Cross-Surface

Challenges and Opportunities for ‘Bring-Your-Own Device’ (BYOD) in the Wild

2016-1

Proceedings of the second Cross-Surface workshop

Editors: Steven Houben, Jo Vermeulen, Nicolai Marquardt, Johannes Schöning, Clemens Klokmose, Harald Reiterer, Henrik Korsgaard and Mario Schreiner.

Publication: Published online in 2016.

Contact: **Steven Houben**
Department of Computer Science
University College London
Gower Street, London WC1E 6BT
United Kingdom

About

In this workshop, we reviewed and discussed challenges and opportunities for Human-Computer Interaction in relation to cross-surface interaction in the wild based on the bring-your-own-device (BYOD) practice. We brought together researchers and practitioners working on technical infrastructures for cross-surface computing, studies of cross-surface computing in particular domains as well as interaction challenges for introducing cross-surface computing in the wild, all with a particular focus on BYOD. Examples of application domains are: cultural institutions, work places, public libraries, schools and education. Please find more details about the workshop, in the submitted proposal [1]. The workshop was held in conjunction with the 2016 ACM Conference on Human Factors in Computing Systems (CHI), that took place from May 7 to 12 in San Jose, USA.

[1] Steven Houben, Nicolai Marquardt, Jo Vermeulen, Johannes Schöning, Clemens Klokmose, Harald Reiterer, Henrik Korsgaard, and Mario Schreiner. 2016. Cross-Surface: Challenges and Opportunities for 'bring your own device' in the wild. In Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16). ACM, New York, NY, USA, 3366-3372.
DOI: http://dx.doi.org/10.1145/2851581.2856490

Editors

Dr. Steven Houben – *University College London*
Dr. Nicolai Marquardt – *University College London*
Dr. Jo Vermeulen – *University of Calgary*
Prof. Johannes Schöning – *Hasselt University – tUL –iMinds*
Prof. Clemens Klokmose – *Aarhus University*
Prof. Harald Reiterer – *University of Konstanz*
Henrik Korsgaard – Aarhus University
Mario Schreiner – University of Konstanz

Program

09:00	Introduction to workshop by the organizers
09:15	Keynote by Professor Susanne Bødker
10:00	Paper Presentations
10:30	Coffee break
11:00	Case studies and brainstorm
13:00	Lunch
14:00	Present ideas + map out design space
16:00	Coffee break
16:30	Group reflections and panel
17:30	Closing

Keynote

Title: "Holding artifacts in common"

This talk will address multiplicity and artifacts: multi-artifact practices, multi-practice artifacts, multi-user artifacts, and multi-artifact users. I will use both empirical examples and theoretical concepts to further discuss how human users hold artifacts in common, and how this holding in common develops over time

Biography:

Susanne Bødker is professor of Human Computer Interaction at the Computer Science Department, University of Aarhus. Her research areas include participatory design, computer-supported cooperative work and human-computer interaction. Her PhD thesis, Through the Interface—a Human Activity Approach to User Interface Design was an early attempt to present activity theoretical HCI to an international audience. Much of her research since can be seen as consolidation and expansion of this theoretical frame.

List of accepted papers

1. **Towards Cross-device Harmony**
   Timothy Neate [a], Matt Jones [a] and Michael Evans [b]
   [a] Swansea University
   [b] BBC Research
2. **Enhancing Context-Aware Computing through Environmental Awareness**
   Mario Schreiner and Harald Reiterer
   University of Konstanz
3. **XDBrowser: Challenges in Engineering a Next-Generation, Cross-Device Web Browser**
   Michael Nebeling and Anind Dey
   Carnegie Mellon University
4. **Human-oriented Infrastructures for Multi-surface Environments**
   Marc-Emmanuel Perrin, James R. Eagan and Michel Beaudouin-Lafon
   LTCI, Telecom Paristech & CNRS, Inria Université Paris Saclay
5. **Advances and Challenges in Ad-hoc Mobile Tracking for Seamless Interaction across Commodity Devices**
   Haojian Jin [a] and Christian Holz [b]
   [a] Yahoo Labs
   [b] Microsoft Research
6. **The Challenges of Using an Existing Cross-Device Interaction Prototype for Supporting Actual Curation Practices**
   Frederik Brudy [a], Nicolai Marquardt [a], Yvonne Rogers [a], Abigail Sellen [b] and Kenton O'hara [b]
   [a] University College London
   [b] Microsoft Research
7. **The benefits of 'In The Wild' studies for successful introduction of 'Bring Your Own Device' policies in the industry**
   Jens Ziegler, Sebastian Heinze and Leon Urbas
   Technische Universität Dresden
8. **Towards Context-Aware Cross-Device User Interfaces in the Wild**
   Fabio Paternò, Giuseppe Ghiani, Marco Manca
   CNR-ISTI, HIIS Laboratory
9. **Prototyping "In The Wild" Interaction Scenarios With RE/Tk**
   Aneesh P. Tarun [a], Andrea Bellucci [b] and Ali Mazalek [a]
   [a] Synaesthetic Media Lab, Ryerson University
   [b] Universidad Carlos III de Madrid

Towards Cross-device Harmony

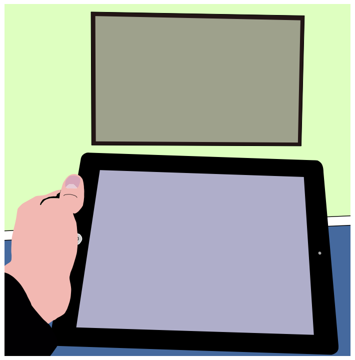


Figure 1: A dual-screen use case – in such scenarios, the users switch their attention between the motion rich TV and the second screen companion content (in this example on the tablet). Though both displays can be considered the primary focus, research shows that generally the more motion rich TV content becomes the 'primary' display. Here the user is holding up the tablet so that the displays meet. However, the device is typically rested on a user's lap (discussed in Figure 2) on the next page.

Timothy Neate
FIT Lab
Swansea University
Swansea, SA2 8PP UK
tdjneate@gmail.com

Matt Jones
FIT Lab
Swansea University
Swansea, SA2 8PP UK
mattjonez@gmail.com

Michael Evans
BBC R&D
Salford, M50 2LH UK
michael.evans@rd.bbc.co.uk

Abstract

In modern home entertainment, our personal devices regularly supplement some 'primary' screen. Such layouts of screens in the living room afford enhanced autonomous browsing, collocated interactions, and give broadcasters the opportunity to enhance TV through multi-device experiences. TV/personal device scenarios are becoming one of the first ubiquitous cross-device situations, and therefore stand as a potential exemplar of the use case. Our research looks at the potential attention bottlenecks in such scenarios, and works towards improving such experiences through informed design of attention.

Author Keywords

Dual-screen; companion content; television; attention; user experience; media; displays; media

ACM Classification Keywords

H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Introduction

The infrastructure for exciting, multi-sensory, cross-device experiences lies in our hands, and adorns our walls – by bringing along our handheld devices, we open up a new world of personalised second screen

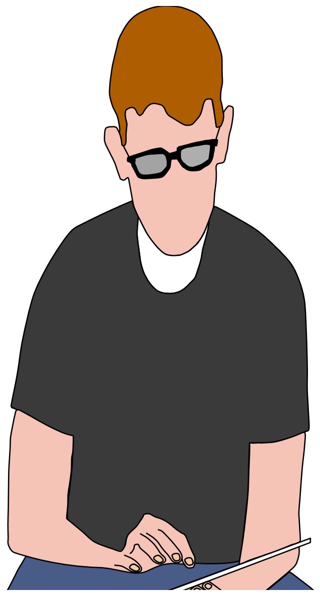


Figure 2: In companion content scenarios users tend to switch attention, as opposed to divide. Typically, users rest the device on their lap, in their peripheral – creating a visual congruence between the foci. This means they must constantly check the device for new content when focusing on the TV, and monitor the auditory stream of the TV for points that pique their interest.

content. The inherent portability, connectability, and sheer computing power of such devices make them prime candidates to supplement larger displays (e.g. Smart TVs) with complementary multimodal stimuli and additional interaction possibilities. Much research has documented how we may appropriate existing technologies to engage in complementary multi-device consumption of media. Therefore, many broadcasters and application developers have began leveraging such research to design between-device media as the next step from beyond traditional linear broadcast.

There are, however, some confounds when it comes to such scenarios – as we divide our attention between devices we create a disjunct between displays. In this workshop paper we reflect on the TV for HCI communities' reflections on attention across devices with an aim to further consider how this applies to cross-device interactions generally.

Context & Brief Background

We increasingly watch TV accompanied by a second screen – we Google tangential information in programmes, social network, or simply browse the web. In 2012, Google suggested that 77% of us second-screened regularly [2] (a statistic later revised up to 87% by Accenture [3]). Predominantly, this growing practice is done on smartphones, tablets, and laptops.

Clearly, then, this use case (Figures 1 and 2) is ubiquitous and (unsurprisingly) broadcasters now wish to lever this enthusiasm for dual-screens to enhance UX. To support dual-screen experiences content providers have began developing companion applications – material developed for second-screen handheld devices that run alongside a TV programme, providing relevant facts, quizzes, and social media content. With the increasing proliferation of internet-driven media and object based broadcasting (see [6]), this area of multi-device media is quickly accelerating and therefore people are considering its fundamental design.

There are many recent commercial examples of programmes which utilise a second device with supporting material (e.g. [4]). In addition to this, much literature in the HCI for TV community explores this scenario from an academic standpoint, to support content creators. Further, for such cases, some work (e.g. Brown et al. [1]) has investigated how attention switches for dual-screen companion content.

Currently dual-screen experiences require users to manage their own attention across displays – they are often overloaded with information to the point where they cannot engage with content as its designers envisioned. Moreover, the visual disjunct between the foci (see Figure 2) can mean users fixating on one display. This, coupled with the inherent cost of display switching [8], negatively impacts on UX.

Due to the lack of thorough investigation on this topic, the dual-screen experience is fundamentally *undesigned*. Therefore, our research looks at how we may better design cross-device media to encompass attention, and design towards experiences in which the displays are complementary, and harmonious. To this end we have been building on the literature by conducting preliminary interviews, designing interventions, and conducting systematic studies with users. We, by working with our industry part-

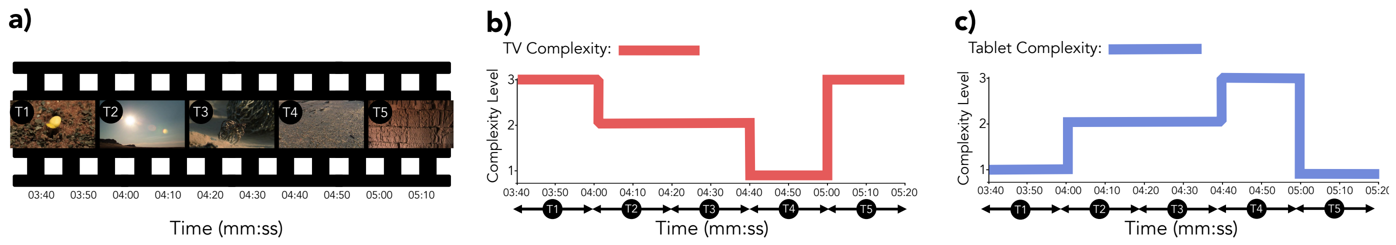


Figure 3: Content *curation* process: a) video is broken down into 20 second time slices; b) complexity of TV content is determined; c) complexity of tablet content is determined by inverting the TV complexity value. Figure from [5]

ners (the BBC), are refining our idea of what creates a positive multi-device media experience through proper cross-device attention management.

Work so Far

To establish the current experience of the dual-screen use case we conducted interviews with participants about their usage habits. In general we found early on that there are some clear attention bottlenecks. For example, when users view content on their phones (be it social media or related web content) they, generally, to some degree need to make some sacrifice – to try to 'block out' one device, in favour of another. Though we can monitor content in our peripheral, in either the audio of visual domain, it requires extraneous effort.

To explore this more systematically we conducted a lab study in which participants watched TV accompanied by companion content – related textual and graphical information (e.g. Figure 4). The independent variables were the companion device's textual and graphical complexity. In general, we saw increased visual attention and increased effort required as the complexity was increased, more so for textual information. We then associated this with our qualitative data from the participants to learn better what TV content requires more effort as the second screen complexity is increased.

Using our qualitative data we established a set of observable and codeable behaviours in the presence of varying complexity, and from this developed systems in which the complexity on a tablet computer adapted (we term this *curated*) based on the complexity of the TV (see Figure 3). We then compared this to a baseline and an adaptable (by the user) case and found that such methods complement more 'lean back' companion experiences. In addition, we found a degree of variability in the participants who enjoyed the adaptable UI. Some, who were more engaged with the second screen material actively increased the complexity of the content. Whereas others, who wanted to engage with the companion content less, turned down the complexity so that they could gain a better gist of the materials. These studies culminated in the work presented in [5].

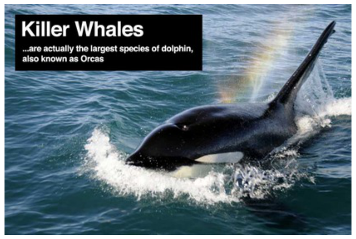

Killer Whales
...are actually the largest species of dolphin, also known as Orcas

Figure 4: Example of a simple piece of trivia typical of companion content. The users are free to swipe through such trivia or view as a slide show. This was used in the mediating attention experiment, and in the work of Brown et al. [1].

Figure 5: Example on an on-TV notification. Such calls to action are often found in current interactive TV, for example encouraging users to follow a hashtag, or press the (BBC) red button for more information.

In our work, we have also considered how we may move a participants gaze between displays. For example, in [7] we looked at a variety of methods to notify users we looked at how we may mediate attention between the foci. For example, we found that if we wish to command attention shifts quickly and effectively we should use peripheral stimuli (on the tablet) of either audio or visual medium. And that if we wish to permit some degree of autonomy to the users calls to action on the TV work effectively (Figure 5). Further, we found that in contexts with updating information that users strongly preferred their attention to be mediated to some extent, to avoid over/under-attendance of a device.

In summary, we have so far looked at key areas of concern in the dual-device media scenario, and have investigated solutions to alleviate. In the future, we hope to reflect on how we may glean insight from the increasingly ubiquitous use case that is multi-device TV, towards considering how we may better design complementary UI for general (non-media) cross-device scenarios.

References

[1] Andy Brown, Michael Evans, Caroline Jay, Maxine Glancy, Rhianne Jones, and Simon Harper. 2014. HCI over Multiple Screens. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14)*. ACM, New York, NY, USA, 665–674. DOI:http://dx.doi.org/10.1145/2559206.2578869

[2] Google. 2012. The New Multi-screen World:Understanding Cross-platorm Consumer Behavior. Google. http://goo.gl/xdbOe1

[3] Gavin Mann, Francesco Venturini, Robin Murdoch, Bikash Mishra, Gemma Moorby, and Bouchra Carlier. 2015. *Digital Video and the Connected Consumer*. Technical Report. Accenture.

[4] NBCUniversal. 2014. Heroes Reborn. Online. (2014). Retrieved 14/09/15 from http://goo.gl/yl93Qb.

[5] Timothy Neate, Michael Evans, and Matt Jones. 2016. Designing Visual Complexity for Dual-screen Media. In *(to appear) proc. CHI 2016*. ACM. DOI:http://dx.doi.org/10.1145/2858036.2858112

[6] Timothy Neate, Matt Jones, and Michael Evans. 2014. Future Media: The Role of HCI in Broadcast. *Seventh York Doctoral Symposium on Computer Science & Electronics* (2014), 75.

[7] Timothy Neate, Matt Jones, and Michael Evans. 2015. Mediating Attention for Second Screen Companion Content. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3103–3106. DOI:http://dx.doi.org/10.1145/2702123.2702278

[8] Umar Rashid, Miguel A. Nacenta, and Aaron Quigley. 2012. The Cost of Display Switching: A Comparison of Mobile, Large Display and Hybrid UI Configurations. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12)*. ACM, New York, NY, USA, 99–106. DOI:http://dx.doi.org/10.1145/2254556.2254577

Enhancing Context-Aware Computing through Environmental Awareness

Mario Schreiner
University of Konstanz
Konstanz, Germany
mario.schreiner@uni.kn

Harald Reiterer
University of Konstanz
Konstanz, Germany
harald.reiterer@uni.kn

Abstract

To counteract the rising complexity of mobile systems, context-aware computing is important to determine user intent and adapt devices accordingly. This position paper proposes to improve context detection by harnessing the fact that most modern devices have become detectable, networked beacons, allowing other devices to pick up their virtual presences and use these presences as indicators for the current physical environment. This paper proposes to improve context detection by a) analysing surrounding devices and b) communicating with nearby devices to exchange environmental status information. The paper describes the approach - both conceptually as well as technically - and describes possible use cases and limitations. We believe that further research in this direction can improve context detection in the future dramatically.

Author Keywords

context-aware computing; environment; context; bluetooth

ACM Classification Keywords

H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Introduction

Modern devices, and in particular mobile devices, have rapidly gotten more powerful over the last couple of years.

With this, they gained new features but also became much more complex to use. For mobile interaction designers, this introduces new challenges: Users tend to use mobile devices to achieve a single task in a short time[2] with their cognitive resources being very limited[3]. But with complexity, the time to solve a task increases as well. As a prominent way to tackle this issue, research has looked into context-aware computing[1]. Context is a proxy for human intent[5] and as such helps to adapt applications to better aid the user. But detecting context is no trivial task: The components that make up “context” are manifold, ranging from location and time over lighting, temperature and noise to people and objects that surround us[1].

Current devices use a limited set of sensor hardware - such as GPS for location or accelerometer to detect motion - in combination with machine learning algorithms to determine context. While useful, this kind of context detection is very limited and insufficient to represent the complexity of people’s life. Additional information about the surrounding could help devices to improve their context-aware behaviour. This position paper will focus on the possibilities of detecting the user’s physical environment. We propose the use of short-range wireless technologies (i.e. Bluetooth Low Energy) to scan the user’s current surrounding. By learning about repeating occurrences of combinations of devices (i.e. by using machine learning algorithms), and combining these occurrences with other sensor data (such as GPS, contact list data or the active application), it is our believe that context detection could be largely improved. The basic approach is not novel in itself - for example, ContextPhone has described the use of “physical environment, including surrounding Bluetooth devices”[4] as a possible sensor. We think, though, that this kind of sensing has tremendous potential to improve context-aware computing and is not sufficiently explored yet, in particular in light of recent advances in technology. Therefore, this paper will describe an approach of a) context sensing using surrounding devices and b) retrieving extended environmental information through communication with nearby devices. We will further detail different use cases where this detection can improve application behaviour and describe the limitations and technical difficulties in making this approach a reality.

Context through Surrounding Devices

With the advance of Weiser’s vision of ubiquitous computing and the emergence of the Internet of Things, almost any modern device communicates with the outside world. Wi-Fi, Bluetooth, and NFC have become prominent communication channels and are found in smartphones, tablets, and computers, but also watches, light bulbs, fridges, TVs, and many more devices. It is likely this trend will continue in the future to incorporate even more types of devices. We believe this fact can be harnessed by scanning and learning about the surrounding of a device to derive context information.

For example, consider workplace detection using location. While suited for regular work at a single workplace, such a detection will fail for a travelling salesman or for extraordinary events, such as external meetings or a dinner. Using surrounding devices, the detection of the work context becomes much more adaptable. During work-related events the user is surrounded by a similar circle of devices: The personal devices of co-workers. Based on this, work events can then be correctly classified. By querying additional sensors, such as time and location, the context can be further narrowed down: For example, at a restaurant in the evening, the work event becomes a dinner with colleagues. Using this knowledge, devices can adapt, e.g. by turning off non-crucial notifications and giving quick access to culinary information such as wine ratings.

In contrast, consider being at the same restaurant at the same time but being surrounded by close friends: Here, all notifications would be enabled, taken photos could be automatically shared, and, when leaving, the location of bars with long opening hours can be suggested.

Environmental detection can also provide valuable meta-data for artefacts. While scribbling down digital notes during a meeting, the artefact can be automatically tagged, e.g. with the project name. When taking a photo, face detection can be improved based on the people present and the photo can further be tagged with the people *not* seen on the photo. Advanced activity tracking, such as determining how long the user spent on his work computer or how often he went up to get coffee, can provide valuable insights for a healthier lifestyle.

Detecting surrounding devices can also provide an indicator for the current level of publicity. When interacting with a large display, the number and type of nearby people can influence the displayed information: With no one nearby, personal information such as the next appointment can be disclosed. At a public place with lots of people nearby, only limited information, such as the time to the next appointment, but no details, are shown.

It is our believe that an environment-based approach to context detection will lead to more adaptable and robust results and performs better at translating to actual user intent.

Enriching Environmental Information

So far, the focus has been on the presence or absence of surrounding devices to determine context. Using modern communication technologies, environmental information can be enriched by allowing devices to retrieve additional information from their surrounding. This information can vary: Most importantly, the type of nearby devices can be retrieved. Devices can also allow to retrieve their current state, for example the currently running movie on a TV or the measured temperature on a thermostat. This enables an even deeper integration with the environment. This kind of exchange can be performed through the same ad hoc communication channels as the detection of devices, such as Bluetooth or NFC.

Such advanced environmental information further enhance a device's ability to adapt. For example, retrieving the currently running movie from a TV allows for second-screen information on the smartphone, such as the actors in the current scene. And knowing the exact type of training tool the user is currently working out at enables advanced fitness tracking through accelerometer and heart rate sensors.

We think that a standardised exchange of local environmental information combined with the refined context detection described previously can enable entirely novel ways of how our devices sense their surrounding and adapt to it.

Technological Approach

Most of today's off-the-shelf consumer devices feature the hardware required for environmental detection. Most prominently, Bluetooth Low Energy is built into almost any modern device. Even home automation devices (such as light bulbs) are often Bluetooth-enabled. Additionally, technologies such as NFC have become more commonly available and could enable cheaper sensing of devices in the future.

Detection of surrounding devices can be done with simple Bluetooth scans. Paired with additional sensors (such as GPS), using information from the user's contacts list, and combined with fuzzy machine learning algorithms, a robust detection of context can become possible. For simplicity, for abstraction and to protect user information such a detection should be implemented on an OS-level, handing only high-level context information to applications. Detailed information, such as the exact devices and people in a user's surrounding, should not be handed to applications.

Exchanging environmental information can be done using the same technologies. Developing a common protocol amongst all the different devices in our environment is required in order to achieve a seamless communication between these devices. This is difficult, in particular considering the large variety of possible devices. A high-level protocol that allows devices to register predefined device types and capabilities, similar to how most home automation protocols work, could tackle this issue, but also restricts flexibility. Applications could register for the desired device types and properties to receive updates about them from the OS. Mixing such a static protocol, implemented on an OS-level, with the ability to exchange a limited amount of custom data, implemented at application level, could enable a trade-off between abstraction and flexibility. For example, this would allow TVs to advertise themselves as a screen device, but also broadcast the current movie for applications interested in this information. Nonetheless, the communication between all different types of devices remains the largest issue in this approach, and an issue in cross-device interaction and ubiquitous computing in general.

Limitations

In a real-world implementation, mapping of virtual device presences to people, which is required for some scenarios, can be difficult. For example, Bluetooth Low Energy alternates the MAC address of a device regularly to make it untraceable. And even with traceable devices, matching devices to contact list entries can still be a difficult task. Also, such a mapping can be considered a security risk and must be implemented with care to ensure user privacy. The excessive data exchange between devices could furthermore exceed the bandwidth possibilities of Bluetooth and a large amount of signals could lead to interference. Future technologies might be able to solve such technical issues.

Furthermore, context remains only a proxy for human intent, and environmental detection does neither guarantee that the context is correctly determined nor that the context is correctly translated into intent. Certain scenarios will remain difficult to detect. This, of course, is a general issue with context detection and can only be improved by further research in this direction, development of better sensors and improvement of algorithms.

REFERENCES

1. Anind K. Dey, Gregory D. Abowd, and Daniel Salber. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-aware Applications. *Hum.-Comput. Interact.* 16, 2 (Dec. 2001), 97–166.
2. Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. 2010. Diversity in Smartphone Usage *(MobiSys '10)*. 179–194.
3. Antti Oulasvirta, Sakari Tamminen, Virpi Roto, and Jaana Kuorelahti. 2005. Interaction in 4-second Bursts: The Fragmented Nature of Attentional Resources in Mobile HCI *(CHI '05)*. 919–928.
4. Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen. 2005. ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. *IEEE Pervasive Computing* 4, 2 (2005), 51–59.
5. B. Schilit, N. Adams, and R. Want. 1994. Context-Aware Computing Applications *(WMCSA '94)*. 85–90.

XDBrowser: Challenges in Designing a Cross-Device Web Browser

Michael Nebeling
Anind K. Dey
Human-Computer Interaction Institute
Carnegie Mellon University
{ mnebelin, anind }@cs.cmu.edu

Abstract

Recent research has focused on web developer toolkits for distributed multi-device user interfaces. We investigate a new solution, *XDBrowser*, where the web browser itself is aware of and able to use multiple devices in parallel. This paper discusses how XDBrowser's interaction and implementation techniques help overcome many challenges of BYOD-based interaction given its increasing ability to adapt existing web interfaces and browsers for cross-device use.

Author Keywords

multibrowsing; semi-automatic distribution; hybrid browser

Introduction

State-of-the-art web browsers have added support for keeping the browser history, bookmarks and settings in sync so that users can use multiple personal devices for browsing the web. This partly addresses the need for more seamless multi-device interaction identified in earlier studies on information work and web use [3, 9, 10, 14]. More recent studies [11, 15] find an increased need to better support parallel device usage so that users can flexibly distribute tasks between devices, taking into account both device capabilities and user preferences. However, there is no native browser support for using multiple devices in parallel. Instead, special cross-device development toolkits and modifications to existing web interface code are required [4, 7, 12, 17].

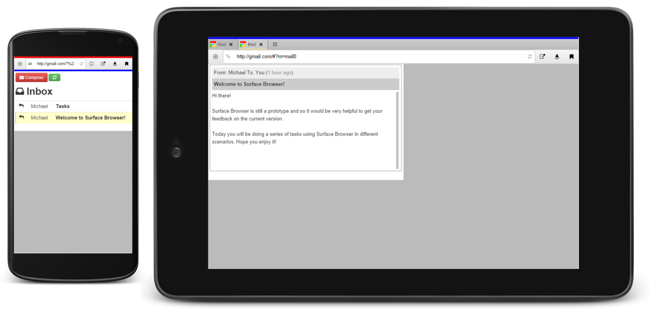
As part of the *XDBrowser* project, we are investigating how existing web browsers can be extended to support rich interactions and parallelism in cross-device use. A first prototype of XDBrowser [11] implemented new end-user customization tools for re-authoring existing web pages so that they can be distributed and synchronized between multiple devices. This prototype was then used to conduct a study on user-defined cross-device web page designs for a given set of five popular web applications. The study generated 144 cross-device designs that can be distilled down to seven core design patterns. While the first version of XDBrowser proved very useful for that study, we are currently working on a new version with two major improvements.

First, the existence of patterns suggests that part of the manual re-authoring process could be automated. Providing automated support for adaptation could be beneficial to users since users could browse new pages they have not visited before, without having to first customize them for cross-device use, and could instead choose from available patterns and switch the design depending on the task. The primary challenge then becomes to detect the desired pattern from only minimal user interaction and distribute existing web pages without prior modification by developers.

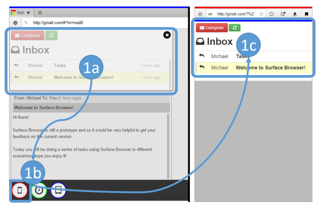
Second, while the chosen architecture was sufficient to enable our end-user customization study, there are technical limitations that we are currently addressing. We discuss the benefits and limitations of our *browser-in-browser* implementation technique which, rather than building on common browser extensions and plug-ins, overloads the browser with a full-screen browser interface that hides the host browser. It is compatible with a wide variety of devices and existing web browsers even if they do not support extensions, but still lacks some advantages of browser extensions that we aim to overcome with a new hybrid approach.

Semi-Automatic Distribution Techniques

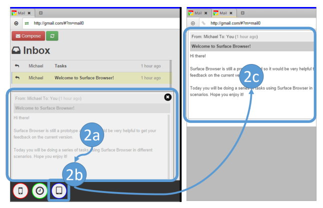
Our end-user customization study using XDBrowser produced 144 desirable cross-device designs leading to seven distinct patterns. The full description of the study and results can be found in [11]. To illustrate the extensions we are designing for XDBrowser, let us focus on two patterns.



(a) Remote control of mail reading pane on the tablet from the phone



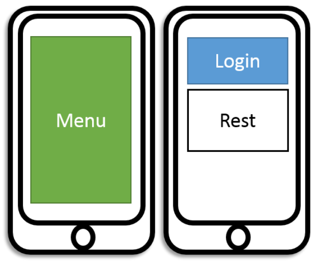
(b) (1a) select inbox on tablet; (1b) push inbox from tablet to phone; (1c) inbox on phone



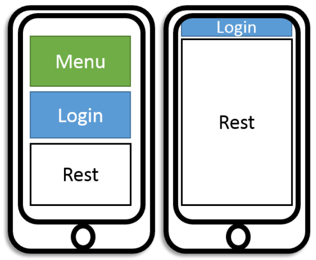
(c) (2a) select mail on tablet; (2b) push mail to tablet; (2c) mail on tablet

Figure 1: Six-step end-user customization in the first version of XDBrowser. Pattern-based semi-automatic distribution achieves the same result simply by double-tapping the inbox page element.

Consider the mail application in Figure 1 with the inbox and the reading pane distributed for *remote-control* from the phone. To distribute the elements in this way, the first version of XDBrowser required users to perform a series

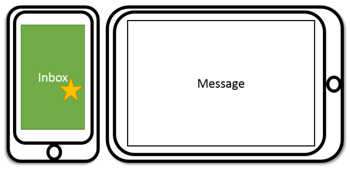


(a) Remote-control

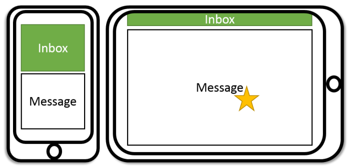


(b) Overview+detail

Figure 3: Patterns for using one device to control the other or for overview and the other for detail



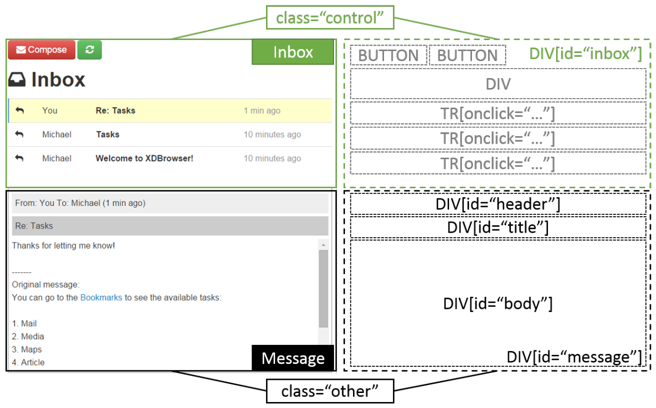
(a) Remote-control



(b) Overview+detail

Figure 4: Segmentation and triggered patterns (star marks element invoked by double-tap)

of manual selections and operations to push selected elements between the devices. The goal of our new techniques is to reduce this effort to a single interaction. The main inspiration for our approach comes from modern browser support for *double-tap to zoom* on mobile devices to view selected portions of the page in more detail. Our idea is to allow users to double-tap the content they want zoomed and make use of connected devices to automatically distribute the page elements pushed out of the browser viewport when zooming the content on the current device.



From: You To: Michael (1 min ago)

Re: Tasks

Thanks for letting me know!

Original message:

You can go to the Bookmarks to see the available tasks:

1. Mail
2. Media
3. Maps
4. Article

Figure 2: Classification of DOM nodes for mail application

Figure 2 shows a breakdown of the main page elements relevant for two of the patterns we want to be able to activate, *remote-control* and *overview+detail* (Figure 3).

Since selecting a message in the inbox controls which message is shown in the reading pane, the *remote-control* pattern should become active when the user double-taps the Inbox element on the phone. As a result, the Inbox element should be kept on the phone and the other elements moved to the tablet (Figure 4(a)). Looking at how the Inbox element is constructed by nesting different types of HTML DOM nodes, from the node that received the double-tap event, we would need to traverse the DOM tree upwards until we find the node wrapping the Inbox element, i.e., the `DIV` with `id` "inbox" (Figure 2). This node is characterized by having an `id` attribute and containing a set of `BUTTON` and `LI` nodes with `onclick` event handlers. Once we have found this node, we can extract it and hide all other elements on the current device. On connected devices, we hide this node and show all other elements instead.

If the user double-taps the Message element, *overview+detail* should become active. As a result, the reading pane should be zoomed on the current device so that it fills the browser viewport (Figure 4(b)). In this case, the Message element is constructed from several nested `DIV` nodes, all of which again have an `id` attribute. Zooming any of them leads to the same result as zooming the "message" `DIV` directly.

Common web page segmentation techniques split the page into smaller blocks of content elements based on text and structure analysis, DOM hierarchy and layout information [1, 2, 6, 16]. These elements can then be extracted as a group. However, they typically require full analysis of the page content to do so, which can be computationally expensive, especially on less powerful devices. Based on insights from our study such as above, we are developing an interaction-based approach that does not require full segmentation and only involves relevant parts of the DOM tree. It constructs the DOM path to the invoked element, performs a classification into three types of elements—*control*, *input*, or *other*—by traversing the DOM hierarchy upwards, and activates the pattern depending on the type of element found by extracting relevant nodes and showing them on one device and hiding them on connected devices.

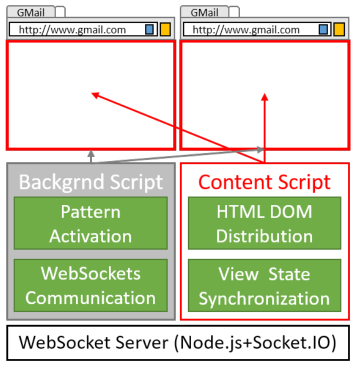


Figure 5: XDBrowser as a Chrome extension using a Node.js-based client-server architecture with WebSocket communication

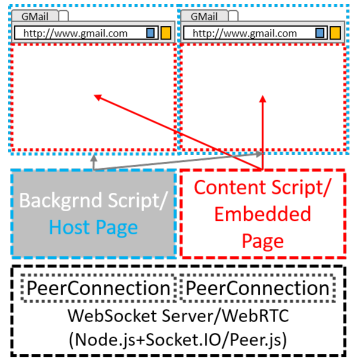


Figure 6: Hybrid approach resorting to *browser-in-browser* and using Node.js or Peer.js for client-server or peer-to-peer communication

In our first evaluations with a set of 50 top-ranked sites by Alexa from 10 different genres, our simple classification proved sufficient to support semi-automatic distribution.

Hybrid Approach to Cross-Device Browsing

We developed two implementations of XDBrowser using different architectures. First, we developed an extension of the popular Chrome web browser, making it possible to run on both Desktop and Android devices, including tablets and smartphones. Note that on mobile devices we built on the Crosswalk web runtime[1] to embed the latest Chrome and add support for extensions which are not supported in Chrome for Android. The architecture is shown in Figure 5. XDBrowser connects multiple browser windows, either running on different devices or as multiple Chrome instances on the same device. The implementation is divided into a client-side *background script* executed once per browser window—it is used to activate patterns and maintain a WebSocket connection between multiple browser windows through the server; a *content script* executed for every page loaded into a tab—it is used to inject our DOM distribution and view state synchronization methods; and a Node.js server using Socket.IO for WebSocket communication. The Chrome API is used for DOM manipulation, Hammer.js for touch events, and Zoom.js to magnify web page elements that were extracted using our classification.

Our second implementation is "a browser within a browser". The host browser, however, is not visible to the user since XDBrowser runs in fullscreen. This implementation has the advantage that iOS mobile devices and even Android Wear smartwatches on which Chrome is not available can be supported. Here, we use WIB[2] as the host browser instead. The client side of our second implementation uses responsive web design based on HTML5, CSS3 and jQuery to adapt to a wide range of devices including smartwatches and phones, tablets, desktops/laptops, and tabletops. Using only native web technologies allows it, in principle, to run on any web-enabled device with modern browser support. The server side is the same as the first implementation.

The two implementations have their pros and cons. The first is compatible with devices running Chrome and basically any web site, but requires installation of a browser plugin or special client. The second supports an even larger set of devices and any browser, but embeds web sites via iframes. Many top sites forbid iframe embedding and browsers prevent cross-site scripting, but a proxy server fixes this [5]. For sites that maintain a session, the user needs to login on each device, but a remote-control architecture [13] or shared virtual browser such as PhantomJS resolves this [8].

We are working on a hybrid approach that combines the best of these techniques (Figure 6). Using the common server side and parts of the Chrome API as the common interface, XDBrowser switches to the *browser-in-browser* approach if the host browser is not Chrome or the XDBrowser extension not installed. Note that the server can be embedded within the browser. We have also experimented with Peer.js rather than Node.js for peer-to-peer communication via WebRTC, which avoids the server after connection brokering and is especially useful for watch-based scenarios.

Acknowledgments

Michael Nebeling started this research with support from ETH Zurich. It was continued at Carnegie Mellon University with two mobility grants from the Swiss National Science Foundation, P300P2_154571 and P300P2_164646.

[1] https://crosswalk-project.org
[2] https://play.google.com/store/apps/details?id=com.appfour.wearbrowser

References

[1] Orkut Buyukkokten, Hector Garcia-Molina, Andreas Paepcke, and Terry Winograd. 2000. Power Browser: Efficient Web Browsing for PDAs. In *Proc. CHI*.

[2] Yu Chen, Wei-Ying Ma, and HongJiang Zhang. 2003. Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices. In *Proc. WWW*.

[3] David Dearman and Jeffrey S. Pierce. 2008. It's on my other computer!: computing with multiple devices. In *Proc. CHI*.

[4] Luca Frosini and Fabio Paternò. 2014. User interface distribution in multi-device and multi-user environments with dynamically migrating engines. In *Proc. EICS*.

[5] Giuseppe Ghiani, Fabio Paternò, and Carmen Santoro. 2012. Push and Pull of Web User Interfaces in Multi-Device Environments. In *Proc. AVI*.

[6] Gen Hattori, Keiichiro Hoashi, Kazunori Matsumoto, and Fumiaki Sugaya. 2007. Robust Web Page Segmentation for Mobile Terminal Using Content-Distances and Page Layout Information. In *Proc. WWW*.

[7] Tommi Heikkinen, Jorge Goncalves, Vassilis Kostakos, Ivan Elhart, and Timo Ojala. 2014. Tandem Browsing Toolkit: Distributed Multi-Display Interfaces with Web Technologies. In *Proc. PerDis*.

[8] Maria Husmann, Michael Nebeling, Stefano Pongelli, and Moira C. Norrie. 2014. MultiMasher: Providing Architectural Support and Visual Tools for Multi-Device Mashups. In *Proc. WISE*.

[9] Shaun K. Kane, Amy K. Karlson, Brian Meyers, Paul Johns, Andy Jacobs, and Greg Smith. 2009. Exploring Cross-Device Web Use on PCs and Mobile Devices. In *Proc. INTERACT*.

[10] Amy K. Karlson, Shamsi T. Iqbal, Brian Meyers, Gonzalo Ramos, Kathy Lee, and John C. Tang. 2010. Mobile Taskflow in Context: A Screenshot Study of Smartphone Usage. In *Proc. CHI*.

[11] Michael Nebeling and Anind K. Dey. 2016. XDBrowser: User-Defined Cross-Device Web Page Designs. In *Proc. CHI*.

[12] Michael Nebeling, Theano Mintsi, Maria Husmann, and Moira C. Norrie. 2014. Interactive Development of Cross-Device User Interfaces. In *Proc. CHI*.

[13] Jeffrey Nichols, Zhigang Hua, and John Barton. 2008. Highlight: A System for Creating and Deploying Mobile Web Applications. In *Proc. UIST*.

[14] Antti Oulasvirta and Lauri Sumari. 2007. Mobile Kits and Laptop Trays: Managing Multiple Devices in Mobile Information Work. In *Proc. CHI*.

[15] Stephanie Santosa and Daniel Wigdor. 2013. A Field Study of Multi-Device Workflows in Distributed Workspaces. In *Proc. UbiComp*.

[16] Evan Schrier, Mira Dontcheva, Charles E. Jacobs, Geraldine Wade, and David Salesin. 2008. Adaptive Layout for Dynamically Aggregated Documents. In *Proc. IUI*.

[17] Jishuo Yang and Daniel Wigdor. 2014. Panelrama: Enabling Easy Specification of Cross-Device Web Applications. In *Proc. CHI*.

Human-oriented Infrastructures for Multi-surface Environments

Marc-Emmanuel Perrin
LTCI
Télécom ParisTech & CNRS
Université Paris-Saclay
75013 Paris, France
meperrin@telecom-paristech.fr

James R. Eagan
LTCI
Télécom ParisTech & CNRS
Université Paris-Saclay
75013 Paris, France
james.eagan@telecom-paristech.fr

Michel Beaudouin-Lafon
LRI
Univ. Paris-Sud, CNRS, Inria
Université Paris-Saclay
91405 Orsay Cedex, France
mbl@lri.fr

Abstract

From offices to public spaces, dynamic multi-surface environments that can leverage the devices that users carry with them are becoming more common. However these environments are often implicit and therefore hard to discover, as are the multi-device interactions that they support. This position paper outlines the challenges that designers of multi-surface environments face to improve service discoverability, to support interactions that leverage users' devices, and to provide software tools to design and develop cross-devices applications.

Author Keywords

multi-surface interaction, wall-sized displays, infrastructures, distributed architectures

Introduction

Recent technological advances have rendered connected, personal devices much more ubiquitous. It is common for users to carry some combination of smartphones, tablets, laptops, and smart jewelry such as watches, bracelets, and rings. Similarly, physical infrastructures such as interactive wall-sized displays and tabletops, as well as systems that track the locations of users and devices, are becoming more prevalent. Advanced users, such as scientists and data analysts, increasingly incorporate such environments into their work. In the workplace, smart meeting rooms

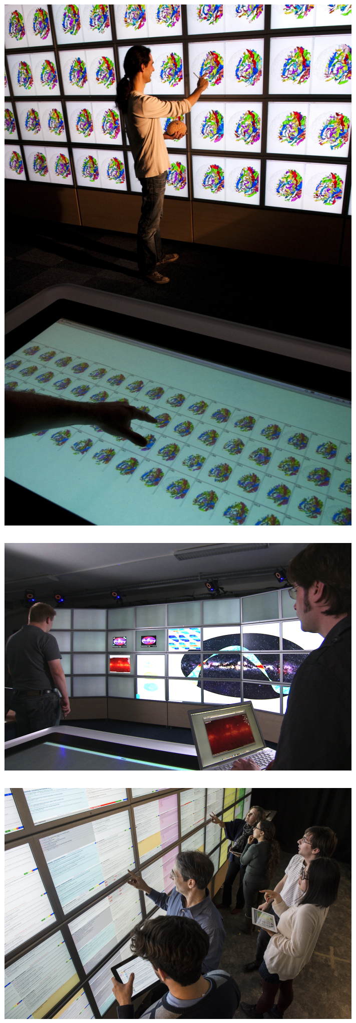


Figure 1: Multi-surface interaction in the WILD Room [2].

are becoming more common. Even in everyday life, such interactive environments are finding their way into shopping malls and airports.

Each of these multi-surface environments, however, affords different interaction styles with different kinds of devices. Users might be able to extend the existing environment to include their own devices and data, or extend their own devices to appropriate the physical infrastructure. Discovering whether such capabilities are available and how to actually perform such operations remain unsolved problems: there are no well-established conceptual models for such distributed interfaces, and therefore users cannot integrate them into their own mental models.

Moreover, building such multi-surface applications, with interactions well-suited to users' needs, requires mastering not only the details of the application domain, but also the intricacies of low-level technologies. While it is possible to create cross-device applications with existing models, they are still too complex to build and often too brittle. To create multi-surface applications, developers need new abstractions for discoverability, management of shared data models, network communication, and adaptability to heterogeneous devices. Interaction designers, on the other hand, need more expressive models based on post-WIMP conceptual frameworks, such as instrumental interaction [1].

Multi-surface environments

The diversity of a users' devices and contexts of use results in a variety of multi-surface environments. We identify three broad categories of multi-surface environments: dedicated platforms, smart meeting rooms and public spaces.

Our work so far has focused on dedicated multi-surface environments in which interaction, processing and rendering may take place on different devices. Such distributed environments take the form of a fixed, dedicated infrastructure such as the WILD room [2] which combines wall-sized displays, motion capture systems, and data and computation clusters with heterogeneous portable devices that users may bring with them (Fig. 1). Since each platform may have specific capabilities, e.g. 3D display or multitouch wall-sized display, not available on other platforms, a design challenge is to create software that takes advantage of the specific capabilities of the platform yet can be ported to other environments.

Less extreme multi-surface environments, such as "smart" meeting rooms, may also create user-centered spaces that leverage cross-device interactions enabled by, e.g., Apple Handoff, Hamilton & Wigdor's Conductor [3], or Webstrates [4]. Unlike dedicated platforms, these environments are more standardized. Most work in this field has focused on interaction involving smartphones, tablets, tabletops and wall-sized displays. However, as wearable devices become more powerful and affordable, users will also want those devices to support new interaction capabilities in such environments.

Finally, multi-surface environments may be experienced in everyday life. In contexts such as shopping malls, airports and train stations, interactive ads and information displays are becoming more common. A user may search for a particular shop at a mall kiosk or consult an interactive subway map in a station. Since these environments are public, users may have a variety of kinds of devices. Interaction must therefore be reduced to the lowest common denominator to accomodate as many users as possible.

User perspective

We are interested primarily in multi-surface environments in which users can dynamically combine interaction between a fixed infrastructure and their own devices. For example, a user may extend the capabilities of her devices to take advantage of the local infrastructure or to enrich the local infrastructure with her own data or device capabilities. In either case, the user must first discover and pair the available devices and services before she can appropriate the new interaction space created by the combination of her own devices and the environment. Thus, the discovery and pairing processes must have low viscosity. Currently, if a user wishes to, e.g., interact with a subway map from his phone, he could easily spend more time connecting the devices together than actually interacting with the map.

Once the user's devices are connected, they create an implicit multi-surface environment that provides interaction capabilities and possibly access to data. To exploit these capabilities, the user must be aware of their existence and understand what interactions are possible and what their effects are. This requires proper feedforward and feedback to make interaction more discoverable.

For example, consider a simple task, such as editing a document on a shared display with other people in the room: how would the user discover that his device can be used to share the document with the display and notify other users that they can interact with it concurrently? What should the interaction look like to achieve that particular task? For now, the commonly used interaction models do not encompass such unified, seamless cross-device interactions. This results in ad-hoc solutions, mixing different interaction metaphors.

Technological perspective

Multi-surface interaction typically involves several devices in the interaction loop, requiring mechanisms to maintain and synchronize a consistent state across the devices as well as manage events coming from multiple devices. However, since current laptops, tablets, and smartphones were designed for standalone use, their operating systems and user interface toolkits do not provide adequate support for multi-surface environments.

The dynamicity of such environments, where devices can join and leave at any time, adds to the challenge. New software architectures and programming models are clearly needed to support these highly-distributed, dynamic and uncertain environments in order to let software developers build cross-surface applications that provide expressive and consistent interactions from the user's perspective. Conductor [3] is an example of a step in the right direction.

Our recent work on Webstrates [4] explores an alternative approach. Webstrates turn the web into shared, dynamic media: the pages served by a Webstrates server are automatically synchronized across the clients viewing them when any client makes a change. Pages can *transclude* the content of other pages [5], creating a host of possibilities to display and manipulate content. For example, an editor is a webstrate that transcludes a set of editing instruments (each is a webstrate) as well as a content page (a webstrate too). The instruments contain code that can edit the shared content. Users can configure and personalize their environment, as well as create and exchange content (including editors and instruments).

By using the web as infrastructure, any web-capable device can access Webstrates and users are immediately familiar with the basic interaction model. We have

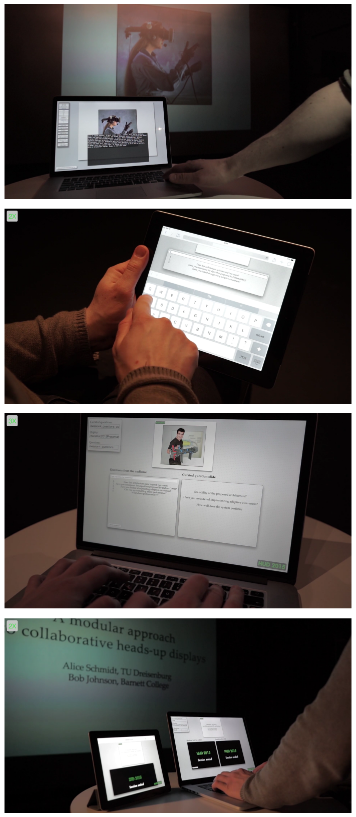

A modular approach
collaborative heads-up displays
Alice Schmidt, TU Dresenburg
Bob Johnson, Barnett College

Figure 2: A slideshow controlled with Webstrates [4]. Top to bottom: presenter view, audience view, moderator view, session chair view.

created a number of scenarios that involve multiple interactive surfaces [4], such as a slide presentation with audience participation and session-chair control (Fig. 2). However, more work is needed to better support interactions that involve multiple devices. We also want to extend Webstrates to devices from the internet-of-things that do not support web protocols.

Three Challenges

We see three primary challenges faced by designers of cross-device applications:

Discoverability How can a user easily discover that pairing one of his devices with the environment might bring new interactions capabilities? Typical approaches include directing the user to a captive web portal, but this requires explicit user actions and several steps. For simple actions such as querying a display for a subway route and downloading it to their smartphone, the cost of discovery and pairing must be minimal or users will simply not use these features.

Interaction How can a user interact with multiple surfaces? Creating interactions that span several devices in a distributed environment is complex, even for simple ones, due to the dynamism of the infrastructure and the need to coordinate and synchronize multiple devices. From the users' perspective, it is critical to create a consistent conceptual model so that users can concentrate on the task at hand rather than struggle to understand what interactions are possible and how to perform them.

Software What architectures and tools should we provide to developers so they can build such applications more easily? WIMP and Post-WIMP toolkits and interface builders help developers create widgets and assemble them into functional applications with relative ease. Similar tools should be developed to design cross-device applications for multi-surface environments, as well as for managing the arrival and departure of devices in the environment.

Conclusion

This position paper has identified three categories of multi-surface environments with different levels of capabilities, and outlined the interaction and technological challenges of multi-surface interaction. We have briefly described our work on Webstrates, and highlighted three challenges for the creation of multi-surface environments.

Acknowledgements

This work was financed in part by the Digitéo project MultiVis and the French ÉQUIPEX DigiScope.

References

[1] M. Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-wimp user interfaces. In *Proc. CHI '00*, 446–453. ACM, 2000.

[2] M. Beaudouin-Lafon, S. Huot, M. Nancel, W. Mackay, E. Pietriga, R. Primet, J. Wagner, O. Chapuis, C. Pillias, J. R. Eagan, T. Gjerlufsen, and C. Klokmose. Multisurface interaction in the WILD room. *IEEE Computer*, 45(4):48 –56, April 2012.

[3] P. Hamilton and D. J. Wigdor. Conductor: Enabling and understanding cross-device interaction. In *Proc. CHI '14*, 2773–2782. ACM, 2014.

[4] C. N. Klokmose, J. R. Eagan, S. Baader, W. Mackay, and M. Beaudouin-Lafon. Webstrates: Shareable dynamic media. In *Proc. UIST '15*, 280–290. ACM, 2015.

[5] T. H. Nelson. The heart of connection: Hypermedia unified by transclusion. *CACM*, 38(8):31–33, 1995.

Advances and Challenges in Ad-hoc Mobile Spatial Tracking for Seamless Interaction across Commodity Devices

Haojian Jin
Yahoo Labs
Sunnyvale, CA
haojian@yahoo-inc.com

Christian Holz
Microsoft Research
Redmond, WA
cholz@microsoft.com

Abstract

In this workshop paper, we assess the progress and lessons learned in developing *ad-hoc* cross-device tracking for mobile *spatial* interaction and point out the challenges existing systems still face. We identify the next steps that are necessary to bring truly fluent cross-device interaction to commodity devices. We finish with a discussion of the emerging opportunities in today's and future mobile devices as well as infrastructure systems that will facilitate accurate ad-hoc tracking on commodity devices.

Author Keywords

Ad-hoc mobile tracking, commodity devices, BLE.

ACM Classification Keywords

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

Introduction

To enable users to interact seamlessly across devices, involved devices need to be aware of surrounding devices. Tracking surrounding devices thereby needs to be ad-hoc, such that users can continue an activity on one mobile device *seamlessly* on another. A big challenge of such tracking is that users interact with devices in their *natural* space, yet current devices lack a notion of this ‘natural’ physical space. To support seamless interaction, devices thus need to understand this 3D space, which is particularly important when multiple users and devices are involved.

Unfortunately, current commodity mobile devices detect only the *presence* of surrounding devices to enable cross-device interaction (e.g., iOS 8 Continuity, Android Wear notifications), typically by analyzing the BLE signal strength to infer the proximity of other devices. This works particularly well if devices are held close.

However, when interacting across several present devices, apps often have to resort to showing a list of devices from which a user must pick. Ideally, though, users would interact naturally—much like they interact with physical objects in their environment.

Recent research projects have made progress in bringing ad-hoc tracking to mobile devices. They often integrate one or more signal types to enable UI apps to detect the *3D location* of surrounding devices. In this workshop paper, we first review the progress in mobile tracking and assess assumptions and shortcomings. We then outline the challenges and directions for future mobile commodity tracking systems.

Advances in ad-hoc tracking

Since most interaction across devices takes place on *mobile* devices, tracking systems need to work ad-hoc. This insight has brought researchers’ attention to the built-in sensors in today’s commodity devices.

Camera-based tracking on mobile devices
Much related work has used the video feed from the camera for ad-hoc tracking. For example, TouchProjector observes the screen contents of other stationary devices’ screens to infer their relative positions [1], but requires a constant visual connection to other screens. Orienteer requires both mobile devices to observe a shared view for registration, such as users’ shoes [4].

While the camera is a strong sensor for vision-based tracking, existing vision technologies are mostly developed for stationary rather than mobile cameras. The mobile nature and suboptimal camera position for mobile vision requires different solutions. Moreover, finger occlusion and table placement will further limit the robustness of ad-hoc camera tracking.

Inertial Motion Units (IMU) for mobile tracking
IMU sensors are the most responsive built-in sensors, which make them suitable for gesture recognition [9] and rough layout inference [6]. However, most mobile devices are not equipped with high-quality IMUs, causing sensor drift and rendering dead reckoning unusable for ad-hoc 3D tracking. To avoid drift, Tracko [9] integrates *temporary* dead reckoning and local coordinate transformation to prevent errors from sensor drift.

Audio processing for ad-hoc mobile tracking
Recent commodity mobile processors can process audio in real-time. BeepBeep [14] is one of the first projects that uses this to determine round-trip *distances* between devices. Tracko [9] establishes 3D tracking based on multiple audio units, producing high accuracy 3D tracking across devices, but is subject to noisy environments and restricted to limited interaction ranges.

Mobile radio to enable device tracking
Radio sensors (e.g., Wifi, GSM, BLE) are common on on today’s mobile devices. Many systems use the signal strength as an indicator to estimate the distance to remote devices. But since these radio sensors are not

designed for ranging purpose, predictions are inaccurate up to several meters (e.g., Tracko [8]). However, using better radio sensors could substantially increase the accuracy, such as QSRCT radio nodes [12].

Towards truly ad-hoc tracking in the wild

While researchers have examined ad-hoc tracking for a long time, it is the new technologies and advances in *commodity* hardware that bring about opportunities that let us think about mobile tracking in new ways—ways that were not possible previously.

Higher-quality built-in sensors

Future devices will pack higher-quality sensors, such as multiple speakers for communication. The popularity of mobile music has brought stereo speakers to many recent phones; the latest iPad Pro even has four speakers. Ad-hoc mobile tracking will benefit from these developments, such as enabling watches or other small devices with only a microphone to be tracked in 3D.

New sensors and low-cost hardware accessories

One potential way to improve commodity ad-hoc sensing is by integrating additional novel sensors into commodity devices—sensors that are expensive now, but will become low-cost through mass production. For example, Google's Tango understands its environment by using special-purpose depth cameras in a tablet.

Cameras on current mobile devices are under-utilized for tracking due to their position and limited field-of-view, which limits the tracking area. One step forward is SurroundSee [16], which is a mobile omni-directional camera that enables peripheral vision around the device to augment daily mobile tasks. We expect that more low-cost hardware, such as camera filters and phone cases [2] will improve tracking significantly.

Internet of Things and Wearables

A plethora of small devices is currently emerging, each dedicated to accomplishing a small task. While such Internet-of-Things devices typically connect using Bluetooth low energy, they often require knowledge of *where* they have been deployed. Wearables face a similar challenge: activity trackers would substantially benefit from an *spatial* awareness of where the user chooses to wear or carry them. For example, WristQue [13] adjusts the local heating and cooling system depending on its (static) location inside a building. We think that emerging systems for ad-hoc tracking, such as Tracko [9], will bring rich capabilities to smart devices that adapt to changed locations in smart environments..

Ubiquitous infrastructure

The recent development of indoor positioning systems makes tracking systems in the infrastructure ubiquitous, which also benefits ad-hoc tracking on commodity devices. WiFi-SLAM [5] determines the physical location of a mobile device based on wireless signal strengths from access points in the environment. Chung et al. [3] and Epsilon [11] explore the space of magnetic filed positioning and light-based positioning. Tracking infrastructure also helps further advancing the relative positioning on current systems to absolute positioning.

Convenient cross-device user authentication

Identification tokens, such as smartphones or wearables increasingly aid users in authentication with local and remote systems. Ad-hoc spatial device tracking will allow current implementations to increase their security by seamlessly ensuring that such identification tokens are close-by [7] and have the potential to fundamentally change scenarios in which multiple users interact with another device simultaneously, tracking and correctly associating all input with a particular user [8].

Development tools
The development in ad-hoc cross-devices is more complex than single device interactions as the developers need to debug on multiple devices at the same time. The efforts to develop ad-hoc cross-device interaction increase exponentially as the number of involved devices increases. We are looking forward to more software development tools like HuddleLamp [15].

Error-prone correction user interface
Tracking accuracy may be not perfect all the time. Thus, the ideal user interface should be able to handle errors implicitly. For example, Corona [10] uses implicit behavior to correct predictions. Depending on the accuracy of the tracking systems, there should be adaptive interfaces for different contexts. Balancing the tracking accuracy and the interactions it can enable can be a promising future research direction.

Conclusion
The recent technology innovations on hardware and software make ad-hoc tracking possible on mobile devices. Developments in this area are still early stage yet promising and indicate that they will impact users' future interactions profoundly. We analyzed the challenges of existing approaches and offered potential paths that we think will allow current systems to make big leaps forward. We believe that in 10 years with next-generation sensors and processors, ad-hoc tracking will be fully mobile and part of commodity devices and operating systems. Future systems will seamlessly integrate 3D ad-hoc tracking much like GPS today.

References

1. Boring, S. et al. Touch projector: mobile interaction through video. Proc. CHI '10, 2287–2296.
2. Butler, A. et al. SideSight: multi-"touch" interaction around small devices. *Proc. UIST '08*, 201–204.
3. Chung, J. et al. Indoor location sensing using geomagnetism. *Proc. MobiSys '11*.
4. Dearman, D. et al. Determining the orientation of proximate mobile devices using their back facing camera. *CHI '12*.
5. Ferris, B. et al. WiFi-SLAM Using Gaussian Process Latent Variable Models. *Proc. IJCAI '07*.
6. Goel, M. et al. SurfaceLink: using inertial and acoustic sensing to enable multi-device interaction on a surface. *CHI '14*.
7. Holz, C., Bentley, F. On-Demand Biometrics: Fast Cross-Device Authentication. *Proc. CHI '16*.
8. Holz, C., Knaust, M. Biometric Touch Sensing: Seamlessly augmenting each touch with continuous authentication. *UIST '15*.
9. Jin, H. et al. Tracko: Ad-hoc Mobile 3D Tracking Using Bluetooth Low Energy and Inaudible Signals for Cross-Device Interaction. *UIST '15*.
10. Jin, H. et al. Corona: Positioning Adjacent Device with Asymmetric Bluetooth Low Energy RSSI Distributions. *UIST '15*.
11. Li, L. et al. Epsilon: A visible light based positioning system. Proc. NSDI 14, 331-343.
12. Marquardt, N. et al. Crossdevice interaction via micromobility and f-formations. Proc. UIST '12.
13. Mayton, B.D. et al. WristQue: A personal sensor wristband. Proc. BSN '13, 1–6.
14. Peng, C. et al. BeepBeep: a high accuracy acoustic ranging system using cots mobile devices. *SenSys '07*.
15. Rädle et al. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. *ITS '14*.
16. Yang, X.D. et al. Surround-see: enabling peripheral vision on smartphones during active use. *UIST '13*.

The Challenges of Using an Existing Cross-Device Interaction Prototype for Supporting Actual Curation Practices

Frederik Brudy[1]
f.brudy@cs.ucl.ac.uk

Nicolai Marquardt[1]
n.marquardt@ucl.ac.uk

Yvonne Rogers[1]
y.rogers@ucl.ac.uk
[1]University College London
UCL Interaction Centre
Gower Street London, UK

Abigail Sellen[2]
asellen@microsoft.com

Kenton O'Hara[2]
keohar@microsoft.com
[2]Microsoft Research
Station Road
Cambridge, UK

Abstract
Volunteer-driven organisations curating historic documents, such as societies and charities, often work within a bring-your-own-device (BYOD) practice and their meetings are in varying situations. A recurring challenge is finding lightweight ways to enable them to share and collectively work with documents using their own devices while in situ. We are working on building novel interaction techniques and applications (prototyped with a custom developer toolkit) for supporting the curation of digital collections – for example, historic documents. We discuss the pros and cons of using an existing prototype system for this purpose and points to consider when taking a prototype from the lab into the wild.

Author Keywords
Cross-device interactions; in-the-wild; bring-your-own-device; using existing frameworks.

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

Introduction
Within volunteer driven organisation, such as historic societies and charities, bring-your-own-device (BYOD) is a common practice. Working collaboratively in a group can help to reveal more connections between various resources during co-located meetings, and allows to bring together people with expertise from different backgrounds. However, collecting, analysing,

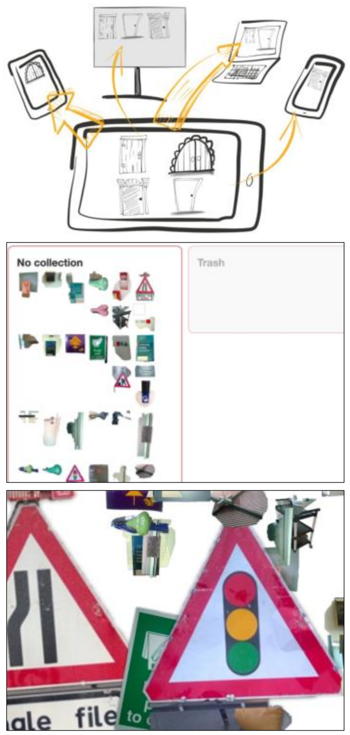


Figure 2: The general flow of items in CollectionsExplorer (top): photos can be moved across devices: all tablets share a digital canvas. Middle and bottom: All photos (in this case mainly photos of street signs) are placed on a virtual canvas that can be explored by moving the iPads on the table. Items can be moved, rotated, scaled and flicked between tablets.

creatively reworking, or sharing digital content as a group across a diverse ecology of devices is difficult: most devices work in isolation, not well supporting any collaborative collection, organization, or sharing activities. In recent years, researchers have produced several different frameworks for spatial tracking of people and devices, as well as supporting cross-device interactions (e.g. [4,5,7]). These frameworks were proposed for rapid development of (research) prototypes and were often mainly used to demonstrate a proof-of-concept in the lab. However, when taking such a tool out of the lab, in order to build a system to deploy in the wild, there are several challenges which need to be addressed. In particular rigidity is one of them: even prototype systems need to be more robust in-the-wild than when tested in controlled situations. Examples are changing environments (such as changing lighting conditions, or cluttered areas), people using applications in (slightly) different ways than what they were intended for, or users using a system for entirely different activities.

In our research we are interested in how co-located curation activities can be supported through cross-device interaction techniques. We are building a specialised developer toolkit, supporting in particular novel cross-device interactions within BYOD practices, collaborative content curation, and blending digital and physical artefacts.

CollectionsExplorer for collaborative curation activities

In order to support these small group collaborations we are developing *CollectionsExplorer*, a set of hardware and software tools that enable content curation [8] tasks to be facilitated when working with multiple tablet devices. However, rather than start from scratch we chose to build *CollectionsExplorer* using an existing platform that had been used for demo purposes beforehand. *CollectionsExplorer* was built on top of *HuddleLamp* [7], which is a technology developed to spatially track devices, providing a way of combining them into a larger surface. HuddleLamp uses a hybrid approach of a depth-sensing and an RGB-camera to identify and track tablets and phones on a table. We deployed *CollectionsExplorer* during informal pilot studies and as part of a workshop to various user groups. We observed how participants approached the system, adapting it to their needs – and adapting their own behaviour in order to avoid the pitfalls of the system.

CollectionsExplorer was built to enable photographic collections to be shared across multiple devices, allowing users to explore individual pictures as well as creating new collections out of existing ones. A user can browse collections of photos, zoom in, rotate, and move and flick individual pictures between multiple devices (Figure 2 top). Figure 2 shows how a user explores multiple picture collections on an iPad. Each collection is organized in stacks of images (Figure 2 middle), which can then be spatially arranged on a digital canvas shared by all connected devices (Figure 2 bottom). In the future, *CollectionsExplorer* will be extended to support other key curation activities (e.g., duplicating current states to take home, or different ways of presenting results) and additional interaction techniques supporting these tasks.

For the spatial tracking of the devices we are using an existing system [7], which requires the tablets to lay flat on a table. When the camera's field of view is clear and not the lighting conditions are controlled, the tracking is stable and precise. However, as soon as people try to use the system outside of a controlled lab situation, using it in an everyday task or as part of their daily routine, new problems arise, e.g. tracking gets lost because of occlusion, lighting conditions change, or people adapt tools in ways that works best for them, not how the developer might have anticipated. Thorough testing is needed, in order to get the technical issues solved and to get the interaction techniques clear enough so that they do not break outside controlled situations.

For example in our case we have observed that when presented with *CollectionsExplorer*, people's first reaction was to pick up one of the tablets to have a closer look. However, since the camera-based tracking only works when the tablets are placed on a table, the system fails. Another issue arose, when people pointed to a specific photo, or reached for it to increase its size or to rotate it. With their arms reaching into the field of view of the camera, overhead tracking does not work properly. Both of this (not being able to pick up tablets and frequently lost tracking through occlusion) distracted participants from their main task. Instead of focusing on their primary task their main focus became how to avoid the system to fail, the technology itself got in the users' way. As a result, participants of the study refrained from further touching the tablets and relied on pointing to the tablets from afar, keeping a safe distance so that their arms did not occlude the camera. Some users reported to start thinking about the technical setup more than about their primary curation task.

Open questions for the workshop

When taking a new tool from the lab into the wild there are several challenges which need to be addressed. In the following they are split up into technical and social challenges.

Technical challenges

Setup and prerequisites. How can we enable cross-device interactions without the need for special devices or complicated setup mechanisms? In particular, for BYOD and walk-up-and-use situations the setup needs to be easy, quick, and allow people with varying technical knowledge to integrate their own and other devices. How should devices be best connected to a system? How should availability and execution of cross-device interactions be best communicated to a user?

Ubiquity and precision of spatial tracking. When employing spatial tracking, how can this tracking and the required devices be integrated in the surrounding? A precise spatial tracking often comes at a cost, e.g. the need for specialised hardware, markers, long setup procedures, etc. If less precise, these requirements could be reduced. However, this varies with the intended use case. We would like to see a discussion about the cost of precise spatial tracking vs. its benefit.

Rigidity for in-the-wild deployments. How does a research prototype need to be changed in order to be taken from the lab into the wild? For in-the-wild purposes they need to be robust, being able to be used in different scenarios than what they were intended for.

Mixed data and environment. To account for the rich nature of documents and the mixed settings of various tasks, a system should account for both, digital and non-digital content and allow to work in environments with cluttered work areas and noisy data. Deployments in the wild might happen in cluttered areas, much unlike a controlled lab study, making possible issues unpredictable.

Social challenges

Communicating availability and ability. How are availability and interactions communicated to people in a walk-up-and-use situation? For example display blindness [6] and the honeypot effect [2] have been observed with public displays and could serve as a starting point for how those could be considered when designing cross-device interaction techniques.

Privacy and personal space. Going from the lab into the wild requires not only technical adjustments, but also careful considerations of other factors, e.g. when employing public displays, issues such as privacy [3], or personal spaces [1] and territories [9] need to be taken into account.

Control and feedback. Different social expectations and cultural backgrounds are further factors to be considered. For example, how much control should a person have over the tracking within a system and how much should the control be hidden and stay in the background?

Acknowledgments

This work is supported by Microsoft Research through its PhD Scholarship Programme.

References

[1] Altman, I. The environment and social behavior: privacy, personal space, territory, and crowding. 1975.

[2] Brignull, H. and Rogers, Y. Enticing people to interact with large public displays in public spaces. Proceedings of INTERACT, (2003).

[3] Brudy, F., Ledo, D., Greenberg, S., and Butz, A. Is Anyone Looking? Mitigating Shoulder Surfing on Public Displays through Awareness and Protection. ACM Press (2014), 1–6.

[4] Houben, S. and Marquardt, N. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. ACM Press (2015), 1247–1256.

[5] Marquardt, N., Diaz-Marino, R., Boring, S., and Greenberg, S. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. Proceedings of the 24th annual ACM symposium on User interface software and technology, (2011), 315–326.

[6] Müller, J., Wilmsmann, D., Exeler, J., et al. Display blindness: The effect of expectations on attention towards digital signage. In Pervasive Computing. Springer, 2009, 1–8.

[7] Rädle, R., Jetter, H.-C., Marquardt, N., Reiterer, H., and Rogers, Y. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces (2014), 45–54.

[8] Rotman, D., Procita, K., Hansen, D., Sims Parr, C., and Preece, J. Supporting content curation communities: The case of the Encyclopedia of Life. Journal of the American Society for Information Science and Technology 63, 6 (2012), 1092–1107.

[9] Scott, S.D., Carpendale, M.S.T., and Inkpen, K.M. Territoriality in collaborative tabletop workspaces. Proceedings of the 2004 ACM conference on Computer supported cooperative work, ACM (2004), 294–303.

The benefits of ‘In The Wild’ studies for successful introduction of ‘Bring Your Own Device’ policies in the industry

Jens Ziegler
Sebastian Heinze
Leon Urbas

Technische Universität Dresden
01062 Dresden, Germany

jens.ziegler@tu-dresden.de
sebastian.heinze1@tu-dresden.de
leon.urbas@tu-dresden.de

Abstract

Bring Your Own Device (BYOD) policies are becoming more and more popular in large corporations - in business, but also in industry. The possible benefits are reduced investment costs and improved productivity, flexibility and satisfaction of the users. However, BYOD policies raise new challenges for industry corporations in terms of device integration and evaluation strategies for the resulting IT ecosystem. In this contribution, we will briefly introduce those challenges. We will give generic recommendations on how to create (industrial) BYOD-enabled applications and systems. Finally, we will present some of our research results and our research agenda for BYOD policies in Industry.

Author Keywords

Cross-device interaction; Bring Your Own Device; Evaluation in the wild; Multi-device interactions; Distributed user interfaces, Wearable, Mobile

ACM Classification Keywords

H.5.2. Information interfaces and presentation: User Interfaces: Graphical user interfaces (GUI); Evaluation/methodology

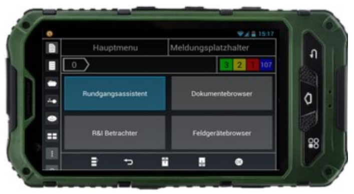


Figure 1: Mobile application to support industrial maintenance tasks on a ruggedized COTS smartphone.

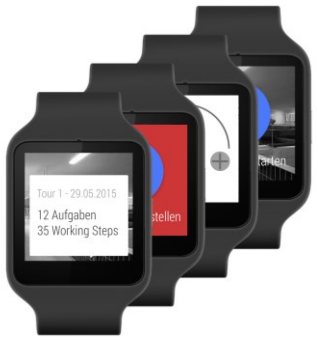


Figure 2: Ubiquitous task support using multiple event-driven Android Wear apps on a COTS smartwatch.

Bring Your Own Device (BYOD) - What does it mean (for industry)?

Bring Your Own Device (BYOD) is an increasingly popular business policy which enables employees to use technology which was first deployed in the consumer market and which is owned by the employees themselves [1]. This allows the company to reduce investment and maintenance expenses for the devices and leads to increased employee satisfaction. One major concern, however is the increased number of potential security risks. Insecure connections, lost or stolen devices, malware and personal privacy issues of the device owner are major challenges of the BYOD approach [4]. These issues are subject to research and development driven by leading software companies, however. A second major concern is the lack of appropriate means to verify BYOD IT-ecosystems in industry-grade usability evaluations. The fragmentation of the system, the very short device and platform lifecycles and the quick evolution of interaction paradigms require an explicit design for evolution. Conventional lab and field usability testing faces practical limits when it comes to continuous evaluation on multiple user-owned devices with multiple operating systems that follow different interaction paradigms [2]. Alternative approaches like remote usability testing still perform significantly below lab tests [1]. Yet, many questions of evaluating the usability of BYOD IT-ecosystems in the industry remain unsolved. In the process industries, where our research is settled, even further requirements may arise [6].

How to create (industrial) BYOD applications and systems?

According to our research experiences over the past years, we follow three basic principles in designing successful mobile and BYOD IT-ecosystems for industry corporations. Firstly, we largely follow the design philosophy and guidelines that come with the platform of the users-owned devices. This may require some restrictions of the accepted operating systems, vendors, versions or models of hardware and software (*Choose Your Own Device* policy). Following the update cycles determined by the platform vendors is crucial in order to not lag behind the state of the art. This is difficult because it sometimes means to redesign an application although the functionality is still sufficient for the task. However, the more the design of an application becomes outdated the more the user's perception of non-functional quality attributes will deteriorate.

Secondly, we largely rely on the device and deployment ecosystem provided by the platform vendor. Most vendors offer specific solutions for industry users. One may adapt, enhance or even replicate the ecosystem if necessary, but this should be done transparently to the users. Complex corporate tasks often call for more sophisticated means for deployment and administration of the IT-ecosystem than non-professional consumers.

Finally, we try to continuously evaluate BYOD-enabled applications and systems in real settings with real users imposing as few artificial limitations as possible. Ecological validity is crucial for usability / user experience evaluations when it comes to BYOD scenarios. *In The Wild* studies might meet the requirements of usability testing of BYOD IT-ecosystems. Study designs with mainly observational character or very limited targeted variation of independent variables or control of confounding factors, however, tend to be insufficient for industry-grade usability evaluations. Same holds for low-effort remote usability testing methods proposed in the literature.

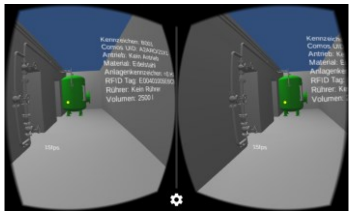


Figure 3: Unity-driven stereoscopic 3D VR application of a production site. The prototype supports interactive highlighting of assets and provides relevant asset data on-the-fly.

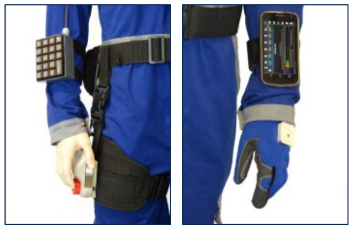


Figure 4: Wearable distributed user interfaces enable users to operate their mobile systems in settings where touch operation is unsuitable. They are also used to operate wearable systems such as VR systems.

Research Results and a Research Agenda for BYOD in Industry

Following the first principle, we largely rely on the Google Android ecosystem. We developed several mobile applications for common tasks in the process industry [4, 7, 9] (Figure 1) and extended them with sophisticated smartwatch support [10] (Figure 2). We also incorporated the Google Cardboard VR environment to virtually explore production sites (Figure 5). Largely relying on the design guidelines of the mentioned platforms we created cross-device applications that are easy to use for employees which are familiar with the Android ecosystem and that are easy to maintain for the app developers.

A main objective of a corporate application is to implement a certain, well defined business process or working procedure. Our *App Orchestration Framework* [7] provides a powerful engine to arrange sets of multiple apps (so-called *app ensembles*) based on BPMN models of the business processes and workflows (so-called *app orchestration*). Following the second principle, this framework is based on *Google Android*, thus the apps may be deployed via the off-the-shelf available *Google Play for Work* infrastructure. In addition, a *Jenkins automation server* has been extended to automatically create app ensembles for the user-owned device and to organize the deployment, either directly from the *Jenkins* server or from the *Google Play for Work* infrastructure. Currently, this infrastructure supports *Google Android* and *Android Wear* applications in order to create convenient cross-device interaction. In the near future, we plan to include *Google Cardboard* applications as well. This deployment infrastructure also provides comprehensive support for different input and output devices (so-called *wearable distributed user interfaces*) [8]. Using this infrastructure, multiple app ensembles can be created for the same task, where each version is optimized for a specific interaction technique (e.g. single-hand game controllers, keypads, gesture or speech input). Thus, users can choose the app ensemble that is best suited for their current task (Figure 4). This flexible multi-device orchestration allows for cross-device interaction under the highly adverse and variable working contexts of use where touch interaction is unsuitable.

In order to bring research one step further, we are developing evaluation strategies that allow for the subtle, yet controlled variation of independent variables and a sufficient characterization and treatment of the participants. The downside of ecologically valid settings is the evaluator's limited control over confounding environmental and situational factors. Following the third principle, one focus of our research is to develop control strategies and measures that reduce or govern confounding effects both in *In The Field* and *In The Wild* experiments.

In The Wild studies are limited in the possibilities to equip the evaluation environment with measuring instruments and in the ability to actively involve users in the evaluation of the system (e.g. by means of a questionnaire). For this reason, we aim at developing a device-centric measurement methodology including cloud-based logging (user input, system reaction), camera and audio data (e.g. for gaze and emotion analysis), bio-signals (coming from smart wearables), context information such as illumination, noise, temperature or weather data from the device nearby-devices in the IT ecosystem (e.g. smart home sensors) or other sources of relevant information.

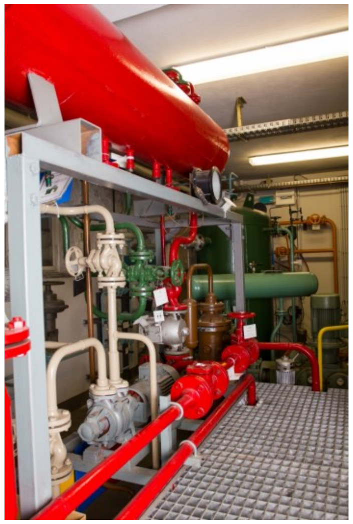


Figure 5: Experimental facility in an industrial scale used to develop evaluation strategies and to do usability testing of mobile and wearable applications for the process industry.

Conclusion – Where are we now?

It is easier than ever to create complex cross-device interactions with commercial-off-the-shelf (COTS) products. There are device and service ecosystems spanning across a wide range of interaction devices, and which are widely used by millions of users every day. BYOD policies allow corporations to gain advantage from these ecosystems. We have developed appropriate deployment infrastructures over the past years and will continue this work by integrating novel COTS platforms such as AR/VR-systems, wearable systems and very large displays (e.g. power walls).

BYOD-enabled IT-ecosystems should continuously be evaluated *In The Wild*. In order to improve the performance of such studies, subtle means for targeted variation and elementary control of the most relevant variables as well as a user-independent, device-centric measurement methodology need to be developed further. Such means can be partly adopted from the area of *In the Field* research, but novel strategies and techniques need to be developed in order to take the specific characteristics of *In The Wild* studies into account. We are using a realistic industrial environment to improve and test our device-centric measurement methodology and evaluation strategy (Figure 5). This approach proved to be a good compromise between the rigidity of lab usability testing and the ecological validity of field testing, especially in domains such as the process industry, where real environments are adverse and dangerous. For large-scale summative usability testing, however, we will take our tool set and methodology and to go *Into The Wild*.

References

1. Bruun, A. et al. (2009). Let your users do the testing: a comparison of three remote asynchronous usability testing methods. In: *CHI 2009*: 1619-1628. ACM.
2. Connelly, K. et al. (2008). Evaluating pervasive and ubiquitous systems. In: *Pervasive Computing 7(3)*: 85-88. IEEE.
3. Forrester Research (2012). Key Strategies To Capture And Measure The Value Of Consumerization Of IT. Forrester Consulting.
4. Münch, T. et al. (2014). Collaboration and Interoperability within a Virtual Enterprise Applied in a Mobile Maintenance Scenario. In: *Charalabidis, Y. et al. (Eds.). Revolutionizing Enterprise Interoperability through Scientific Foundations*. IGI.
5. Shumate, T.; Ketel, M. (2014). Bring Your Own Device: Benefits, Risks and Control Techniques. In: *Proc. IEEE SOUTHEASTCON 2014*: 1-6. IEEE.
6. Urbas, L. (2012). Process Control Systems Engineering. Deutscher Industrieverlag GmbH.
7. Ziegler, J. et al. (2012). Beyond app-chaining: Mobile app orchestration for efficient model driven software generation. In: *Proc. IEEE ETFA 2012*: 1-8. IEEE.
8. Ziegler, J.; Urbas, L. (2013). Enhancing Mobile Interactions with Distributed Wearable User Interfaces. In: *Proc. IADIS IHCI 2013*: 288-292. IADIS Press.
9. Ziegler, J. et al. (2014). App-based System Diagnosis using Mobile Information Systems. In: *Proc. IEEE ETFA 2014*: 1-8. IEEE.
10. Ziegler, J.; Heinze, S.; Urbas, L. (2015). The potential of smartwatches to support mobile industrial maintenance tasks. In: *Proc. IEEE ETFA 2015*: 1-7. IEEE. DOI: 10.1109/ETFA.2015.7301479

Towards Context-Aware Cross-Device User Interfaces in the Wild

Fabio Paternò, Giuseppe Ghiani, Marco Manca
CNR-ISTI, HIIS Laboratory
Via Moruzzi 1, 56124 Pisa, Italy
fabio.paterno@isti.cnr.it

Abstract
In recent years a number of frameworks for easing design and development of cross-device user interfaces have been put forward, mainly in research contexts. In general, they provide support for connection management, data synchronization, and user interface distribution. However, the applications that can be obtained can be accessed from a wide variety of contexts of use that vary in terms of available devices and connectivity, surrounding environment, user preferences and abilities, and social relationships. Thus, one of the main limitations in the adoption of such applications in the wild is the difficulty to customize them for different needs in such diverse contexts. This position paper indicates and discusses the issues that should be addressed for this purpose and intrdouces possible approaches to solve them.

Author Keywords
Cross-device user interfaces, Context of use, End-user development.

ACM Classification Keywords
H.5.2 [Information interfaces and presentation (e.g., HCI)]: User Interfaces - Input devices and strategies.

Introduction

The increasing availability of various types of devices in our daily life is often a missed opportunity since current applications are limited in supporting seamless task performance across them. Users often perceive device fragmentation around them rather than an ecosystem of devices that supports their activities. In order to address such issues a number of frameworks, platforms, and authoring environments have been proposed, mainly in research environment. The goal is to facilitate design and development of multi-device user interfaces. We can distinguish various types of multi-device user interfaces depending on the features that they support: *migratory user interfaces* are able to

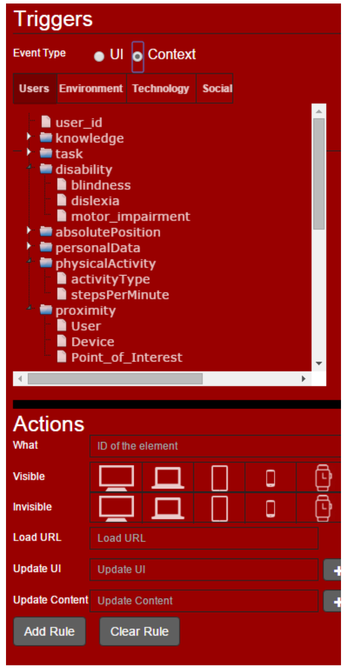

Event Type UI [✓ Context]

Figure 1 Tool for editing trigger action rules

dynamically migrate from one device to another in order to follow users' movements while preserving their state; *distributed user interfaces* allow users to interact with an application through multiple devices at the same time; *cross-device user interfaces* are distributed user interfaces, with the additional capability to synchronise their state, so that the interactions through some element in one device update the state of the corresponding elements (if any) in another device. Such categories are not mutually exclusive, so for example it is possible to have user interfaces that are both migratory and cross-device.

Cross-device Frameworks and Authoring Environments

In recent years some frameworks that provide useful support for developing cross-device user interfaces have been proposed. The proximity toolkit [5] simplifies the exploration of interaction techniques by supplying fine-grained proxemics information between people, portable devices, large interactive surfaces, and other non-digital objects in a room-sized environment. We have designed a framework supporting user interface distribution in multi-device and multi-user environments with dynamically migrating engines has been proposed [2]. It does not require a fixed server to manage the distribution. The elements of the UI can be distributed by specifying specific device(s), group(s) of devices, specific user(s), and groups of users according to roles. Panelrama [7] is a solution able to categorize device characteristics and dynamically change UI allocation to best-fit devices. For this purpose, this framework lets developers to specify the suitability of panels to different types of devices. The increasing use of wearables in the context of cross-device user interfaces has been addressed by Weave [1], a framework for developers to create cross-device wearable interaction by scripting. It provides a set of JavaScript- based APIs to easily distribute UI output and combine sensing events and user input across mobile and wearable devices. Other cross-device frameworks involving smartwatches have been proposed (e.g. [4]). In addition to frameworks, also some authoring environment to ease the development of cross-device user interfaces has been proposed. An example is XDStudio [6], which supports two complementary authoring modes: simulated and on-device. In the former mode, authoring is carried out on a single device in which the user interfaces distributed are simulated. In the latter mode, design and development actually takes place on the target devices themselves. However, this type of authoring environment does not provide support for specifying context-dependent behavior. This aspect has been addressed by our context-aware authoring environment [3], which supports development of user interfaces able to adapt to the various types of contextual events (that can be related to users, devices, environments, and social relationships), with the possibility of distributing the user interface elements across multiple devices. The context-dependent behavior is modelled through trigger / action rules (an example tool for editing them is in Fig.1), and can even be applied to extend the capabilities of Web applications that were not originally designed to be context-aware.

An Architecture for Context-aware Cross-device User Interfaces

In order to correctly execute the applications according to the adaptation rules specified it is necessary to have a specific architectural support at run-time. The main goals of such support are to manage and apply the user interface adaptation or distribution rules, and detect the events that trigger their performance. Such run-time support exploits the functionalities of three components: the context manager, the adaptation engine, and the distribution manager. The context manager is composed of a context server and a set of external modules delegated to monitor relevant parameters of the context of use (e.g. environmental noise, device coordinates, user physical activity).

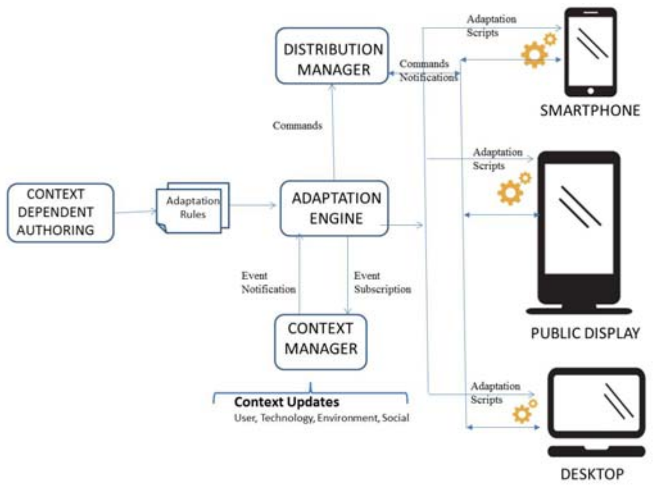


Figure 2 Architecture for Context-aware Cross-device User Interfaces.

The purpose of the context manager is to detect contextual events and inform the adaptation engine, which stores and manages the contextual rules, and requests changes in the cross-device user interface according to the triggered rules. The distribution manager handles user interfaces distributed across multiple devices in order to allow dynamic migration of components and keep their state synchronized. Figure 2 shows how such components interact with each other. The adaptation engine subscribes to the context model manager in order to be informed of the occurrence of the events relevant for the rules associated with the active applications. When one or more of such events occur, the adaptation engine sends the actions to the applications in order to perform the corresponding changes. Such update commands are interpreted by the scripts included in the application by the authoring environment. They can modify properties of user interface elements or content, activate functions or navigation, and change the distribution of some user interface parts across devices. In the latter case the adaptation engine can directly send the corresponding command to the distribution manager, which notifies the involved devices. Such distribution manager contains the current distribution profile, which indicates how the various parts of the user interface are currently distributed across the devices that have subscribed to the environment. A distribution command mainly determines whether a user interface element or the elements included in a container should be visible or not on one specific device or a group of devices that have the same role or on all devices of a given platform.

Issues for Deployment in the Wild

The approach to context-aware cross-device user interfaces is general and can be deployed for a wide variety of applications (for example smart retail, museums, smart cities, e-learning, ...). For this purpose various aspects should be considered.

Interoperability.

We need the possibility to operate on various types of devices (smartwatches, smartphones, tablets, desktops, public displays, ..) from various vendors. Only Web applications can be accessed through almost all of them with limited effort. However, the run-time supporting the cross-device user interfaces should be able to work even when network connections to remote external Web servers is not possible (a possible solution is described in [2]). This means that the underlying architecture should be able to create peer-to-peer organization amongst the involved devices.

End-user development

In the end only the users know the best way to configure their cross-device user interfaces in their specific contexts of use, thus we need to provide them with authoring environments and customization tools that allow them to directly specify the contextual rules even if they do not know how the underlying technology works. For this purpose the use of subset of natural language to indicate the desired behaviour with familiar, domain-dependent terms can be effective.

Flexibility

The control on the cross device user interface by developers and users should be able to address various granularity levels when allocating or dynamically changing which user interface parts should be in each device. We can identify four possible granularity levels: some distribution changes can involve the entire user interface, others can only involve groups of elements, or be limited to single user interface elements (e.g. a list or a text input), or even parts of single elements (e.g. their prompt or feedback).

Modalities

Some approaches only consider graphical cross-device user interfaces but natural interaction can be achieved if the associated environments are also able to support other various modalities that users can exploit depending on the context (vocal, gestural, graphical, …) in an integrated manner.

Mixed-initiative Triggers

The changes in the configuration of the cross-device user interface can be made on explicit request through customization tools or triggered automatically by the context-dependent rules. In the latter case it is still important to make users aware when the changes occur, with also the possibility to reject them if they are not deemed useful at a given time.

References

1. Chi, P. and Li, Y. 2015. Weave: Scripting Cross-Device Wearable Interaction. CHI 2015, ACM
2. Frosini, L. and Paternò, F. 2014. User Interface Distribution in Multi-Device and Multi-User Environments with Dynamically Migrating Engines. Proceedings of EICS 2014, ACM, pp. 55-64.
3. Ghiani G. Manca M. Paternò F., Authoring Context-dependent Cross-device User Interfaces based on Trigger/Action Rules, In MUM2015, pp. 313-322.
4. Houben, S., and Marquardt, N. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. Proceedings of CHI 2015, ACM, pp. 1247-1256.
5. Marquardt, et al.. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. UIST 2011, pp. 315-326.
6. Nebeling, M., Mintsi, T., Husmann, M., Norrie, M. C. 2014. Interactive development of cross-device user interfaces. In CHI 2014, ACM, pp. 2793-2802.
7. Yang, J. and Wigdor, D. 2014. Panelrama: enabling easy specification of cross-device web applications. In Proceedings of CHI 2014, ACM, pp. 2783-2792.

Prototyping "In The Wild" Interaction Scenarios With RE/Tk

Aneesh P. Tarun
Synaesthetic Media Lab
Ryerson University
Toronto, Ontario, Canada
aneesh@ryerson.ca

Ali Mazalek
Synaesthetic Media Lab
Ryerson University
Toronto, Ontario, Canada
mazalek@ryerson.ca

Andrea Bellucci
Universidad Carlos III de Madrid
Avenida de la Universidad, 30
28911, Leganés, Madrid, Spain
abellucc@inf.uc3m.es

Abstract

Building interactive environments that blend digital information into the physical world is hindered by the complexity of setting up the technological medium. We developed the Responsive Ecologies Toolkit (RE/Tk) to provide researchers and developers with a toolkit that would cut down the low-level technical demands thus making it easier to prototype applications for heterogeneous networked devices. This position paper argues for a better conceptual model to support design of interaction experiences "in the wild". It also proposes extensions to the RE/Tk to support design iterations where the potential interaction devices and their capabilities are not known.

Author Keywords

Toolkit; Prototyping; Cross-Device Interfaces; Interactive Environments.

ACM Classification Keywords

H.5.2. Information Interfaces. User Interfaces – input devices and strategies, prototyping.

Introduction

A growing body of research in cross-device interfaces has focused on providing interaction techniques for sharing information across devices [1], as well as mapping gestures across devices [3]. Much of the effort

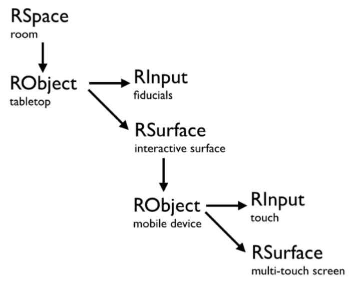


Figure 1: An example of the nested hierarchical structure.

put into such research involves bootstrapping low-level technical challenges such as programming for multiple platforms, communication across devices, and developing an ad-hoc framework for comprehending the complexity of cross-device interfaces. This has led researchers to revisit a more fundamental question in cross-device interface research. What tools and techniques are beneficial for simplifying the exploration of cross-device interaction techniques?

There exist numerous software toolkits for the rapid prototyping of spatially-aware interactions [5], tangible and physical computing [2] [4] and cross-device interactions [7][8][9]. However, there is limited support for the design of complex interactive environments involving heterogeneous off-the-shelf as well as custom devices. Weave [8], for instance, provides an authoring environment for cross-device interactions which supports off-the-shelf wearable and mobile devices. WatchCONNECT [8] explores sensor-based interactions focusing on smartwatches. Panelrama [7] aims at easing the development of distributed user interfaces, thus it does not consider smart objects without a display surface.

Our goal with RE/Tk is to provide support to quickly build responsive and interactive environments, which can include people personal devices, interactive surfaces and custom-made tangible objects (e.g., arduino-based devices). In addition, we wish to provide a fully customizable and extensible toolkit that supports multiple aspects of developing cross-device applications.

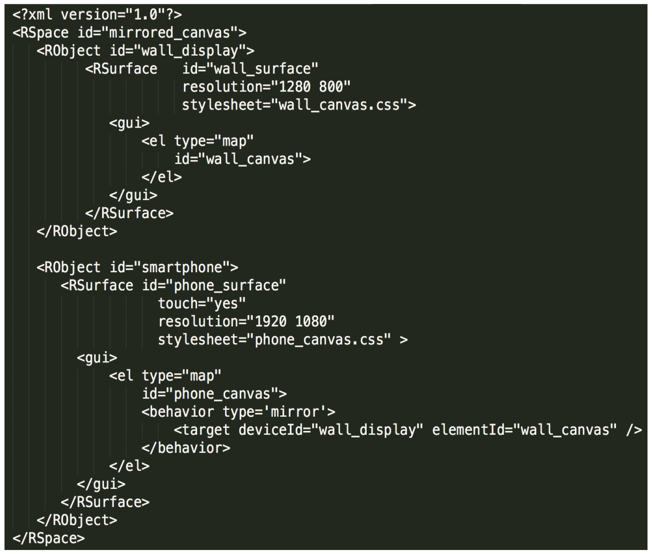
There are several reasons for this. Firstly, the landscape of interaction devices and their capabilities is ever-changing. The heterogeneity of the underlying runtime environments and the communication protocols of these devices makes it challenging to provide a toolkit that stands the test of time. In addition, scaling up or modifying existing toolkits to support cross-device interface development does not address the underlying fundamental issue: there is no conceptual framework that allows designers to visualize and discuss a complex interaction scenario at a higher level of abstraction.

We designed the Responsive Ecologies Toolkit (RE/Tk), building on the work of the ROSS Toolkit [6], for prototyping applications that span across different (off-the-shelf and custom) devices. The RE/Tk provides a conceptual framework for designing multi-device applications and toolkit for prototyping such applications.

However, in its current state, the toolkit assumes that the "players" in the interaction space i.e. the underlying devices and sensors are known before hand and conform to a fixed device hierarchy within the interaction space. Below, we discuss the current state of the RE/Tk and propose a conceptual framework and new features to the toolkit that will enable prototyping & development of "in the wild" interaction experiences such as bring-your-own-device (BYOD) scenarios.

RE/Tk

RE/Tk provides a conceptual framework that allows the developers to conceptualize and design interactive environments as hierarchical nested structures. Every object, screen, sensor in an interaction space is mapped within a hierarchical structure. This hierarchical tree (Figure 1) encapsulates relationships between various entities and determines how they interact: (a) direct interaction and communication occurs between a child node and its parent node, (b) interaction between sibling nodes is mediated by their parent node, (c) the tree structure is also used to determine the communication path between two (directly unrelated) nodes, and (d) the structure allows for easier computation of spatial interactions between different devices.



```xml
<?xml version="1.0"?>
<RSpace id="mirrored_canvas">
   <RObject id="wall_display">
         <RSurface   id="wall_surface"
                     resolution="1280 800"
                     stylesheet="wall_canvas.css">
            <gui>
                <el type="map"
                    id="wall_canvas">
                </el>
            </gui>
         </RSurface>
   </RObject>

   <RObject id="smartphone">
      <RSurface id="phone_surface"
                  touch="yes"
                  resolution="1920 1080"
                  stylesheet="phone_canvas.css" >
        <gui>
            <el type="map"
                id="phone_canvas">
                <behavior type='mirror'>
                    <target deviceId="wall_display" elementId="wall_canvas" />
                </behavior>
            </el>
        </gui>
      </RSurface>
   </RObject>
</RSpace>
```

Figure 2: An example XML descriptor file for a scenario where a wall display mirrors the interactions on a smartphone.

Designers and developers outline this structure in an XML descriptor file (see Figure 2). This document is fed to the toolkit which generates the application code and manages the communication between different sensors and devices.

All generated server and client components are in JavaScript. The server application runs on Node.js (a cross-platform runtime environment) while the client applications run on the devices' browsers. As an exception to the rule, we generate deployable code for Arduino and other microcontrollers that do not support a JavaScript runtime environment.

A Javascript API exposes the functionality of the toolkit and provides an alternative way of developing or iterating the generated interaction software for expert developers.

In addition, to support "moving targets", RE/Tk is designed to be extensible. Designers and developers can extend the XML structure as well as the JavaScript API to suit custom workflows.

This approach simplifies the development, deployment and management of applications onto a variety of hardware and software platforms. Since the JavaScript code is compiled at runtime, modifying a part of the application code does not require recompilation for the entire interactive environment. Code can also be locally modified and inspected for debugging purposes. These features encourage faster iteration of application designs.

In addition, the toolkit provides different levels of abstractions in terms of application behavior, GUI and widgets, sensor mapping, spatial mapping and communication.

Application Behavior Abstraction. XML-based application authoring is the first level of abstraction provided by the toolkit. All the features and functions of the API are abstracted and accessible as XML tags. Large and complex XML files can also be split into multiple XML files, each defining a different feature of an application, i.e. one XML file for describing the application functionality and another one for the UI.

GUI Abstraction and Widgets. An authored XML description includes not only the device structure but also the user interface design for all the devices. This provides a unified way of designing UIs for different types of screens. This also allows for easily porting applications to native code when needed. In addition to this, we have implemented a few generic widgets (e.g. Maps widget) that can easily be included in any application.

Sensor Mapping Abstraction. RE/Tk supports straightforward mapping of sensor output data of one device as an input to a function of another device or an actuator in just a few lines. The underlying implementation of listener events, data range mapping, network routing etc. are abstracted from the developers (they are still accessible to the developers through the generated application code for modifications). Custom data filters can also be easily attached to a sensor output.

Spatial Mapping Abstraction. The hierarchical nested structure of devices is leveraged to provide easy access to spatial information of each device. Developers can directly query position information of each device relative to the interaction space or another device. Converting spatial data from one coordinate system to another is also supported by the built-in functions.

Communication Abstraction. RE/Tk uses and builds upon TUIO to simplify cross-device communication. Developers can extend the protocol for custom scenarios. The hierarchical structure, used to automatically setup the underlying communication channels between devices, forms another level of communication abstraction.

In addition to these abstractions, the modular nature of the XML files (and the API in general) allows developers to easily extend the features of RE/Tk.

Supporting interactions *in the wild*

An interaction environment with a fixed network of devices poses sufficient challenges for a developer to envision and deploy interactive experiences. However, prior access to the network information, device capabilities, and access control provides a handle to the developer for managing the overall interaction experience. RE/Tk leverages this information to simplify authoring of interactive applications. "In the wild" scenarios involve additional challenges that need to be addressed while prototyping.

In this section we discuss some novel features for the RE/Tk to support interactive BYOD scenarios. One assumption we make about the BYOD scenarios is that the developers of such scenarios have access to at least one device within the interaction environment that will be assigned the role of a server/arbiter. An example of a typical BYOD scenario: an interactive surface computer in a public space that allows walk-in users to lay their smartphones on the surface and interact with the displayed information on their smartphones as well as the larger surface.

Challenge 1: Since the interaction devices and their capabilities are not know during the design process, the toolkit has to provide a new framework and additional tools to design adaptive experiences. We propose the following two features to address this challenge.

Designing for device categories: We propose to extend the hierarchical tree model to support for hierarchies of generic device categories rather than specific devices. This allows the developers to think about and design experiences for groups of devices while the underlying toolkit manages the heterogeneous nature of such groups. Developers can declare a category of devices (e.g., smartphones or smartwatches) that interact within an interactive space for a given scenario. In addition, the developers can set the maximum number of devices supported within a device category based on the context and limitations of use. This information is used to deploy a generic Javascript application for each declared device category. We believe that this conceptual framework breaks down the barrier for entry for authoring such interaction scenarios.

Supporting adaptability of UI and interactions: Breaking down of application features by device categories may not be sufficiently granular for certain interaction scenarios. In addition, this does not guarantee that an application will run on all the *walk-in* devices as envisioned by the developer. We propose custom XML tags that allow for declaring multiple versions of the user interface or UI elements and interaction logic during the design phase. The developers can include branching logic within these tags that may include specific device properties (eg. specific UI layouts are targeted for specific range of screen sizes OR specific interaction scenarios run only if an accelerometer is present) or specific user permissions. These alternate versions of interface or interaction logic are selected & deployed at runtime. This is made possible by the automated API calls that query a newly joined interaction device for its capabilities. This approach allows the deployment of a generic interaction scenario with high level of adaptability at runtime. It also provides an easy method for developers to support graceful degradation of rich interactive experiences.

Challenge 2: BYOD scenarios may need to support authentication and permission management for different groups of users and different networks of devices.

While a single communication network and a common device permission may be sufficient for most BYOD scenarios, certain scenarios may require developers to support different permission levels for different user groups as well as support devices that may be on different networks altogether. We propose two additional features to extend the toolkit's capabilities.

Multi-Step Device Handshake: Currently every client application developed using the RE/Tk goes through a single-step handshake procedure which queries the device for its capabilities to automatically identify its role within the interaction scenario. We introduce multi-step device handshake procedure to include additional initialization routines. Firstly, we extend the device feature querying to be exhaustive that better supports the adaptability of UI and interactions previously discussed. In addition, we include an authentication phase that allows a walk-in device to be authenticated as a valid interaction device and be assigned a specific permission level. This may happen prior to the application launch as well. Lastly, we include a third step to the handshake procedure - user-side permission management. In BYOD scenarios where a walk-user can share information from her device within the interaction space, this step gives her the control to manage permissions for what gets shared. The user can also control what features of her device can be used within the interaction scenario.

Multi-step device-handshake procedures can be customized via the XML files and extended to more steps as required.

Device Authentication and Heterogeneous Networks: There are two points of entry for a typical BYOD scenario. A walk-in user can connect to a local wireless network thereby gaining access to an interaction experience through a web browser or a native application. Alternatively, users can access a specific URL on the world wide web that points to the interactive application. Since users gain such knowledge when they are in the proximity of the physical interaction space, this is a sufficient level of authentication for most BYOD scenarios.

For further permission management, web-based applications generated by the RE/Tk can take advantage of both the points of entry. Multiple access points can be placed for multiple levels of authentication to the same interactive experience. A simpler approach is to provide a unique passphrase that walk-in users can enter during the initial application launch. The passphrase, combined with the branching interaction logic previously mentioned, provides sufficient granularity for multiple levels of access control. The web-based nature of application and access control ensures that devices in different networks can be managed easily through a single interface as long as the underlying application is deployed onto the world wide web. In situations where devices are connected via Bluetooth, USB or other forms of communication, the RE/Tk provides custom modules to connect with different communication protocols. All of the different devices and protocols communicate with a server-side routing logic that is abstracted by the API. This ensures that the developer can ensure homogeneity of communication and interaction among heterogeneous mix of devices and networks with minimal effort.

Workshop Goals
Our proposed changes to the RE/Tk and the underlying conceptual framework enables us to provide a powerful prototyping tool that simplifies the design of interaction scenarios where target devices are not known. We hope to engage the community in a discussion of further opportunities and challenges for tools that support interactions "in the wild". In addition, we wish to contribute to the efforts of standardizing protocols, platforms, and tools for building cross-device interfaces.

Acknowledgements
This work has been supported by the Canada Research Chairs program and the 2015 UC3M postdoctoral mobility scholarship.

References
1. Xiang ‘Anthony’ Chen, Tovi Grossman, Daniel Wigdor, & George Fitzmaurice. 2014. Duet:

exploring joint interactions on a smart phone and a smart watch. *In Proc. CHI'14*, 159-168.

2. Saul Greenberg and Chester Fitchett. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proc. UIST'01*.
3. Peter Hamilton and Daniel J. Wigdor. 2014. Conductor: enabling and understanding cross-device interaction. In Proc. *CHI '14*, 2773-2782.
4. Scott R. Klemmer and James A. Landay. 2009. Toolkit Support for Integrating Physical and Digital Interactions. *Human-Computer Interaction* 24(3), 315–366.
5. Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. In Proc. *CHI'11*, 315–324.
6. Aneesh P. Tarun, Ahmed S. Arif, Andrea Bellucci, & Ali Mazalek. 2015. Responsive Objects, Surfaces and Spaces (ROSS): Framework for Simplifying Cross-Device Communication. In *TEI'15 Workshop on Interactive Infrastructures – Towards a Language for Distributed Interfaces*.
7. Jishuo Yang and Daniel J. Wigdor. 2014. Panelrama: enabling easy specification of cross-device web applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*(CHI '14). ACM, New York, NY, USA, 2783-2792.
8. Pei-Yu (Peggy) Chi and Yang Li. 2015. Weave: Scripting Cross-Device Wearable Interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 3923-3932.
9. Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 1247-1256.

Extending a Learning Platform with Cross-Device Functionality

Maria Husmann
Department of Computer Science
ETH Zurich
husmann@inf.ethz.ch

Nicola Marcacci Rossi
Department of Computer Science
ETH Zurich
nicolamr@student.ethz.ch

Moira C. Norrie
Department of Computer Science
ETH Zurich
norrie@inf.ethz.ch

Abstract

We report on our experience of adding cross-device functionality to a learning platform with a substantial user base. The extension allows students to display an exercise sheet on one device (typically a notebook) and use a paired handheld device to submit photographed solutions to the exercise. We outline the design and implementation of the feature and discuss lessons learned in the process of working outside of a controlled lab environment.

Author Keywords

cross-device; in-the-wild; education.

ACM Classification Keywords

H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Introduction

Today's device ecologies offer exciting opportunities for interactions between multiple devices [3]. Tools and frameworks facilitate the design [4, 2] and implementation [5, 6, 1] of cross-device applications. Nevertheless, we have encountered few such applications in the wild. Two notable exceptions have been extensions to Google Maps[1] and YouTube[2]. Both of these are small extensions to appli-

[1] https://support.google.com/maps/answer/6081481
[2] https://support.google.com/chromecast/answer/2995235

cations with a huge user base. Building up a user base is challenging for any application. Having access to an existing base is thus an interesting opportunity to explore cross-device designs in the wild, but comes with the challenge of having to integrate with an existing system.

We had the opportunity to extend a web-based education platform called Taskbase[3] with cross-device functionality. Taskbase manages collections of theory and exercise materials and allows exercises to be grouped into exercise sheets and published to a class of students. Students can give feedback on the exercises and rate their difficulty. Taskbase is currently in use at several schools and universities in Switzerland, including ETH Zurich with over 3500 users in the mathematics department and the University of St. Gallen with over a 1000 users. ETH Zurich has its own installation[4] and this was used in the work described here. The platform was developed by the startup company edTechLab in cooperation with the mathematics department of ETH. The authors of this paper were not involved with development prior to the cross-device project that we describe in this paper. However, one of the authors is acquainted with the CEO of edTechLab and the company expressed interest in exploring cross-device functionality in Taskbase.

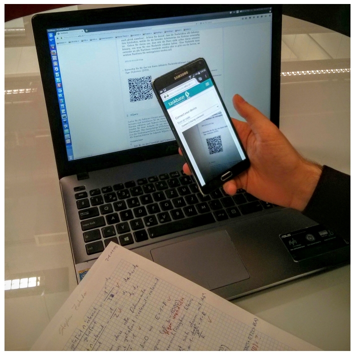
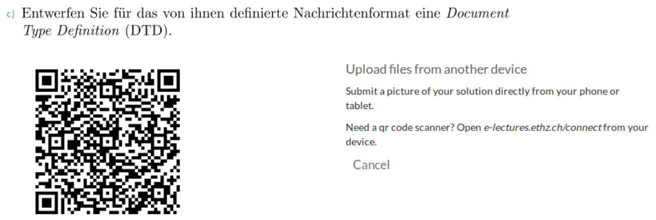


Figure 1: The student scans a QR code on their notebook.

Cross-Device Extension

We started by analysing how Taskbase is used and where there is potential added value with a cross-device extension. We then designed a cross-device feature and, after consulting with edTechLab, implemented it in Taskbase.

[3]http://www.edtechlab.ch/taskbase
[4]https://e-lectures.ethz.ch/


c) Entwerfen Sie für das von ihnen definierte Nachrichtenformat eine *Document Type Definition* (DTD).

Upload files from another device

Submit a picture of your solution directly from your phone or tablet.

Need a qr code scanner? Open *e-lectures.ethz.ch/connect* from your device.

Figure 2: For each exercise, a QR code is displayed that takes the student to the exercise submission site when scanned with a phone.

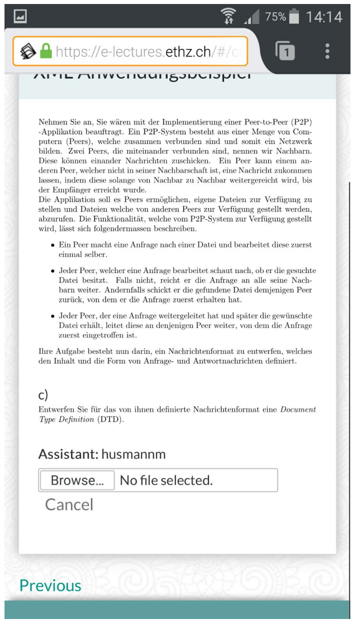
Analysis

While the exercise sheets can be printed, instructors observed that students often use their devices (notebooks or tablets) to access them during exercise sessions. This was confirmed when we analysed access patterns to the system. We also noted that many students used more than one device to access the platform. Within one week, we observed roughly 2700 users accessing the platform with two different devices, while fewer than 400 users only used a single device. Fewer than 100 users accessed Taskbase with three or more devices.

In our own teaching (independent of Taskbase), we are increasingly experiencing students submitting either scanned or photographed versions of their handwritten assignments via email. The teaching assistant then prints the submissions, marks them on paper, and returns them in the next exercise session.

Design

Based on our analysis, we decided to design a feature that allows students to submit their handwritten solutions to exercises using the camera of their phones or tablets. An exercise sheet may consist of multiple exercises and a stu-



Nehmen Sie an, Sie wären mit der Implementierung einer Peer-to-Peer (P2P) -Applikation beauftragt. Ein P2P-System besteht aus einer Menge von Computern (Peers), welche zusammen verbunden sind und somit ein Netzwerk bilden. Zwei Peers, die miteinander verbunden sind, nennen wir Nachbarn. Diese können einander Nachrichten zuschicken. Ein Peer kann einem anderen Peer, welcher nicht in seiner Nachbarschaft ist, eine Nachricht zukommen lassen, indem diese solange von Nachbar zu Nachbar weitergereicht wird, bis der Empfänger erreicht wurde.
Die Applikation soll es Peers ermöglichen, eigene Dateien zur Verfügung zu stellen und Dateien welche von anderen Peers zur Verfügung gestellt werden, abzurufen. Die Funktionalität, welche vom P2P-System zur Verfügung gestellt wird, lässt sich folgendermassen beschreiben.

- Ein Peer macht eine Anfrage nach einer Datei und bearbeitet diese zuerst einmal selber.
- Jeder Peer, welcher eine Anfrage bearbeitet schaut nach, ob er die gesuchte Datei besitzt. Falls nicht, reicht er die Anfrage an alle seine Nachbarn weiter. Andernfalls schickt er die gefundene Datei demjenigen Peer zurück, von dem er die Anfrage zuerst erhalten hat.
- Jeder Peer, der eine Anfrage weitergeleitet hat und später die gewünschte Datei erhält, leitet diese an denjenigen Peer weiter, von dem die Anfrage zuerst eingetroffen ist.

Ihre Aufgabe besteht nun darin, ein Nachrichtenformat zu entwerfen, welches den Inhalt und die Form von Anfrage- und Antwortnachrichten definiert.

c)

Entwerfen Sie für das von ihnen definierte Nachrichtenformat eine *Document Type Definition* (DTD).

Assistant: husmannm

Figure 3: After scanning the QR code, the student is taken to the corresponding exercise on their phone.

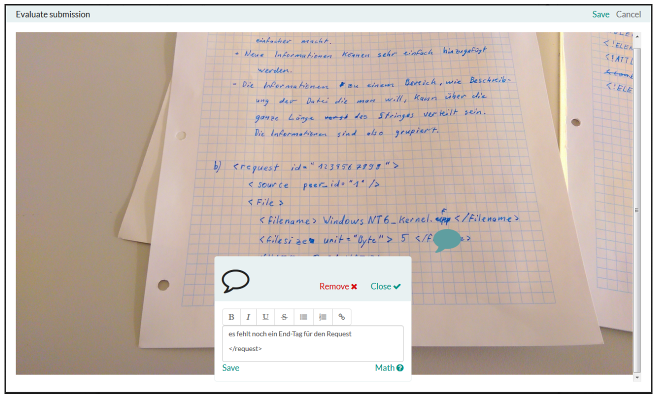
dent can submit a solution for each exercise. The interface offers options to directly upload a file from the current device (typically a notebook or a desktop computer) or to use another device for uploading. If the latter option is chosen, the system displays a QR code (Fig. 2). When the code is scanned with a phone or tablet (Fig. 1), it opens a URL that points to the same exercise as that currently opened on the first device (Fig. 3) and the student is automatically logged in. Now they can simply photograph their solution (Fig. 5) and it will be associated with the exercise and uploaded to Taskbase. The responsible teaching assistant can access all submissions from their students and provide feedback as annotations on the pictures (Fig. 4) which can then be viewed by the student.

Implementation

As Taskbase is already in use, it was a clear goal not to disrupt users in any way and to keep the implementation as lightweight as possible, while integrating with the current architecture based on Java on the backend and AngularJS on the frontend. As current cross-device frameworks are still experimental, it was deemed too large a risk to integrate one of them and the changes were kept to a bare minimum. Consequently, no direct communication between the devices was implemented. Consequently, when a students scans their submission with their mobile phone, they need to manually refresh the notebook to see their submission. While we would have preferred synchronisation between the two devices, we had to compromise for the sake of complying with the current architecture which does not support any push messages from the server.

Deployment

As the electronic hand-in is only useful if assistants are willing to accept submissions made through the system, the feature was disabled by default and had to be enabled per



es fehlt noch ein End-Tag für den Request
</request>

Figure 4: The UI for evaluating submitted exercises. Teachers or assistants can add comments and give feedback.

course. Taskbase then sent an email to professors informing them of the change and asking for volunteers to opt in. Unfortunately, there were few responses and the ones that we got were negative. The feature was rolled out in the second half of the semester and professors were reluctant to change the process half-way in. One professor worried that the feature would decrease personal interaction between assistants and students and feared dehumanisation of the process. We therefore decided to do a pilot test of the feature in one of the author's classes which had not been using Taskbase previously. The platform was introduced and the feature was demonstrated in class. Students were encouraged to use it, but were still allowed to hand in their solutions in person or by email. In total 44 students were enrolled in that class and roughly 30 students typically attended the exercise sessions. Handing in assignments was voluntary (a master solution was provided) and generally done by 6 to 10 students to get feedback. After the introduction of the cross-device feature, two students submitted

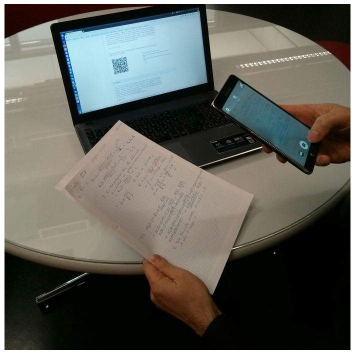


Figure 5: The student takes a picture of their solution and uploads it to the system.

their solution via their phone. 6 students used the new UI to upload a file from their notebook. No more assignments were submitted in person or by email. One student reported a problem with his installed QR code scanner that could not display the Taskbase website correctly. The rather low number of photographed submissions can be partly explained by the nature of the exercise which was better suited to being solved on a computer as well as the fact that it was the last exercise of the semester where the number of submissions of non-mandatory assignments typically drops.

Lessons Learned

While we had experience in developing cross-device applications, we typically work in a controlled environment where we can choose the architectures, technologies and devices and have full control over the applications that we build. Users are introduced to our applications in user studies or demonstrations where we, again, control many parameters. Working with Taskbase forced us to give up some of that control, confronted us with real users and stakeholders, and taught us the following lessons.

Focus on the user, not the devices. While our main interest was in integrating the phone, a significant amount of time was spent on the UI for giving feedback to the assignments. Had we skipped that part, students would have had no reason to use the system to hand in their solutions. It was important to develop a whole use-case, rather than just focusing on the cross-device part. All in all, a small fraction of time was spent on actual cross-device development.

Stakeholders are not necessarily excited about making something cross-device. Mainly students and teaching assistants benefited from our changes and the Taskbase team was very supportive. Professors however were more sceptical but important decision makers. Getting them on board will be critical for the success of the cross-device feature.

Cross-device frameworks may introduce a big change to the existing architecture. Cross-device frameworks typically introduce additional servers or services and protocols (for example WebSockets). Furthermore, companies may shy away from using frameworks that have not been proven to be production ready. In our case, the risk and effort was considered too high for a simple feature and, despite our experience with our own framework[5], we decided not to use it.

Don't make any assumptions about software and hardware. Provide alternatives to support a wide range of platforms and devices. We integrated a web-based QR scanner that could be accessed from a short URL for those users who had none installed. Our system required no installation on any device, thus maintaining a low barrier to entry. We did not force users to use their phones for submissions but rather supported file uploads from any device.

Carefully plan the introduction to the users. In our case, the timing of the introduction towards the end of the semester was not ideal. However, we were constrained by the timing of the student project and could not wait until the beginning of the next semester. Also, consider how users can learn to use cross-device features. The demonstration in class proved to be a good solution, however, it does not scale very well to a large user base.

Acknowledgements

We would like to thank edTechLab for giving us the opportunity to work with them and all their support. This project was supported by grant No. 150189 of the Swiss National Science Foundation (SNF).

[5]https://github.com/mhusm/XD-MVC

REFERENCES

1. Sriram Karthik Badam and Niklas Elmqvist. 2014. PolyChrome: A Cross-Device Framework for Collaborative Web Visualization. In *Proc. ITS*. DOI: http://dx.doi.org/10.1145/2669485.2669518

2. Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1247–1256. http://doi.acm.org/10.1145/2702123.2702215

3. Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 315–326. http://doi.acm.org/10.1145/2047196.2047238

4. Michael Nebeling, Theano Mintsi, Maria Husmann, and Moira C. Norrie. 2014. Interactive Development of Cross-Device User Interfaces. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2793–2802. http://doi.acm.org/10.1145/2556288.2556980

5. Mario Schreiner, Roman Rädle, Hans-Christian Jetter, and Harald Reiterer. 2015. Connichiwa: A Framework for Cross-Device Web Applications. In *Proc. CHI EA*. DOI:http://dx.doi.org/10.1145/2702613.2732909

6. Jishuo Yang and Daniel Wigdor. 2014. Panelrama: Enabling Easy Specification of Cross-Device Web Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2783–2792. http://doi.acm.org/10.1145/2556288.2557199

Enabling multi-device interaction on the go in the MAGI project

Kenny T.W. Choo
School of Information Systems
Singapore Management University
80 Stamford Road
Singapore 178902
kenny.choo.2012@smu.edu.sg

Quentin Roy
School of Information Systems
Singapore Management University
80 Stamford Road
Singapore 178902
quentin@quentinroy.fr

Richard C. Davis
School of Information Systems
Singapore Management University
80 Stamford Road
Singapore 178902
rcdavis@smu.edu.sg

Abstract

We discuss the MAGI project's vision for multi-device interaction and how we plan to support application developers. Key aspects of our vision are the presence of multi-device gestures and adaptation to changing physical contexts and device contexts. Our gesture recognition architecture reduces power consumption and recognition latency through a pipelined HMM approach with early discard of processing samples of unlikely candidates. To help developers build applications that adapt to changing contexts, we propose the use of *Midgets* (MAGI widgets); context-dependent interaction components that span across devices. Application designers choose the device configurations that an application will support, assign them to physical contexts, and lay out *Midgets* manually. The MAGI system chooses the best configuration based on users' current activity and the input/output channels available.

Author Keywords

Wearable devices; Multi-device interaction; Distributed User-Interface; Gesture; Gesture recognition; Power consumption; Distributed recognition.

ACM Classification Keywords

H.5.2 [Information interfaces and presentation]: User Interfaces – Input Devices and Strategies.

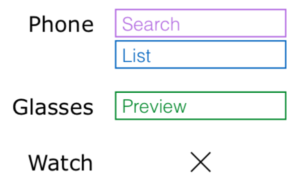


Figure 1: *Midgets* running on each device in scenario 2 part (a): seated with the phone out.

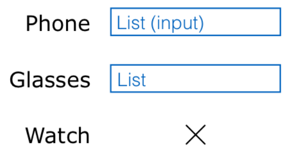


Figure 2: *Midgets* in scenario 2 part (b): walking with phone in hand.

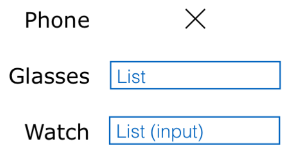


Figure 3: *Midgets* in scenario 2 part (c): sitting without phone.

Introduction

Wearable devices such as smart watches are gaining widespread interest from consumers. While wearables provide easy access to content outputs such as notifications, emails or text messages, each device has limited input and output capabilities. We envision a personal multi-device ecosystem where applications span across all of a user's devices; ranging from smart watches, smart glasses, smart bands to smart phones. Inputs and outputs are distributed automatically, depending on what devices are available and the automatically detected context of use (e.g. walking, sitting or cycling).

This paper briefly outlines our vision, called MAGI (Multi-device Adaptive Gestures & Interfaces) [3]. We aim to enable developers with an Android framework that provides high-performance, multi-device gesture recognition and primitives for interface adaptation. Here, we illustrate this vision with two scenarios. We then show how we are addressing the challenges of multi-device gesture recognition with a novel pipelined HMM approach with early discard of unlikely candidates. Finally, we describe our vision for *Midgets* (MAGI Widgets), which are context dependent hierarchical interaction components that span across devices.

Multi-Device Scenarios

Consider the following scenarios, which show the types of applications that we intend MAGI to support.

Scenario 1: Steering in a driving game.
While riding on a train, Alice is playing a driving game on her virtual reality display, smart watch and smart phone. With her watch on one hand, and phone in the other, she turns an imaginary steering wheel. She uses the touchscreen buttons on the phone to accelerate or brake. Her phone vibrates, indicating an incoming call, and Alice raises her phone to her ear. When she does this, the game pauses, and Alice takes her call.

Scenario 2: News feed consultation.
(a) Clara is sitting at a café, browsing the titles of her news feed on her phone. As she scrolls through the list of articles, her augmented reality (AR) glasses preview the first lines of the topmost article at the periphery her vision. A search field allows her to look for specific content (see Figure 1). (b) Later, Clara gets up and starts walking to the metro station, holding her smartphone at her side, without looking at it. The list of articles moves to her glasses where she can see it, and she scrolls the list by swiping on her phone (see Figure 2). (c) When Clara boards the crowded metro train, she puts her phone in her pocket to keep it safe. Still browsing articles on her glasses, she now begins scrolling by swiping on her watch (see Figure 3).

These scenarios illustrate two key challenges. The first scenario uses a continuous, multi-device gesture that is particularly sensitive to latency. Recognizing such gestures will place high processing demands on wearable devices, which increases power consumption. The second scenario shows a user's changing context. The application must adapt by choosing optimal input and output channel. MAGI will address these challenges.

Gesture Recognition Challenges

Each device employed in the personal space will have to deal with two problems (1) recognition latency, and (2) energy consumption. By nature of their size, the computational capabilities and energy resources of mobile devices are limited. Most previous works adopt a

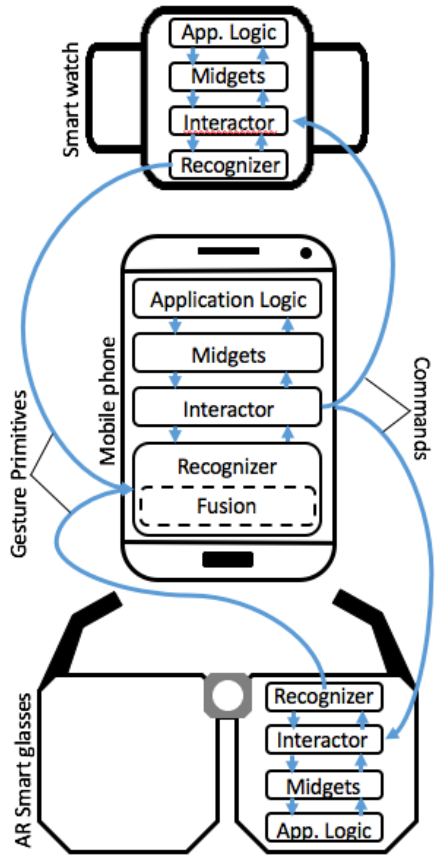


Figure 4. The MAGI architecture supports distributed gestural interfaces that are able to automatically adapt themselves to different contexts of use.

centralized approach [1,4]. Sensor inputs (e.g. accelerometer and gyroscope data) are all streamed to a single device that handles the recognition process. This approach has two disadvantages. The first is continuous sensor data transmission, which consumes excessive energy in both the sender and the receiver. The second is poor scaling. As the number of devices in the ecosystem increases, a centralized recognizer on a mobile device can rapidly be overwhelmed. We propose to tackle these demands through collaborative and adaptive systems and algorithms.

Previously, we developed an on-device, early-filtering Hidden Markov Model-based (HMM) gesture recognizer that improves the speed and reduces the latency (and hence energy efficiency) of two-handed gesture recognition (see Figure 4) [3]. Gesture recognition is distributed as much as possible to reduce network usage. The method attempts to recognize gestures early. If an entire gesture is recognized with high likelihood by an HMM, then a prefix (an initial part) of the gesture should also have high likelihood. Our approach thus permits *early discard*, avoiding processing samples whose prefix indicate that it is unlikely to be a gesture. Also, our method is heavily *pipelined*, reducing the latency of gesture detection by performing Viterbi decoding concurrently with the generation of subsequent gesture samples. Since our method performs gesture recognition on-device, we significantly reduce the throughput which otherwise would comprise sending multiple raw sensor feeds into a central device and having that perform all the work. In the tests reported previously, our system recognized gestures with 89.86% accuracy in approximately 0.2ms [3].

In the future, we plan to let devices assist each other in recognition process by notifying them when it is safe to halt an HMM for a candidate gesture. An advantage of performing gesture recognition on-device is that with independent recognition streams and continuous gesture fusion, any one of the streams may provide enough information about the activity such as to trigger early termination of some gesture candidates for processing or terminating recognition altogether on any of the other devices. For example, while executing a steering gesture in a driving game (see Scenario 1), the movement of one hand (equipped with a smart watch) out of the norm for steering gestures, may indicate termination of processing on the smart watch on the other hand, which can indicate to the system that the game should be paused.

Context Adaptation Challenges

Previous researchers have studied automatic layout of distributed applications depending on device context (e.g. Panelrama [4]). Such approaches can lead to unpredictable behavior. We prefer to give designers more precise control over the input and output channels that their application will use, while still allowing easy coordination of components and adaptation to context. This led us to develop the concept of *Midgets*, which are context dependent interaction components that span across devices. *Midgets* are shared across devices in an application, but they can have different behavior depending both on a user’s device context and physical context (recognized automatically, as in [2]).

Consider the application from Scenario 2 in three contexts. At first (Figure 1), a phone and a pair of glasses are used in conjunction. The *Midgets* of the application are spread across the two devices. When the user starts walking and stops looking at her phone, the phone screen switches to input only and the glasses hold all visual information (Figure 2). Showing all *Midgets* at once on the glass may not be a good design choice as it is likely to occlude a large portion of the user's vision. Thus, in this context, a designer may prefer to give access to only one *Midget* at a time. Finally, when the phone is put away (Figure 3) input switches from the phone to another device.

Our vision gives designers more control, because designers will specify all the device configurations allowed by an application and will have an opportunity to design each one separately. The designer will also specify the context when each configuration is preferred. Scenario 2, for example, uses three configurations. When sitting with the phone and glass available, configuration (a) in Figure 1 is preferred. When walking with phone in hand, configuration (b) in Figure 2 is preferred. When sitting with phone unavailable, configuration (c) in Figure 3 is preferred. MAGI will determine the user's context and choose the best configuration, allowing the user to override when context recognition errors occur.

Inspired by Panelrama's method of automatically arranging output layouts [4], we are also considering adding an automatic layout engine so that a designer can set some *Midgets* to automatically re-arrange themselves without additional intervention.

Conclusion

We presented our vision of a multi-device ecosystem used on the go. We addressed system-related issues with a new distributed recognition approach, presented the architecture of our multi-device framework and discussed how it can be used by a designer to create applications that automatically adapt their inputs and outputs to their context of use.

This first work opens several challenges we are interested to address in the future. What *Midgets* will a designer need to create robust applications? How can we manage multi-application contexts and the inputs and outputs allocation between concurrent applications? How can we support multi-user applications and allow applications to span across personal ecosystems?

Acknowledgements

This work was supported by Singapore Ministry of Education Academic Research Fund Tier 2 under research grant MOE2014-T2-1063. All findings and recommendations are those of the authors and do not necessarily reflect the views of the granting agency, or Singapore Management University.

References

[1] Houben, S. and Marquardt, N. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. *Proc. of CHI'15*, ACM (2015), 1247–1256.

[2] Lee, Y., Iyengar, S.S., Min, C., et al. MobiCon: A Mobile Context-monitoring Platform. *Commun. ACM 55*, 3 (2012), 54–65.

[3] Tran, V.H., Choo, K.T.W., Lee, Y., Davis, R.C., and Misra, A. MAGI: Enabling Multi-Device Gestural Applications. *PerCom'16 Workshops*, IEEE (2016).

[4] Yang, J. and Wigdor, D. Panelrama: Enabling Easy Specification of Cross-device Web Applications. *Proc. CHI'14*, ACM (2014), 2783–2792.

TomoSense: Towards Multi-Device Spatial Awareness Based on Independent Plane Sensing

Przemysław Kucharski
Institute of Applied Computer Science
Lodz University of Technology
Łódź, Poland

Andrzej Romanowski
Institute of Applied Computer Science
Lodz University of Technology
Łódź, Poland
androm@iis.p.lodz.pl

Krzysztof Grudzień
Institute of Applied Computer Science
Lodz University of Technology
Łódź, Poland

Paweł W. Woźniak
Institute for Visualisation and Interactive Systems
University of Stuttgart
Stuttgart, Germany

Abstract
We present a plane surface non-invasive sensing system for developing new cross-device interaction in the wild, based on electrical capacitance tomography (ECT) measurement setup. The core element of the system is a plane capacitance sensor consisting of 32 electrodes built in underneath the surface of an experimental tabletop assembly. The interaction is enabled through physical objects interfering with the electrostatic field in proximity of the surface. This principle of operation is shown here for a set of mobile devices in order to develop spatially-aware applications. The system is independent in terms of sensing. It can detect all solid objects in its proximity. Here, we present the basic features of TomoTable and discuss future usage scenarios for the sensing system.

Author Keywords
position sensing; tomography; multi-device environments

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

Introduction
As recent research shows that many rationales for using multi-device systems in the wild exist [6, 3], spatial awareness emerges as one of the key enablers for rich multi-

device interactions. Recent work on sensemaking [7] shows specific scenarios for multi-device interactions emerging while the range of possible areas where users already use multiple devices simultaneously grows. Yet, the sensing techniques allowing for easy access to such interactive environments are still limited. Most research on enabling spatial awareness involves either external sensors/cameras (which suffer from line-of-sight and other issues e.g. [5]). Consequently, most of solutions are explored only in the laboratory conditions and provide limited possibilities for in-the-wild studies. As an alternative solution, we propose a TomoTable — an ECT-based prototype sensing system that uses a plane capacitance sensor easy to be embedded within an ordinary table. The sensor is accompanied by process tomography measurement equipment and a dedicated measurement protocol. TomoTable enables identifying devices in the close proximity of the electrodes that are hidden below the measurement plane (e.g. the surface of a table) thus enabling ad-hoc interactions in different locations as proposed in [4]. This paper contains a short technical overview of the prototype concept as well as preliminary results and a discussion on further studies.

TomoTable

TomoTable is a rectangular planar sensor that uses electrical process tomography measurement. It consists of 32 electrodes arranged in 4 rows and 8 columns, all embedded just below the surface of an ordinary table (Figure 1). The sensor is intentionally made visible (showing the principle of operation, it can be easily hidden below an opaque layer during normal operation) underneath a transparent plastic layer ($3mm$ thick) of polycarbonate. It is worth noting that the system can be a part of virtually any other surface as well.

The table we currently use is $450mm$ x $610mm$ while each electrode is $95mm$ x $60mm$. The gaps in between the electrodes are $10mm$. The bottom part of the sensor ($10mm$ below the table surface) was electrically screened in order to improve the signal-to-noise ratio (SNR), thus improving the sensing properties of the system in the space above the sensor. Electrical capacitance tomography is based on the principle of measuring the change of capacitance between all the consecutive pairs of sensing electrodes irrespective of their position or orientation by quickly switching the excitation to successive electrodes while grounding the rest. Experiments were conducted using 32-channel equipment capable of real time on-the-fly monitoring of the measurement space [1]. Next, we show 2D reconstructed images (using a basic LBP algorithm to illustrate the working concept of the system). The images, however are not needed to design spatially-aware interactions, reveal potential of the system and the image processing algorithm can simple output the relative positions of the devices on the sensing surface.

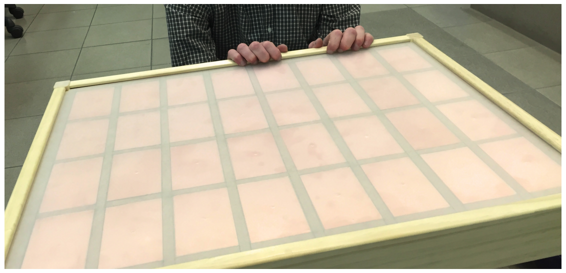


Figure 1: TomoTable embedded in a table. A semitransparent surface is used to show the sensors.

Preliminary results

Despite us using an early prototype with basic image processing and a low sensing resolution, the size and orien-

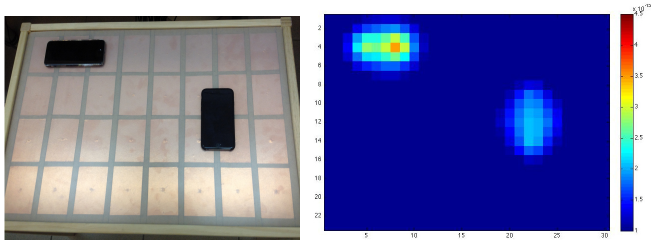


Figure 2: Two iPhone 6 units placed on TomoTable sensor (left figure) and the resulting reconstructed image (right figure). The color scale show measurement intensity that varies from empty space (navy blue) to the fully occupied space above the sensor (dark brown).

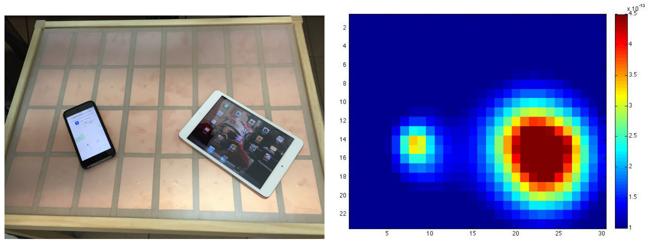


Figure 3: An iPhone 6 and an iPad mini placed on TomoTable sensor (left figure) and the resulting reconstructed image (right figure). The spatial arrangement of the devices is misaligned with TomoTable´s electrode array. Such a case is more likely in an in-the-wild scenario.

tation of the devices can be easily seen on the resulting reconstructed image. We show two cases with different orientation of the devices with respect to each other and different alignments with respect to the orientation of the electrodes (Figures 2 and 3, left) as well as the resulting reconstructed images of the distribution of the objects on the sensed surface (Figures 2 and 3, right).

There are observable differences in the produced images depending on how the devices are positioned in relation to the electrode array and other devices. This showcases the potential of electrical capacitance tomography to provide accurate sensing for multi-device systems. We demonstrate that it is feasible to: (1) obtain the position of the device (2) distinguish the size of the device (3) detect the rotation angle and (4) sense several distinct devices.

Discussion and further work

Measurement records taken during experiments revealed significant changes in sensed capacitance which are several orders of magnitude higher than the sensing resolution of the processing unit used. This leads to a conclusion that simpler and cheaper embedded devices can be used to provide accurate sensing for the table. Moreover, the system design and operational foundations in terms of both simplicity and the measurement protocol make the equipment easily scalable. Future work will include refining the measurement protocol coupled with dedicated contextual data processing. With the help of inverse problem solving techniques, we hope to achieve increased accuracy without the need for extensive additional computation.

We hope that TomoTable will provide opportunities for in-the-wild studies of multi-device systems. If our development progresses as planned, we imagine that we will be able to introduce multi-device setups to meeting places such as pubs or cafés. Much like in the case of interactive tabletops (e.g. [2]), we hope that observational studies of how users interact with such systems over a long period of time will yield new insights that will inspire new interaction techniques.

Conclusions

This paper gives an overview of the technical concept of TomoTable - an ECT-based position sensing system for developing of interactive applications in multi-device environments. While spatial awareness may emerge as a key interaction paradigm for multi-device environments, a question of a balance between the required sensing accuracy and the complexity and ease of deployment remains open. Our prototype offers limited spatial resolution compared to more complex and expensive systems i.e. marker-based motion tracking. However, further research on refining the design in terms of different electrode arrangements coupled with dedicated measurement protocols as well as contextual data processing algorithms shall provide sufficient accuracy. The key advantage of TomoTable is that it has a large potential for easy embedding in existing meeting environments. As it can be easily made visible, we anticipate it will enable us to run in-the-wild studies with multi-device systems.

References

[1] Paweł Fiderek, Tomasz Jaworski, Radosław Wajman, and Jacek Kucharski. 2015. Fuzzy Clustering Of Raw Three Dimensional Tomographic Data For Two-Phase Flows Recognition. *IAPGOS* 5(4) (2015), 12–15. DOI: http://dx.doi.org/10.5604/20830157.1176565

[2] Uta Hinrichs and Sheelagh Carpendale. 2011. Gestures in the wild: studying multi-touch gesture sequences on interactive tabletop exhibits. In *Proc. CHI '11*. ACM, 3023–3032. DOI: http://dx.doi.org/10.1145/1978942.1979391

[3] Tero Jokela, Jarno Ojala, and Thomas Olsson. 2015. A Diary Study on Combining Multiple Information Devices in Everyday Activities and Tasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, 3903–3912. DOI: http://dx.doi.org/10.1145/2702123.2702211

[4] Christian Müller-Tomfelde and Morten Fjeld. 2012. Tabletops: Interactive Horizontal Displays for Ubiquitous Computing. *Computer* 45, 2 (2012), 78–81. DOI: http://dx.doi.org/10.1109/MC.2012.64

[5] Roman Rädle, Hans-Christian Jetter, Nicolai Marquardt, Harald Reiterer, and Yvonne Rogers. 2014. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces - ITS '14*. ACM Press, 45–54. DOI: http://dx.doi.org/10.1145/2669485.2669500

[6] Stephanie Santosa and Daniel Wigdor. 2013. A field study of multi-device workflows in distributed workspaces. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing - UbiComp '13*. ACM Press, 63. DOI: http://dx.doi.org/10.1145/2493432.2493476

[7] Paweł Wozniak, Nitesh Goyal, Przemysław Kucharski, Lars Lischke, Sven Mayer, and Morten Fjeld. 2016. RAMPARTS: Supporting Sensemaking with Spatially-Aware Mobile Interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2447–2460. DOI: http://dx.doi.org/10.1145/2858036.2858491

"Bring-your-own-app?!" – Why apps hinder us from achieving true cross-device BYOD interaction

Hans-Christian Jetter
University of Applied Sciences
Upper Austria
Hagenberg, Austria
hans-christian.jetter@fh-hagenberg.at

Abstract

I believe that our community is widely ignoring a fundamental challenge that stands between our ambitious visions for cross-device interaction and what we actually achieve when deploying our prototypes in the real world. The problem is that we still "think" in apps and design BYOD prototypes as apps for a few selected tasks and for clearly defined combinations and configurations of devices. We therefore support only a tiny fraction of the wealth of possible BYOD usages, device combinations, and collaboration styles. To build and observe BYOD technologies that fundamentally change how we interact with computing systems, we have to move beyond the concept of single BYOD apps and find ways to make our prototypes more *adaptable* and *interoperable* so that they support unanticipated and fundamentally new usage patterns in the wild.

Author Keywords

Cross-Device Interaction; Bring-your-own-device; Adaptivity; Adaptability; Interoperability; Instrumental Interaction; Object-Oriented User Interfaces; Webstrates.

Introduction

Most likely all participants of this workshop will agree that cross-device interaction promises a fascinating new way of using our increasingly diverse device ecosystems for solving real-world problems. Many of us share the vision of a world in which users can rapidly shape "symphonies" [4] or "communities" [6] of devices that feel like one seamless natural UI for cross-device applications. We hope to achieve this not only for single users but also for multiple users. As a result, our community has started to explore bring-your-own-device (BYOD) scenarios in which users join their personal devices to create a shared community of devices for collaboration (e.g. [14]).

Overall, HCI research has made great progress in this field. I will illustrate this by shamelessly using two examples from my own work: My work on ZOIL at the University of Konstanz [7] explored how the mobile and stationary devices inside a physical interactive space (e.g. tabletops, data walls, tablets, PCs) could be combined for multi-user sensemaking in a shared visual workspace. In subsequent work at UCL, we worked together to create a more lightweight and portable cross-device technology. The result was HuddleLamp [14] that enables users to combine off-the-shelf mobile devices for spontaneous collaboration simply by putting them under a desk lamp.

Lessons from ZOIL and HuddleLamp

Naturally both results are not perfect. For example, applications built with ZOIL are not easy to deploy in the wild because they can be installed only on Windows devices and need network connections that are often blocked by firewalls in real-world settings. As a result, we decided that HuddleLamp applications should become HTML5 browser applications to overcome the problem of incompatible devices or operating systems and eliminating the need for local installation or configuration of apps. Moreover, all communication should happen using web sockets, so that firewalls become much less problematic. This strategy proved to be very successful.

ZOIL also does not support BYOD scenarios: There is no possibility to easily detect the presence of a new device and automatically connect it without manually configuring network addresses and ports. In HuddleLamp this is much easier by simply opening a web page (e.g. by scanning a QR code) that will briefly flash a marker on the screen to identify the device and establish a connection without any manual setup.

A further problem is that ZOIL does not track spatial positions of devices, so that interactions or object transfers between devices require choosing device IDs from lists or placing objects in shared locations in a visual workspace. This feels much less fluid and more difficult than with other cross-device systems that extensively make use of inter-device spatial relations or *proxemics* [3]. I repeatedly discussed this important role of physical space and gestures for cross-device interaction in my work. Two examples are a workshop paper [9] at CHI 2014 and a resulting full paper at CHI 2015 [15]. As a consequence, already the very first ideas for HuddleLamp were centered around spatially-aware interactions [6,8]. However, all these interactions can only be detected inside the field of view of HuddleLamp's camera system. This is why Jin et al.'s recent work for sensing device locations without external hardware or device modifications is a very important step forward [10].

The really hard problems with BYOD apps

I believe that all above lessons and challenges have something in common: They will be solved within a few years. It is fair to assume that we will soon be able to detect, identify, and connect multiple devices of different types and sizes and to track their positions and the gestures between them. We also already know a lot about how to design such gestures, so what is the real challenge for BYOD in the wild?

In my opinion, there is a widely underrated challenge that stands between our ambitious visions for cross-device interaction and what we currently achieve when deploying our prototypes in the real-world. I believe that this challenge is only seldomly addressed in research yet, because it sits between the traditional research topics of HCI (e.g., gesture design, user studies, new sensors & algorithms) and software engineering (e.g., software architectures, distributed systems, standards for interoperability). Furthermore, it focuses on a concept that we are so familiar with that we find it "natural" and hardly recognize it as a deliberate design choice that has been made for us decades ago and that we need to challenge: the concept of packaging and distributing computing functionality as *applications* or *apps*.

Why are apps problematic for cross-device interaction and BYOD? First of all, they exacerbate the problem of *adaptability*. Monolithic walled apps are inherently bad at adapting to sudden changes in context, e.g. in the number and kind of present devices. As I discuss in [4], such changes will happen permanently and it is impossible to enumerate the set of contextual states that may exist. Therefore the traditional idea of designing an app for a clearly defined number and combination of devices and trying to predict all possible states cannot keep up with the complexity of real-world users and usage. If "bring-your-own-device" (BYOD) should not mean "bring-your-own-device *(as long as your task is T, you are using app A, your device is a phone running operating system X and browser Y, has a screen size of S, and there are only between N and M other devices involved)*" we must find more flexible and adaptable ways of providing functionality such as *commands*, *objects*, or *instruments*.

Second, there is the problem of missing *interoperability* between apps: Even apps that serve a very similar purpose, (e.g. different apps for taking notes with a stylus, different apps for visualizing data in bar charts) cannot talk to each other in BYOD settings. In the best case, they share a file format, so that data can be exchanged between them via cloud services by manually storing and opening files on different devices. But this is far from the seamless real-time collaboration across devices that we intend to realize. We must find alternatives that enables user to flexibly connect or combine functionality across devices, even in ways that were not anticipated by the developers but successfully emerge from usage in the wild.

If we keep on thinking about BYOD in terms of a single app with companion devices that can only be used according to the plan of the app's designer, we only scratch the surface of what could be achieved. But introducing new ideas how multiple devices could provide, share, distribute, and combines functionality and content across different devices in unanticipated ways could truly revolutionize the way how we interact with computing.

Alternatives to apps

The good news is that there are some alternatives out there. Researchers & practitioners have proposed a number of approaches for replacing the application-centric model with alternatives that are more flexible and arguably closer to the way we work and think in the real word, e.g. the *object-oriented user interfaces* (OOUIs) of the early 1990s [2], the *instrumental interaction* of the early 2000s [1], and its more recent incarnations as *VIGO* [11] or *Webstrates* [12]. Therefore I believe that we should use the current shift from the single-device to the multi-device era as an opportunity to critically reflect about the role that monolithic apps or applications should have in future and if alternatives such as *objects*, *instruments*, or *webstrates* would not meet the requirements of true BYOD and cross-device computing much more.

References

1. Beaudouin-Lafon, M. 2000. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In Proc *CHI '00*. 446-453.
2. Collins, D. 1995. *Designing Object-Oriented User Interfaces*. Benjamin/Cummings.
3. Greenberg, S., Marquardt, N., Ballendat, T., Diaz-Marino, R. and Wang, M. 2011. Proxemic interactions: the new ubicomp? *interactions*. 18, 1 (Jan. 2011), 42-50.
4. Hamilton, P. and Wigdor, D.J. 2014. Conductor: enabling and understanding cross-device interaction. In *Proc. CHI '14*, 2773–2782.
5. Jetter, H.-C., Zöllner, M., Gerken, J. and Reiterer, H. 2012. Design and Implementation of Post-WIMP Distributed User Interfaces with ZOIL. *International Journal of Human-Computer Interaction*. 28, 11 (2012), 737–747.
6. Jetter, H.-C. and Reiterer, H. 2013. Self-Organizing User Interfaces: Envisioning the Future of Ubicomp UIs. *Workshop "Blended Interaction" (CHI '13)*.
7. Jetter, H.-C. 2013. *Design and Implementation of Post-WIMP Interactive Spaces with the ZOIL Paradigm.* PhD Thesis, University of Konstanz.
8. Jetter, H.-C. 2013. Visual and Functional Adaptation in Ad-hoc Communities of Devices. *Workshop on Visual Adaptation of Interfaces (In conjunction with ITS '13).*
9. Jetter, H.-C. 2014. A Cognitive Perspective on Gestures, Manipulations, and Space in Future Multi-Device Interaction. *Workshop "Gesture-based Interaction Design" (In conjunction with CHI '14).*
10. Jin, H., Holz, C. and Hornbæk, K. 2015. Tracko: Ad-hoc Mobile 3D Tracking Using Bluetooth Low Energy and Inaudible Signals for Cross-Device Interaction. In *Proc UIST '15*. 147-156.
11. Klokmose, C. N. and Beaudouin-Lafon, M. 2009. VIGO: instrumental interaction in multi-surface environments. In *Proc CHI '09*, 869-878.
12. Klokmose, C. N., Eagan, J. R., Baader, S., Mackay, W., Beaudouin-Lafon, M. 2015. *Webstrates*: Shareable Dynamic Media. In *Proc UIST '15*. 280-290.
13. Marquardt, N., Hinckley, K. and Greenberg, S. 2012. Cross-device interaction via micro-mobility and f-formations. In *Proc UIST '12.*
14. Rädle, R., Jetter, H.-C., Marquardt, N., Reiterer, H. and Rogers, Y. 2014. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. In *Proc. ITS '14*, 45–54.
15. Rädle, R., Jetter, H.-C., Schreiner, M., Lu, Z., Reiterer, H. and Rogers, Y. 2015. Spatially-aware or spatially-agnostic? Elicitation and Evaluation of User-Defined Cross-Device Interactions. In *Proc. CHI '15,* 3913-3922.

Composition and mediation in cross-surface interaction

Henrik Korsgaard
Aarhus Univeristy
8200 Aarhus N, Denmark
korsgaard@cs.au.dk

Clemens Nylandsted Klokmose
Aarhus University
8200 aarhus N, Denmark
clemens@cs.au.dk

Abstract

In this position paper we propose two perspectives on interaction in cross-surface systems: compositon and mediation. We advocate for a focused effort to expand our theoretical and analytical vocabulary when it comes to cross-surface interaction.

Author Keywords
Cross-surface interaction; device ecologies; analytical tools

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

Introduction

Over the years we have build, deployed and studied a number of multi-device and cross-surface systems 'in the wild'. We have seen how the roles of devices and surface, whether large or small, can differ quite significantly given the use case and application: A personal device may e.g. provide a private interface for interacting with shared surfaces (as in our own Local Area Artworks [2]) or become part of a shared distributed interface (as in HuddleLamp [6]). Our vocabulary for talking about human-computer interaction beyond *one user—one device* is still limited, and we believe it is important to continuously refine and expand this vocabulary. In this position paper we propose two analytical perspectives on interaction with cross-surface systems: *composition* and *mediation*.

We build upon a growing body of work on taxonomies and theoretical frameworks for post-desktop, multi-device and cross-surface interaction. Notably Terrenghi et al. [9] examine multi-display ecosystems with the intent of understanding the relationship between scale (form-factor), social interaction and the interaction methods that couple devices and displays and make interaction possible. Müller et al. [4] have developed a taxonomy capitulating how people perceive public displays, interaction modalities and supported interaction. Sørensen et al. [8] present the 4C framework for (collaborative) interaction in digital ecosystems. The 4C framework derives principles of interaction design in digital ecosystems from a 2x2 matrix of 'many users' vs. 'many artifacts' and 'sequential' vs. 'simultaneous interaction'. The themes of the quadrants are communality (many sequential users), collaboration (many simultaneous users), continuity (many sequential devices) and complementarity (many simultaneous devices).

Theoretical premise

This work is part of our ongoing efforts in trying to grasp and theorise on the relationship between human activities and the role artifacts play. This work is strongly positioned within an activity theoretical understanding of activities, mediation and cultural-historical analysis of artifacts as crystallised knowledge. The core tenet of activity theory is that artifacts mediate human activity and that in order to understand artifacts we take the activity they are part of as the minimum meaningful unit of analysis. If a given artifact, device, software application, service etc., is used in an activity, we take that it has a meaningful instrumental role in the context of the activity, as it mediates intentional action and help users to realise specific goals.

With the proliferation of personal and ubiquitous computing, the artifacts available and their capabilities have changed significantly. In previous empirical work we have described these systems as artifact ecologies and made tentative distinctions related to some of their characteristics. The concept of artifact ecologies is socio-technical and encompasses both the actual technologies and how they are appropriated and used in meaningful activities. In the empirical and theoretical work, we have primarily focused on social aspects, e.g. the dynamics of personal artifact ecologies [1] and how a community appropriate and use multiple artifacts as part of their activities [3]. Individuals have a rich personal ecology of devices, although not always an active part of the activity at hand. Throughout an activity, a person only uses a subset of their ecology depending on the activity. We posit that the active artifacts are selected through an (unconscious) assessment between what is to be accomplished in a given activity and the potential artifacts knowingly available to the person – in the situation and in their knowledge of the artifacts capabilities. Here we distinguish between the *potential* and *actual* artifacts available to and used in a given activity. The potential is the "pool" from which an individual or group selects the actual artifacts to be used within the activity at hand (see also [7] on constellations of artifacts and group negotiation), and the actual artifacts are those that are part of the specific activity.

Perspectives on cross-surface systems

In the following we outline two perspectives on cross-device systems and artifact ecologies.

Composition. The composition is the actual artifacts in use as part of an activity. It may span multiple personal and shared devices which may or may not share resources or technical coupling. The composition might involve dedicated devices developed specifically for the particular activity or may be more or less impromptu use and coordination across heterogeneous devices – personal and shared. The composition of cross-device system changes as the activity changes and the individual devices might change role in the activity. The changes can either be adding, removing or substituting a device. Here we distinguish between a *horizontal* and a *vertical* change in the composition. When horizontal changes occur the base functionality of the multi-device system and its role in the activity does not change. Participants may add another device with identical capabilities of an existing, e.g. adding a larger display or another tablet that can interact with a specific component. A vertical change is when functionality is added or removed to the activity and system, e.g. adding a sketchpad or digitizer to a system that allow participants to embed sketches within a document. Understanding how the composition changes and what parts of the potential ecology (personal and shared devices) are active and the role they play are extremely important in supporting individual and collaborative activities and the various transitions that occur.

Mediation. Cross-surface systems mediate activities of people with certain goals and motives. We characterize the relationship between people to be either: *individual*, *social* or *collaborative*. Individual interaction with a cross-surface system is e.g. to distribute a web page across multiple personal devices [5]. Social interaction is where the interaction is influenced by the actions of others, but not directly affected. E.g. when posting images from personal devices to a public display. Finally collaborative interaction is when there is a common goal and interactions are directly affected by other users, e.g. collaborative editing of text on a shared display [2].

The way goals are realized mediated by the system we call the *instrumentality* of the interaction. Interaction can be *consumption* of digital content through reading, watching or listening; *communication* with other users through a digital medium either synchronously or asynchronously; *production and manipulation* of any kind of digital content, whether text, images videos etc.; *control* of the state of a system, whether digital (e.g. playback of a video) or physical (e.g. controlling the lighting of a room); *search and retrieval* of digital content; and finally *configuration* of a digital workspace (e.g. personalization or window placement). Each of the aspects of instrumentality can naturally not happen in isolation: search requires consumption, consumption requires control etc.

Some discussion

Returning to the 4C framework [8], here the focus is almost exclusively on collaborative control and consumption in their Netflix example; a screen is used for watching a movie and smartphone apps are used to control what is being watched. The composition of the example used in the 4C paper is simple and the capabilities/role of the artifacts are closely tied to the instrumentality of the system. In this case adding or removing a control device would be a horizontal change, whereas adding a device that allowed to review and discuss the movie would be a vertical addition to the composition. Note that this might already be possible within the system, but has to be actually used as part of the activity to be a part of the composition.

A recent study shows that while it is commonly assumed that the larger the surface the better for collaborative sense making, this may not be true for cross-surface situations where sense-making and search and retrieval are distributed across personal and shared surfaces [10]. This means that mediation and composition influence the affordances of interactive surfaces, which emphasises the need for being able to articulate them.

Not all combinations of mediation and composition are common-place today. Collaborative, cross-surface production and manipulation of digital content is rarely seen. This may point to a deeper challenge, namely that our tools for production and our understanding of those tools are deeply rooted in traditional personal computing.

Going forwards

Our own everyday confusion in articulating and working with cross-surface/multi-device perspectives on computing are motivating us to develop a conceptual framework that allow us to analyse and design novel systems within this space. This position paper attempt to do just that anchored in familiar theoretical territory. By basing our tentative vocabulary in activity theory we want to emphasise activities, and not individual use, as a primary focus. This is also an attempt to identify what's next in computing and in particular, how to address some of the fundamental (design) flaws of personal computing and move forward.

REFERENCES

1. Bødker, S., and Klokmose, C. N. Dynamics in artifact ecologies. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, ACM (2012), 448–457.
2. Bødker, S., Klokmose, C. N., Korn, M., and Polli, A. M. Participatory it in semi-public spaces. In *Proc. of the 8th Nordic Conference on Human-Computer Interaction*, ACM (2014), 765–774.
3. Bødker, S., and Korsgaard, Henrik, a. S.-S. J. 'a farmer, a place and at least 20 members': The development of artifact ecologies in volunteer-based communities. In *[Forthcoming] Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work &#38; Social Computing*, CSCW '16 (2016).
4. Müller, J., Alt, F., Michelis, D., and Schmidt, A. Requirements and design space for interactive public displays. In *Proceedings of the international conference on Multimedia*, ACM (2010), 1285–1294.
5. Nebeling, M., and Dey, A. K. Xdbrowser: User-defined cross-device web page designs. In *Proc. CHI 2016*, ACM (2016).
6. Rädle, R., Jetter, H.-C., Marquardt, N., Reiterer, H., and Rogers, Y. Huddlelamp: Spatially-aware mobile displays for ad-hoc around-the-table collaboration. In *Proc. of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ACM (2014), 45–54.
7. Rossitto, C., Bogdan, C., and Severinson-Eklundh, K. Understanding constellations of technologies in use in a collaborative nomadic setting. *Computer Supported Cooperative Work (CSCW) 23*, 2 (2014), 137–161.
8. Sørensen, H., Raptis, D., Kjeldskov, J., and Skov, M. B. The 4c framework: Principles of interaction in digital ecosystems. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, ACM (New York, NY, USA, 2014), 87–97.
9. Terrenghi, L., Quigley, A., and Dix, A. A taxonomy for and analysis of multi-person-display ecosystems. *Personal and Ubiquitous Computing 13*, 8 (2009), 583–598.
10. Zagermann, J., Pfeil, U., Radle, R., Jetter, H.-C., Klokmose, C., and Reiterer, H. When tablets meet tabletops: The effect of tabletop size on around-the-table collaboration with personal tablets. In *Proc. CHI 2016*, ACM (2016).

Towards Cross-Surface Content Sharing Between Mobile Devices and Large Displays in the Wild

Wolfgang Büschel
Interactive Media Lab
Technische Universität Dresden
bueschel@acm.org

Tom Horak
Interactive Media Lab
Technische Universität Dresden
horakt@acm.org

Ricardo Langner
Interactive Media Lab
Technische Universität Dresden
langner@acm.org

Raimund Dachselt
Interactive Media Lab
Technische Universität Dresden
dachselt@acm.org

Ulrich von Zadow
Interactive Media Lab
Technische Universität Dresden
uzadow@acm.org

Abstract

Large vertical displays are increasingly widespread, and content sharing between them and personal mobile devices is central to many usage scenarios. Research has already led to manifold interaction techniques. In most cases however, they do not lend themselves for realistic, in-the-wild usage. In this paper we present our research towards bridging the gap to real world usage. We address the issues of *awareness & connectivity* as well as *privacy*, which we believe to be two important aspects of BYOD (bring your own device) content sharing between public displays and mobile devices.

Author Keywords

Cross-device interaction; data transfer; privacy; large displays; mobile phones; proxemic interaction

ACM Classification Keywords

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

Introduction & Background

Hardware advances are making very large vertical displays more common in a variety of scenarios, e.g., as public displays. At the same time, personal devices such as mobile phones have become ubiquitous over the last years, as they allow people to conveniently manage their digital identi-

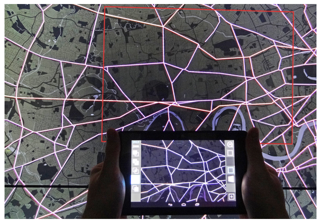


Figure 1: Tango tablet used in combination with a wall-sized display. The position is tracked using the internal sensors only.

ties and content. In combination these two device classes provide the advantages of both settings: among others, personalized interaction, private data storage, and on-demand data sharing. While these issues have been studied extensively in the past, there is a lack of work specifically addressing scenarios where users bring their own devices to share data with large public or semi-public displays outside of typical lab settings. In our research we address two aspects that are often still problematic in real world usage scenarios: (i) *awareness & connectivity* and (ii) *privacy*. In the following we will give a brief overview of the related work before exploring these two aspects in more detail.

For a general introduction to interaction with wall-sized displays, we refer to the overview by Müller et al. on public displays [7]. Additionally, Marquardt et al.'s work on Gradual Engagement [6] provides a design framework for integrating the relative positions of the devices involved in cross-device interaction. A related notion is Greenberg et al.'s Proxemic Interaction (e.g., [4]), in which interactions are based on spatial relationships between people and devices.

Much of the work on cross-device data transfer considers single-item transfer in close proximity. Rekimoto's Pick-and-Drop [8] is early work on cross-device data transfer using a pen as interaction device. More recently, Schmidt et al.'s PhoneTouch associates touches on a large display with a mobile phone by correlating the phone's motion sensor signals, covering both the technology [9] and numerous interaction techniques [10]. In SleeD [14], von Zadow et al. use an arm-worn device; transfer involves touching the large display with the hand the device is strapped on. Alt et al. [1] compare content creation for and exchange with public displays using multiple modalities, while Seifert et al. [12] introduce a number of interaction techniques that allow privately selecting the data to share before performing the actual transfer. In PointerPhone [11], Seifert et al. investigate the interactions possible when remote pointing is combined with interactions on the phone. Dachselt and Buchholz's Throw and Tilt [3] utilizes expressive gestures for data transfer, while Hassan et al.'s Chucking [5] is interesting because it also supports positioning of items on the large screen.

Awareness & Connectivity

A particular challenge for in-the-wild interaction between mobiles and public displays is the issue of *awareness & connectivity*. In this context, both technology and user interface have to be considered, which involves providing standardized protocols (location and services) as well as ui components and interactions.

Concerning the user interface, awareness is the knowledge of the user that there is a public display, that it supports interaction, and that it is ready to accept user input. Known approaches such as Marquardt et al.'s Gradual Engagement Pattern [6] can support this. We believe that in the future a background service on the mobile device could continuously check for available connections in the environment and give appropriate feedback, e.g., status bar messages or vibration to get the user's attention.

Technologically, awareness means that both the mobile can react to public displays, as described above, and the large display can react to the mobile device. However, there are different levels of sensing to this. In fact, they form a whole spectrum between only knowing that a device is in the vicinity and full six DoF tracking. These technical capabilities influence the availability of the following interaction styles:

- In the case where no additional tracking data is available, the mobile device can only be used similar to a TV remote. Data transfer by techniques such as

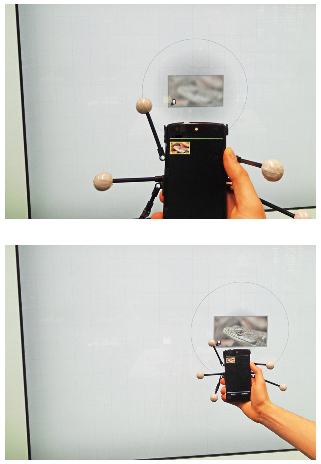


Figure 2: Top: The preview image on the public display is blurred to preserve privacy. Bottom: Blurring can be distance aware. When the user stands near the large display and shoulder surfing is harder, less blurring is applied.

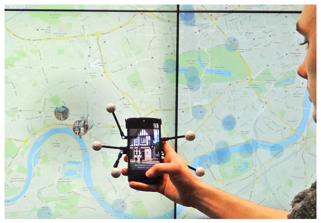


Figure 3: By default, only the position of pictures on a map is shown. Only when directly pointing at them, their preview is revealed.

swiping is possible but lacks positional information. Alternatively, drag gestures on the phone can (relatively) control the movement of a pointer on the large display.

- If the large display allows sensing of contact positions (e.g. it is touch capable or via marker tracking of the phone), precise transfer to the touch position becomes possible when in touching distance (e.g., [9, 14]).
- Relative motion tracking of the mobile device, e.g., using the internal IMU, enables device gestures such as throwing [3].
- Finally, with full 6 DoF tracking available, pointing is available and allows for precisely targeted transfer of digital objects.

Up to now, tracking of mobile devices in front of large vertical displays usually involves instrumentation of the room and in many cases also the device, using marker based tracking systems or depth cameras (e.g., [13]). Lately, devices such as Google's Tango[1] allow non-instrumented, visual motion tracking. We tested Tango in a typical interaction scenario in front of a wall-sized interactive display (Figure 1) and found the tracking precision sufficient for most application cases. Drift, however, can be a problem, especially when the display content changes rapidly.

For meaningful interaction between the devices, not only files transfer has to be considered but also specific, intuitive user interfaces. General file transfer applications, e.g., a file browser, would be applicable to a wide range of situations. However, we believe that carefully crafted interfaces that take the context (display type, application scenario, ...) into consideration could provide a richer experience. Additionally, such interfaces could also integrate not only content sharing but other uses for mobile devices, such as personal tangible magic lenses, as well. Therefore, in our investigations we explore a streaming-based thin-client architecture. The application running on the large display streams user interfaces to the mobile device and collects user input. Our current prototype renders the UI into an h.264 video stream using the Python-based libavg[2] framework. The stream is then decoded and displayed on the mobile device by a small, native application. User input, i.e., touch and motion data, is streamed back via OSC protocol. This allows applications that use a simple and thin client to provide arbitrary, application-specific user interfaces. Depending on the precise details of the usage scenario (e.g., number of clients), we think that this solution could serve as a basis for various settings, because the bandwidth of available network connections increases continuously and todays mobile devices are equipped with efficient decoders.

Privacy

Privacy is a general concern in many multi-user application scenarios. In particular, we believe that this is a major factor to be considered in content sharing applications between public and private devices. There has already been work regarding the presentation of private data on public displays and specific interactions, such as on limiting shoulder surfing [2] and password input [9], respectively. Given that users often store very personal pictures on their phones, they might be hesitant to let others see them unfiltered. To counteract privacy concerns in such cases, we currently explore three different interface strategies: (i) limit what data is displayed at all, (ii) hide information through blurring and other image-based methods, and (iii) show abstract representations of data items, e.g., by only showing metadata.

[1] https://www.google.com/atap/project-tango/

[2] http://libavg.de/

In our investigations, we use a simple two-step process for data sharing activities: First, instead of sharing and displaying all documents at once, the user manually selects them on his or her device, thus filtering them on a smaller, more private display, similarly to [12]. Secondly, when transferring to the public display, we show a heavily blurred version of the content (Figure 2). This allows the user, who knows his or her photos, to cancel the transfer if an undesired picture is shown, while hiding details from the public. For use cases such as geotagged photos on a map, we transfer metadata of items first, allowing the large display to show preview circles at corresponding positions. The photos themselves are not revealed until the pointing cursor is over their position, preserving privacy (Figure 3).

Besides the issue of uploading content, *downloading* from the public display can also lead to privacy concerns: Imagine a public information terminal in a medical center with info flyers on different medical issues or a digital notice board with anonymized publicly available exam results. These are situations in which a user might not want others to see which digital objects he or she copies to their personal device. To this end, we propose the concept of *blind pointing*: During pointing/selection, no visual representation of a selection cursor is shown on the public display. Instead, the user only gets vibrotactile feedback. We believe that proprioception could allow users to reliably select items on a large display solely supported by such unspecific feedback in combination with small, inconspicuous pointing gestures.

Conclusion

In this position paper, we outlined our on-going explorations on content sharing between large displays and mobile devices, focusing on the two important aspects *awareness & connectivity* as well as *privacy*. In the context of BYOD and to consider different device platforms, maximize compatibility, and ease the process of device connections, we explore the usage of non-instrumented devices (i.e., without additional markers) such as Tango and deploy a streaming-solution in which the display application sends the user interface (output) and collects user input. Furthermore, we developed early concepts and prototype implementations addressing the issue of privacy, such as abstracting visual user feedback on large displays or using *blind pointing*.

We believe that, together with various other investigations in this field, our research helps the community to determine underlying principles and enable the development of seamless content sharing between large displays and personal mobile devices in the wild.

References

[1] Florian Alt, Alireza Sahami Shirazi, Thomas Kubitza, and Albrecht Schmidt. 2013. Interaction Techniques for Creating and Exchanging Content with Public Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1709–1718. DOI: http://dx.doi.org/10.1145/2470654.2466226

[2] Frederik Brudy, David Ledo, Saul Greenberg, and Andreas Butz. 2014. Is Anyone Looking? Mitigating Shoulder Surfing on Public Displays Through Awareness and Protection. In *Proceedings of The International Symposium on Pervasive Displays (PerDis '14)*. ACM, New York, NY, USA, Article 1, 6 pages. DOI: http://dx.doi.org/10.1145/2611009.2611028

[3] Raimund Dachselt and Robert Buchholz. 2008. Throw and Tilt – Seamless Interaction across Devices Using Mobile Phone Gestures. In *Proc. Workshop MEIS '08*. German Informatics Society (GI), 272–278.

[4] Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob

Diaz-Marino, and Miaosen Wang. 2011. Proxemic interactions: the new ubicomp? *Interactions* 18, 1 (Jan. 2011), 42–50. DOI:http://dx.doi.org/10.1145/1897239.1897250

[5] Nabeel Hassan, Md. Mahfuzur Rahman, Pourang Irani, and Peter Graham. 2009. Chucking: A One-Handed Document Sharing Technique. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction (INTERACT '09)*. Springer-Verlag, Berlin, Heidelberg, 264–278. DOI:http://dx.doi.org/10.1007/978-3-642-03658-3_33

[6] Nicolai Marquardt, Till Ballendat, Sebastian Boring, Saul Greenberg, and Ken Hinckley. 2012. Gradual Engagement: Facilitating Information Exchange Between Digital Devices As a Function of Proximity. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces (ITS '12)*. ACM, New York, NY, USA, 31–40. DOI:http://dx.doi.org/10.1145/2396636.2396642

[7] Jörg Müller, Florian Alt, Daniel Michelis, and Albrecht Schmidt. 2010. Requirements and Design Space for Interactive Public Displays. In *Proceedings of the International Conference on Multimedia (MM '10)*. ACM, New York, NY, USA, 1285–1294. DOI:http://dx.doi.org/10.1145/1873951.1874203

[8] Jun Rekimoto. 1997. Pick-and-drop: A Direct Manipulation Technique for Multiple Computer Environments. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology (UIST '97)*. ACM, New York, NY, USA, 31–39. DOI:http://dx.doi.org/10.1145/263407.263505

[9] Dominik Schmidt, Fadi Chehimi, Enrico Rukzio, and Hans Gellersen. 2010. PhoneTouch: A Technique for Direct Phone Interaction on Surfaces. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 13–16. DOI:http://dx.doi.org/10.1145/1866029.1866034

[10] Dominik Schmidt, Julian Seifert, Enrico Rukzio, and Hans Gellersen. 2012. A Cross-device Interaction Style for Mobiles and Surfaces. In *Proceedings of the Designing Interactive Systems Conference (DIS '12)*. ACM, New York, NY, USA, 318–327. DOI:http://dx.doi.org/10.1145/2317956.2318005

[11] Julian Seifert, Andreas Bayer, and Enrico Rukzio. 2013. PointerPhone: Using Mobile Phones for Direct Pointing Interactions with Remote Displays. In *Human-Computer Interaction – INTERACT 2013*, Paula Kotzé, Gary Marsden, Gitte Lindgaard, Janet Wesson, and Marco Winckler (Eds.). Lecture Notes in Computer Science, Vol. 8119. Springer Berlin Heidelberg, 18–35. DOI:http://dx.doi.org/10.1007/978-3-642-40477-1_2

[12] Julian Seifert, David Dobbelstein, Dominik Schmidt, Paul Holleis, and Enrico Rukzio. 2014. From the private into the public: privacy-respecting mobile interaction techniques for sharing data on surfaces. *Personal and Ubiquitous Computing* 18, 4 (2014), 1013–1026. DOI:http://dx.doi.org/10.1007/s00779-013-0667-x

[13] Martin Spindler, Wolfgang Büschel, Charlotte Winkler, and Raimund Dachselt. 2014. Tangible Displays for the Masses: Spatial Interaction with Handheld Displays by Using Consumer Depth Cameras. *Personal Ubiquitous Comput.* 18, 5 (June 2014), 1213–1225. DOI:http://dx.doi.org/10.1007/s00779-013-0730-7

[14] Ulrich von Zadow, Wolfgang Büschel, Ricardo Langner, and Raimund Dachselt. 2014. SleeD: Using a Sleeve Display to Interact with Touch-sensitive Display Walls. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces (ITS '14)*. ACM, New York, NY, USA, 129–138. DOI:http://dx.doi.org/10.1145/2669485.2669507