanitoba.ca

Rasit Eskicioglu

University of Manitoba
Winnipeg, MB, Canada
rasit@cs.umanitoba.ca

Pourang Irani

University of Manitoba
Winnipeg, MB, Canada
irani@cs.umanitoba.ca

ABSTRACT

Software, particularly for mobile devices, has not traditionally supported collaboration. In our investigation of multi-user software requirements for co-located mobile collaboration, we find that awareness is of utmost importance to the user. We present our concept for a system that enhances user awareness through a novel combination of information visualization and mobile projection. We choose visualization techniques appropriate for portraying specific aspects of awareness and modify them to suit the corresponding display medium.

Author Keywords

Mobile projection, collaboration, visualization, groupware, awareness, CSCW, small screens.

ACM Classification Keywords

H5.3. Information interfaces and presentation: Group and organization interfaces – Computer-supported cooperative work.

INTRODUCTION

Groups of people who work collaboratively impose atypical requirements on software designers, who have traditionally focused on the individual user. Multiple users can share a computer screen, but standard input controls are built for one. Designers of a collaborative workspace must attend to technical issues like connectivity and data concurrency as well as user requirements. In particular, they must provide the user with awareness of others' presence, activity and focus. The small screens of mobile devices compound the difficulties of developers and users. The developer is restricted in the type and amount of content they can choose to display. Users cannot easily view another's output by looking over their shoulder.

We propose two methods for providing awareness for co-located collaborators aimed at optimizing performance in the domain of mobile map navigation. First, we augment the user interface with information visualization techniques to provide information about the current and past activities

of others. Second, we adapt each technique to a high-level view of the shared workspace furnished by a mobile projector. With these combined augmented views we can produce an intuitive and effective multi-user interface, supporting awareness of other users' interaction with the workspace, including elements such as their location, activity, navigation history and viewing scale.

Scenario

Imagine two visitors to a city, both browsing a list of suitable restaurants or hotels on their mobile devices (Figure 1). Because they are co-located, they have the benefit of verbal communication, facial expressions and gestures. There is no technical support for their task, however, potentially resulting in a less-than-optimal product of their efforts. Each individual is focused on her own device, with no awareness of what the other is doing, leading to duplicated effort. It is awkward for one person to point out an item of interest, and not necessarily trivial for the other to navigate to the same location on their own view. The situation described here is not uncommon, yet no adequate support currently exists to facilitate co-located mobile group interaction.



Figure 1. Scenario: Imagine two visitors using their mobile devices to search for a restaurant or a hotel.

AWARENESS AS A BASIC USER REQUIREMENT

'Groupware' is a general term used to describe a multi-user software application with a shared workspace. A survey of research literature concerning groupware system requirements turns up a list of recurring themes. These can be loosely classified into user and system requirements. User requirements are central to usability, whereas system requirements concern technical issues that typically remain invisible to the user when the system is functioning well. Our focus is on high-level user requirements.

Herskovic et al. [6] provide a concise summary of requirements for designing groupware for mobile devices: autonomy, flexibility, consistency/availability, connectivity, heterogeneity/interoperability and communication. The majority of the items on this list can be classified as system requirements. The one exception of significant concern to the user is functionality for awareness. (Communication is also mainly technical, and specifies the application's networking capabilities and supported modes of operation, for example synchronous or asynchronous).

In the domain of collaborative Web search, Morris and Horvitz [7] identify awareness along with two other high-level user requirements: division of labor and persistence. Of these, we can single out awareness as the single overarching requirement. Persistence, from the user's perspective, is a manifestation of consistency and availability. These require data concurrency, and support the disconnected and asynchronous modes of collaboration by allowing a user to 'take away' information from a session or revisit it later. Division of labor is important for any group endeavor, although we believe that, in the majority of situations, existing social mechanisms for delegation will function naturally given sufficient support for awareness. This is supported by the negative reactions garnered by the automatic search division methods that Morris and Horvitz investigated.

Greenberg et al. [3] break awareness into several types and single out 'workspace awareness' as a fundamental requirement for groupware systems. They identify three other forms of awareness, all of which intersect with workspace awareness: informal awareness concerns who's who in a work environment; social awareness pertains to physical and social cues, such as emotion and focus of attention; and group-structural awareness regards the roles of group members, including the division of labor. In contrast, workspace awareness is about changes to the physical state of the shared workspace and allows one user to be informed about the actions of another. All forms of awareness are automatic in a real-world environment, but must be thoughtfully supported in groupware systems.

These and many other examples show that adequate support for awareness is vital to the success of any multi-user application. We approach the problem of awareness support with a two-part solution: the first leverages state-of-the-art information visualization techniques to augment the mobile device views of a shared workspace; and the second similarly augments a shared view provided by a mobile projector. We argue that these two mechanisms offer different advantages and disadvantages, which we can tease apart through careful analysis of empirical results.

USER-CENTERED DESIGN

To inform our design goals, we followed our initial requirements analysis with a user inquiry. We conducted several informal user studies to gain an understanding of collaborative usage patterns with current awareness

techniques and identify the barriers to collaborative activity on mobile devices. All participants were university computer science students who routinely use mobile devices. The studies included surveys and two sets of focus groups. In the first, we asked participants how they would go about engaging in map-based collaborative tasks using paper maps, desktop computers and mobile devices. In the second study groups of two were given two Nokia N900 smart phones and a route planning task so we could observe what they did.

Preliminary qualitative analysis of the data we collected confirms our intuition about obstacles in mobile collaboration. Comments similar to the following were made by several users:

"Small screens... Multiple people [are not] able to [view] input at the same time. Those are the two main barriers."

"When you have a map laid out or if you have a bigger computer screen, it's a lot easier to look over someone's shoulder..."

In other words, mobile devices lack multi-user support, for both input and output. People would prefer to crowd around a desktop computer or paper map than use mobile devices for collaborative tasks. If each user is restricted by an individual disconnected viewport, collaboration is severely impaired.

There is a caveat to the above conclusion, however, summarized by the following comments:

"If the same 'map' could be looked at on multiple devices without forcing the same view between all the displays and give the user the option to 'point out' points of interest that other collaborators could look at..."

"The screen size is no longer a limitation in that case, nor is everyone not being able to have input because you can each do your own thing."

What these participants are describing is a shared workspace. In other words, if their devices are connected to a central workspace, people are no longer reluctant to use them to participate in collaborative activity.

For future user studies, we will implement a model application with a shared workspace, in which participants can perform real-world map-based tasks. We will examine the user acceptance to, and performance gains of several visualization methods, including those described in the following section. In addition, we will contrast user trials conducted with and without a projected workspace. This will allow us to empirically measure user performance and qualitatively analyze users' reactions and preferences. Development over several iterations, interlaced with pilot studies, will allow us to solve issues such as scalability of the visualization cues, the success in each of our features towards providing its intended type of awareness, and what types of tasks they will improve.

VISUAL AUGMENTATION FOR AWARENESS

To design features for awareness, we must examine the aspects of awareness we want to address. First, we identify two main categories of awareness: spatial and temporal.

Spatial Awareness

Spatial awareness is lacking on mobile devices, mainly because of their inherently small screens. Paper maps, in contrast, typically fold out to a relatively large size, allowing their users to view a large workspace together and to reference a wide range of locations. Desktop monitors are restricted in size, but compensate with interactive navigation support. Researchers have attempted to mitigate the major disadvantages of a limited display area by devising ways to extend the effective area of the interaction space. For example, Greenberg et al. [3] use a fisheye view to allow more content to fit on a screen than otherwise possible. By applying a fisheye view for a text document, the researchers include the scope of several users editing a text document simultaneously. Unfortunately, the distortion caused by fisheye views on maps is known to counter productivity gains.

Another technique that provides spatial information is the overview, which provides a scaled-down view of a large workspace (e.g. [8]). Although overviews provide much useful information, such as the relative positions of multiple objects, they consume scarce screen space.

As an alternative, we use visualization cues that provide spatial information about objects beyond the screen edge, such as Wedge [4]. Wedge provides both direction and distance information and scales to several targets, while remaining resistant to negative effects from clutter. Thus, we can use the technique for several purposes simultaneously, for instance to indicate the region of the document where another user is browsing as well as regions they have visited previously.

The issue of spatial awareness is complicated by the fact that multiple users may choose to view a shared workspace at different scales. We use the term *intra-scalar* to describe an application that supports interaction between many possible scalar combinations. Two people viewing the same location at different scales in a naïve application may be unaware of differences they see, potentially hampering communication. Our intra-scalar application will mitigate such difficulties by providing awareness of differences in scale. We propose to do so by displaying a bounding box indicating the scope of any other users' intersecting view.

Temporal Awareness

Temporal awareness is related to information about users' past actions. Corresponding features currently provided by single-user applications include history tracking, the ability to save and transfer a file and support for within-document revisitation. An example of the latter is the footprints

scrollbar developed by Alexander et al. [1]. As a user navigates within a document, the application passively records locations where the user maintains focus for more than a few seconds. Under the assumption that people will be likely to view important locations more than once, such places are automatically marked with icons that allow for navigation. We plan to implement visit wears in our application to provide two types of awareness, as determined by the context of the task: two users engaged in parallel search may choose to avoid areas visited by others in the interest of efficiency; or, a user may prefer to search these locations to retrace a user's history.

In some instances, a user may wish to deliberately provide awareness cues about a point of interest (POI), rather than rely on passive system features. Although co-located users can verbalize such information, a collaborative system can record the location for future reference, provide details to other users, and give others an option to quickly and easily navigate to and from a location at their convenience.

One further feature we will explore, inspired by our initial user study, will allow users to collaboratively sketch on a document using their device's touchscreen. While map-based applications currently provide features for calculating efficient routes, a user may desire a simple way to communicate a path of their own choosing, analogous to tracing a route on a paper map with their fingertip. By enabling sketching, we can provide a flexible tool for route-tracing and unlimited other purposes.

Type of Awareness	Visual Augmentation
Present location	Blue wedge
Previous location	Visit wear with red wedge
Point of interest	POI marker with green wedge
Current activity	Sync to view
Past activity	Sync to history record
Scale	Bounding box
Other information	Sketching

Table 1. A summary of awareness features and their corresponding visual augmentation cues.

When adding features to a user interface, such as those to support awareness, it is easy to produce a bloated and cluttered screen, leading to features that are confusing, under-utilized or completely ignored. Our aim is to add functionality that is intuitive and seamlessly integrated with the application environment. Table 1 summarizes the features we will implement and the techniques we will use to accommodate them. Figure 2.a shows the device view along with the conceptual shared workspace.

MOBILE PROJECTION FOR AWARENESS

The recent availability of affordable off-the-shelf mobile projectors offers yet another option for providing



Figure 2. a) The mobile display view and conceptual shared workspace. The second user's navigation region is shown by a bounding box and is indicated by a red wedge. Other wedges reveal the locations of a POI and a visit wear. b) A shared projected display shows the location and scale of each user's view. Clouds provide information about their respective navigation histories.

awareness. For example, Cao et al. [2] use handheld projectors to display a shared workspace. Because their projectors are spatially tracked, the projected images can be united to produce an enlarged view of the same workspace. We intend to extend our shared mobile workspace to the view provided by a single mobile projector, which can be placed on a nearby table while connected to our system. The multiplicity of views provided by this set-up offers the best of both worlds: a large, shared high-level overview, coupled with individual displays for low-level interaction.

Hang et al. [5] have investigated the combined use of a mobile phone with projection for map navigation by single users. Their results indicate that, while people prefer the larger view provided by the projector over the mobile screen only, performance is task-dependent. In particular, task times are penalized by context switches when the device screen and projection show different views. Regardless, projection is a useful tool for providing mobile device users with an extended view of their workspace.

In our multi-user setting, we can augment the projected view with visualizations in a manner similar to that described above for mobile device screens. There are differences, however, that we must consider in choosing which visualizations to use and how to apply them. Wedge, for example, is very useful for expanding the effective size of a small screen, but provides no benefit on the projected view. On the other hand, the utility of the bounding box that represents a user's view location and scale is improved on a projection, because it is visible to all users at any time. The projected view is larger and more resistant to effects of clutter, giving us a larger degree of spatial freedom. For instance, we can conceptually expand the visit wear icons into a point cloud containing a greater breadth of history information (Figure 2.b).

SUMMARY

Shared workspaces open the door to collaborative activity for groups of mobile device users. Awareness is a fundamental requirement of such multi-user software system interfaces. With our collaborative, intra-scalar system, we will explore and empirically test two methods for expanding the effective area of interaction: first, the

integration of information visualization cues into the user interface to bring distant information to the user's fingertips; and second, mobile projection as an avenue for providing group awareness by fitting a large display into a person's pocket. User evaluation in a real-world setting will allow us to discover the best options for endowing collaborative users with several types of awareness: present focus, activity and scale; past navigation history; active and passive information sharing; and interactive sketching.

ACKNOWLEDGEMENTS

We thank Nokia Products Ltd and MITACS Inc for funding.

REFERENCES

1. Alexander, J., Cockburn, A., Fitchett, S., Gutwin, C. and Greenberg, S. Revisiting read wear: analysis, design and evaluation of a footprints scrollbar. In *Proc. CHI 2009*. ACM Press (2009), 1665-1674.
2. Cao, X., Forlines, C. and Balakrishnan, R. Multi-user interaction using handheld projectors. In *Proc. UIST 2007*. ACM Press (2007), 43-52.
3. Greenberg, S., Gutwin, C. and Cockburn, A. Awareness through fisheye views in relaxed-WYSIWIS groupware. In *Proc. GI 1996*, Morgan-Kaufmann (1996), 28-38.
4. Gustafson, S., Baudisch, P., Gutwin, C. and Irani, P. Wedge: Clutter-free visualization of off-screen locations. In *Proc. CHI 2008*, ACM Press (2008), 787-796.
5. Hang, A., Rukzio, E. and Greaves, A. Projector phone: A study of using mobile phones with integrated projector for interaction with maps. *MobileHCI 2008*, ACM (2008), 207-216.
6. Herskovic, V., Ochoa, S.F., Pino, J.A., and Neyem, A. General requirements to design mobile shared workspaces. *CSCWD 2008*, IEEE (2008), 582-587.
7. Morris, M. and Horvitz, E. SearchTogether: An interface for collaborative web search. *UIST 2007*, ACM Press (2007), 3-12.
8. Ware, C. and Lewis, M. The DragMag image magnifier. In *Proc. CHI 1995*, ACM Press (1995), 407-408.

Organizers

Université catholique de Louvain **UCL**



Belgian Lab. of
Computer-Human
Interaction



ISE Interactive
Systems
Everywhere



UNIVERSITAS
Miguel
Hernández



Web Decision
Research Group

Sponsors



ITEA 2



Serenoa

USIXML

USer Interface
eXtended Markup Language



Wallonie



Service public
de Wallonie

