XDBrowser: User-Defined Cross-Device Web Page Designs

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

ABSTRACT
There is a significant gap in the body of research on cross-device interfaces. Research has largely focused on enabling them technically, but when and how users want to use cross-device interfaces is not well understood. This paper presents an exploratory user study with *XDBrowser*, a cross-device web browser we are developing to enable non-technical users to adapt existing single-device web interfaces for cross-device use while viewing them in the browser. We demonstrate that an end-user customization tool like XDBrowser is a powerful means to conduct user-driven elicitation studies useful for understanding user preferences and design requirements for cross-device interfaces. Our study with 15 participants elicited 144 desirable multi-device designs for five popular web interfaces when using two mobile devices in parallel. We describe the design space in this context, the usage scenarios targeted by users, the strategies used for designing cross-device interfaces, and seven concrete mobile multi-device design patterns that emerged. We discuss the method, compare the cross-device interfaces from our users and those defined by developers in prior work, and establish new requirements from observed user behavior. In particular, we identify the need to easily switch between different interface distributions depending on the task and to have more fine-grained control over synchronization.

Author Keywords
Cross-Device Web Design; End-User Customization Study.

INTRODUCTION

The current proliferation of new devices ranging from smartwatches and phones, to tablets, to interactive tabletops and walls has prompted researchers to investigate combined use of multiple devices. We identify three main research streams.

First, a number of studies have been conducted on multi-device workflows [6, 17, 28, 30] and specifically cross-device web use [18, 20, 19, 34]. These studies clearly identified a need to support sequential and parallel use of multiple devices to more easily share tasks and interfaces between them. However, they were insufficient for producing concrete guidelines on how to adapt existing interfaces for cross-device use.

Second, many different cross-device interaction techniques [3, 4, 29, 31, 32] have been explored. While they are promising for improving cross-device interactions, most techniques are limited to a particular set of applications or combination of devices, and are generally not available in current interfaces.

A third stream of research has investigated new cross-device design tools [5, 7, 8, 9, 11, 12, 13, 14, 25, 36]. The vast majority of these tools, however, are targeted at developers, allowing them to distribute interfaces between devices programmatically or using special authoring tools. While a number of possible cross-device interfaces were showcased to illustrate technical support, whether or not these address actual user needs and are desirable by end-users was not explored.

As part of the *XDBrowser* project, we are investigating how existing web browsers can be extended to support parallel usage of multiple devices as motivated by previous studies. This paper presents a prototype of XDBrowser that enables non-technical users to adapt existing single-device interfaces for cross-device use. We use XDBrowser primarily as a tool for conducting a user-driven elicitation study on cross-device interfaces. Asking users to customize and try out alternative cross-device designs during active use, enabled us to observe user behavior both when designing and when using cross-device interfaces. The main contribution of this paper is therefore our study, which aims to build a better understanding of when and how users want to use cross-device interfaces.

Our work is inspired by the early cross-device paper prototyping study by Hutchings and Pierce [15]. In their study, users were asked to demonstrate how they would want interfaces to be distributed between devices when used in different multi-device environments. Our own research also started out using paper prototyping to experiment with different cross-device interfaces and considered designing for many different cross-device settings. However, we quickly realized that a crucial element for users is to experience the interaction as closely as possible to the final interface. We wanted to avoid the three main limitations we found with their study design.

First, paper prototyping has strong limitations when testing response times and when comparing small variations of an interface. The latter became particularly evident when designing for a combination of handheld mobile devices, where interfaces are small and preferences depend on the details of the interaction, making the paper prototyping method unsuitable.

Second, they studied three multi-device settings that resulted in a rather high-level analysis of user-defined distributions, mostly showing that users wanted to distribute interfaces in all conditions, but providing little detail on how. We focus on one likely cross-device setting using two mobile devices in parallel to study this in detail.

Third, their study asked users to customize each interface for one particular task. Our study instead is exploratory in nature, asking users to customize for all the tasks that they would like to perform with the cross-device version of the interface and use the think-aloud protocol to describe how their interactions and tasks would benefit from cross-device use.

In our study, 15 participants created 144 desirable multi-device designs for five popular types of interfaces when using two mobile devices in parallel. We report on usage scenarios suggested by users, the design strategies employed by users when designing for this cross-device setting, and seven mobile multi-device design patterns underlying the user-defined interfaces. Based on user feedback, we analyze both ease of design and ease of use of different cross-device interfaces and identify the reasons why some cross-device interfaces were more difficult to create than others, and how to better support those in future cross-device design tools. We close with a discussion of our method and how the user-defined designs from our study compared to developer-defined designs from prior work, as well as suggesting ways of using our results in future research.

RELATED WORK

We build on prior work on *multi-device workflows*, *cross-device systems and toolkits*, and *end-user customization*.

Multi-Device Workflows

Early studies by Oulasvirta and Sumari [28] and Dearman and Pierce [6] in the context of mobile information work identified a strong need for seamless multi-device interaction because many activities spanned multiple devices. However, multi-device workflows had to be carefully planned as they were commonly hindered by two facts. First, information and actions are tied to the device rather than the user. Second, devices are not aware of their role in the user's larger task.

In addition to the more serial workflows identified by these works, a recent study by Santosa and Wigdor [30] found new patterns of parallel device use. Despite widespread adoption of cloud services for multi-device synchronization, the authors still report a lack of support for parallelism in cross-device interactions. Jokela *et al.* [17] distinguish between resource lending, related parallel use, and unrelated parallel use. They identify three challenges users commonly face when using multiple devices in parallel: deciding which devices to use, accessing content between devices, and keeping devices updated and charged. Display size, input capabilities, and level of multi-tasking support are the main factors in choosing which device to use in a specific situation.

With a particular focus on cross-device web use, Kane *et al.* [18] analyzed web-based usage data across devices and found that often the same sites were visited in desktop and mobile settings. Follow-up studies by Karlson *et al.* [20] using log analysis identified temporal usage patterns with information workers as well as strong device preferences for certain tasks, such as preferring the smartphone for keeping track of emails and device switching rarely occurring within a given task. A second screenshot-based diary study [19] found that smartphone tasks were often suspended until they could be completed on the PC. They identified and discussed major task completion barriers such as missing functionality, input/output challenges, complex tasks and cost/benefit choices.

Together, these studies identify a lack of support for synchronizing relevant task state across devices to make it easy to transition a task between devices. They also highlight that, in order for users to benefit from using multiple devices to complete a task, they need to be able to decompose the task into meaningful subtasks that they can then distribute and perform using multiple devices. Our work adds to these studies by detailing when and how users want common web browsing tasks to be distributed between multiple devices and the cross-device interactions demanded by users.

To enable our study, we first studied the technical support available in existing cross-device systems and tools. Using this knowledge, we developed our prototype of XDBrowser and used it as a tool to study user-defined cross-device interfaces.

Cross-Device Systems and Toolkits

Johanson *et al.* [16] introduced the idea of *multibrowsing* and developed special software enabling users to send web pages to, or open them from, other devices. A lot of the research since then has focused on improving the technical aspects of multibrowsing. For example, Ghiani *et al.* [9] developed push-and-pull techniques that are not limited to transferring URLs between devices, but provide advanced support for HTML DOM capture and JavaScript state persistence. This enables full migration between devices and even multiple users, but comes at the cost of additional infrastructure and requires its own web proxy-based migration environment [10].

Recently, a number of cross-device systems and toolkits were proposed. We focus our review on those that informed the design of XDBrowser's distribution tools the most.

Panelrama [36] is a web-based toolkit with support for interface division and automated distribution between devices based on developer-specified groups of web page elements, so-called panels, and their suitability for distribution. Panels need to be specified in HTML and panel state captured and synchronized in JavaScript using shared variables.

With the Tandem Browsing Toolkit [12], developers can create multi-display web sites consisting of multiple parts which can be viewed on several devices in parallel. Geared towards web applications that have stages, *e.g.,* a shopping web site that consists of a login, item selection and checkout stage, the toolkit synchronizes transitions from one stage to another across multiple devices through a finite state machine.

Frosini and Paterno [8] propose a flexible framework based on the idea of dynamically migrating engines that provides developers with an API for interface distribution and/or migration between dynamic sets of devices. Distribution can be dynamically controlled based on the types of a specific device or group of devices and the roles associated to a specific user or groups of users, which can be switched at any time.

XDStudio [25] is the first distributed interface builder following a long line of previous multi-device authoring tools such as Damask [22], Gummy [24] and D-Macs [23]. The specific focus of XDStudio is on enabling interactive development of cross-device interfaces by adding cross-device capabilities to the authoring environment itself. Distributed interfaces can be designed by simulating target devices, by directly involving them *via* on-device authoring, or both. At its core is the notion of distribution profiles that enable reuse of existing interface distributions and their adaptation to new cross-device settings not originally designed for at run-time.

While this past work is very motivating to the goal of enabling more seamless cross-device interaction, the techniques have so far been limited to developers. Our goal with XDBrowser is to adapt and integrate the cross-device support that has already been implemented and demonstrated as separate systems targeted at developers. We want this support to be available directly in a web browser without requiring developers to modify interfaces to enable cross-device use and without requiring users to install and use special migration infrastructure and browsing clients. To avoid these constraints, our prototype necessarily does not offer a full range of support for cross-device interfaces. With our prototype, however, non-technical users are able to browse existing single-device interfaces and customize them for serial or parallel usage of multiple devices *via* built-in visual tools. Most importantly, browser view state and input can be flexibly controlled and synchronized between multiple devices. This was crucial for letting users experiment with different kinds of cross-device interfaces and eliciting both technical and design requirements from the usage of XDBrowser and observed user behavior.

End-User Customization

End-user customization techniques commonly aim to empower non-technical users with simple and easy-to-use tools for adapting existing interfaces to their requirements. Our prototype of XDBrowser adds to the research in this area and shares commonalities with the following systems.

PageTailor [2] provides end-users on PDAs with a direct manipulation interface and simple controls for hiding, resizing and repositioning elements in a web page. Using PageTailor, web pages can be customized over multiple browsing sessions and are automatically reconfigured when a page is loaded.

The Highlight [27] system takes the inverse approach in that, while browsing a web page on the desktop PC, end-users can demonstrate interactions central to the application for browsing on mobiles. The interaction sequences and corresponding web page elements are learned and used to generate a smaller, mobile-friendly version of the original page. Highlight uses a separate authoring tool that provides a storyboard view and a preview to browse and refine the mobile versions.

The PlayByPlay system [35] shares browsing sessions in the form of instant messages that can be executed in another browser using the CoScripter technology [21]. This is particularly powerful for collaborative web browsing with multiple users and can also be used for synchronizing web browsing actions across multiple desktop and mobile devices.

CrowdAdapt [26] builds on crowdsourcing and introduces new adaptation operations to support, not only desktop-to-mobile adaptation, but a much wider range of devices including large screens. With CrowdAdapt, users can interactively adapt the web page layout similar to PageTailor and share their customizations so that others browsing with similar devices can immediately benefit from user-contributed layouts.

Our XDBrowser prototype extends these techniques so that existing interfaces can be partitioned, copied and moved between devices as they are viewed in the browser. Compared to previous developer-oriented techniques [10, 14, 26, 25, 36] which are all based on HTML DOM element selection, XDBrowser does not require the user to understand the page's internal structure and composition of the layout. In fact, it works independent of the HTML DOM and easily caters to cases where page elements are specified at different levels of the DOM hierarchy. Inspired by WinCuts [33], we implemented a rectangular selection tool allowing users to frame any part of the web page that they want to share between devices. In addition, multi-select is supported so that different parts of the same page can be shared. As a result, web interfaces can be cut, copied and moved between devices, making it as easy as constructing paper prototypes for multi-device environments [1, 15], but allowing this directly in the browser *via* visual tools that are suitable for non-technical users.

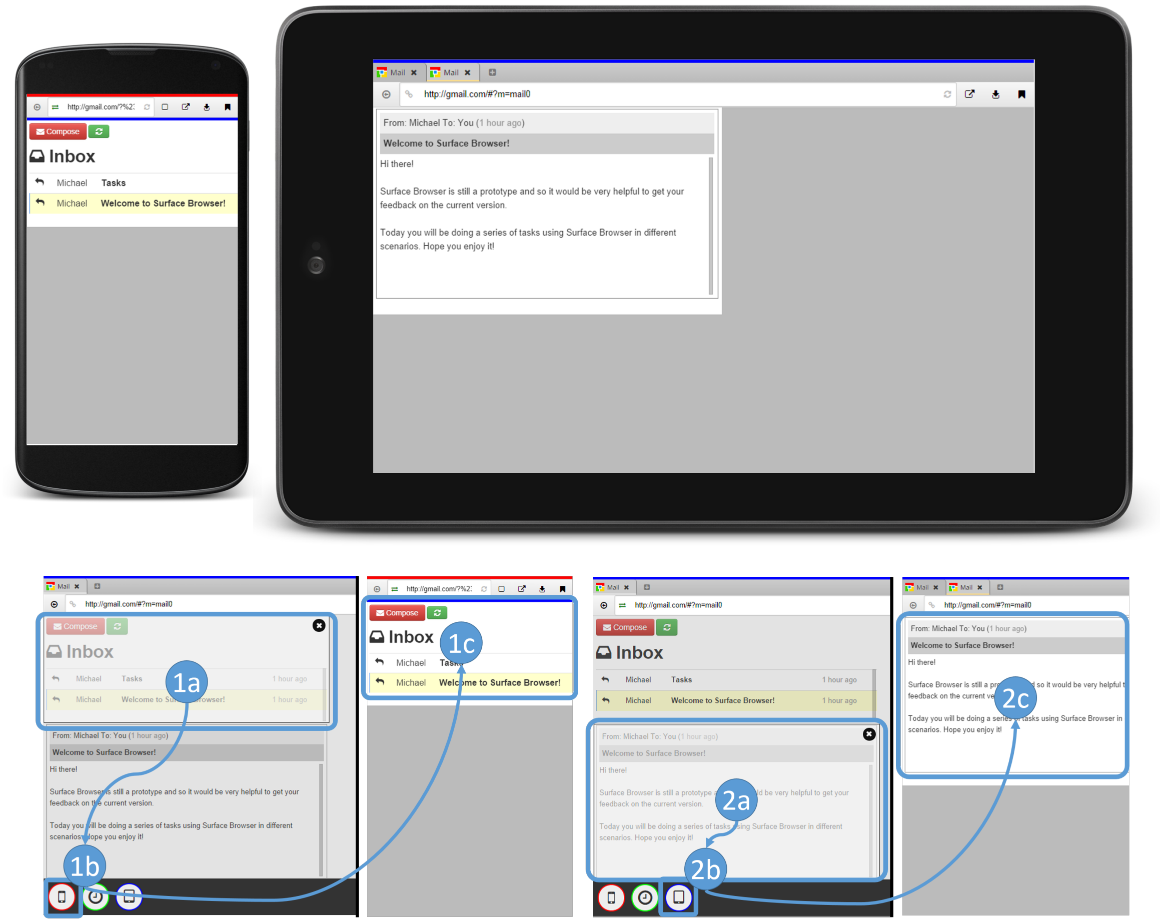
CROSS-DEVICE INTERFACES WITH XDBROWSER

Our larger goal with XDBrowser is to support more seamless cross-device use when browsing on multiple devices in parallel. We developed a prototype to better understand user needs and study when and how users want to use cross-device interfaces. The main idea of our study was to ask users to adapt popular single-device interfaces for cross-device use as they view and interact with them using the browser.

XDBrowser supports three common functions for multi-device interface distribution adapted from existing cross-device systems and tools: *(1)* pushing an entire page to a device, *(2)* pushing user selected parts of a page, and *(3)* pulling a page from another device. When pushing or pulling pages using these functions, XDBrowser opens the URL in a new tab on the target device and checks whether the user selected regions within the page. If that is the case, the page will be split into the corresponding regions.

To illustrate the three functions and the resulting actions in XDBrowser, let us use an example. Figure 1 shows a scenario in which XDBrowser was used to combine a smartphone and a tablet to work around individual device limitations and maximize screen usage by distributing parts of a mail interface. The phone is used for viewing the inbox and selecting mails (Step *1)*), while the tablet is used as the primary device for reading mails (Step *2)*). Clicking a message on the phone loads the respective mail on the tablet. This is an example of the *overview+detail* design pattern commonly employed by users to view details on the larger device and keep an overview on the smaller device.

Users can choose which tabs or shared page they want to keep in sync. When synchronization is activated for a page,



Michael Tasks 1 hour ago
Michael Welcome to Surface Browser! 1 hour ago

From: Michael To: You (1 hour ago)
Welcome to Surface Browser!
Hi there!
Surface Browser is still a prototype and so it would be very helpful to get your feedback on the current version.
Today you will be doing a series of tasks using Surface Browser in different scenarios. Hope you enjoy it!

Figure 1. Mail interface customized using XDBrowser to view the inbox on a smartphone (Steps *1a)* select inbox on tablet; *1b)* push inbox from tablet to phone; *1c)* inbox on phone) and read selected mails on a tablet (Steps *2a)* select message on tablet; *2b)* push message to tablet; *2c)* message on tablet)

XDBrowser will broadcast changes of the viewport and input provided by the user to connected devices. All tabs that have the same page loaded will be updated with the new view state and input. XDBrowser automatically activates synchronization for pages pushed to, or pulled from, another device. Users can toggle synchronization at a per-tab level, or can disable synchronization for all tabs at once to operate the device completely independently from other connected devices.

XDBrowser supports four synchronization options that can be toggled to achieve different cross-device interactions:

1. *Click sync* captures click events at the document level, therefore covering the entire page, and replays them in all synced tabs by locating the target element and triggering the associated click handler. To prevent multiple form data submissions, clicks on submit buttons are not replayed.

2. *Scroll sync* monitors scrolling along both axes and maintains the scroll state by transferring and setting the scroll position in proportion to the viewport size on synchronized devices.

3. *Input sync* captures keyboard, mouse and touch input on any form input field (text fields, checkboxes and radio buttons, comboboxes and multi-selects) part of the page. For each field, the current value is serialized and restored in synced browser tabs opened on the same and connected devices.

4. *URL sync* tracks changes to the URL loaded into a tab including changes to the fragment or hash part. The hash is not only used by web browsers to jump to anchor links, but also by many modern web applications, such as GMail, Google Maps, or YouTube, to store page-internal state, such as the selected mail folder or message, the location and zoom level of the map, or the video and timeline index. When a change of the URL is detected, XDBrowser updates all synchronized tabs with that URL so that page-internal state is transferred and updated on all connected devices.

In combination, these four synchronization options enable XDBrowser to fully restore the browser's view state and input when a page is shared between devices. Synchronization keeps the interactions in one tab or shared page synchronized with all other tabs on the same device or on connected devices that have the same page opened. Therefore, with all synchronization options enabled, clicking buttons, scrolling the page, entering form data, and jumping to anchors or modifying the URL on one device will perform the same action on all connected devices. Disabling synchronization means that the interactions on the current device will have no effect on the other devices. Synchronization can be toggled at any time so that it is possible to decouple devices to operate them independently and reintegrate and resynchronize them on demand. Note that this all still works when pages are cut into regions and even if multiple cuts of the same page are shared with other devices since XDBrowser only restricts the viewport according to the user's selection. This essentially hides, but does not extract or remove, HTML DOM elements and therefore does not run the risk of breaking web pages. While these features may not be sufficient to support all the cross-device interfaces that could possibly be created with previous systems and toolkits, our study reported later shows that they do make XDBrowser a powerful tool for users to explore and experiment with the kinds of cross-device interfaces they find useful and usable. We developed the synchronization options prior to the study and believe they are generic and could apply to a wider range of applications than the ones we studied.

XDBROWSER IMPLEMENTATION

XDBrowser is implemented as a fullscreen browser interface running within the OS's default web browser. The fact that only native browser features based on HTML5 and CSS3 are used enables XDBrowser to run on any web-enabled device providing modern browser support, with *no need for modification or special software*, making it a lightweight solution that can readily be adopted. The browser interface is responsive to the device in use, enabling its interface to adapt to a wide range of mobile devices including phones, tablets, and laptops.

XDBrowser uses a client-server architecture. The two main client-side components are: (1) *browser windows* containing the browser's navigation controls and wrapping the actual web pages in iframes and (2) *selection boxes* for cutting browser windows and transferring parts of a page. When distributing pages, first the iframe wrapping the entire page is cloned. For each selection box, the cloned iframe is then wrapped again in a new container sized to fit the selection box and translated to match the scroll position. The main components are implemented using the jQuery Widget Factory. Modernizr and CSS3 media queries are used for XDBrowser's responsive design. Depending on the device's screen dimensions and available touch support, either jQuery UI or Hammer.js are used for draggable and resizable behaviors. The server is based on Node.js. Socket.IO is used for realtime bidirectional event-based communication between all connected devices.

While sufficient for our study, this implementation has some limitations. Our approach works best with pages served locally or stored offline. Many top sites forbid iframe embedding and browsers prevent cross-site scripting, but a proxy server fixes this [9]. For sites that maintain a session, the user needs to login on each device, but a remote-control architecture [27] or shared virtual browser such as PhantomJS resolves this [14].

STUDY DESIGN

Our end-user customization study with XDBrowser comprised three parts. First, a questionnaire collected data on participants' use of devices and interfaces, their experience using touch as input and designing interfaces, and demographic information. They were then introduced to the core features of XDBrowser, explaining the push-and-pull functions and means for synchronization by showing examples of multi-device distributions. This part of the study was used both for training and priming of participants regarding possible cross-device interfaces.

Second, participants were asked to perform five customization tasks for the interfaces shown in Figure 2. Task order was randomized and counterbalanced between participants. For each task, participants used XDBrowser on two mobile devices of different size, a Nexus 4 phone and a Nexus 7 tablet. Using the features of XDBrowser and the think-aloud protocol, they were asked to perform each customization task in three steps: *(1)* demonstrate a task they commonly perform with the interface, *(2)* create desirable interface distributions to perform the task using the two devices in parallel, and *(3)* perform the task using the custom cross-device interface. Immediately after each customization task, participants were given a short questionnaire where they rated three statements related to the task: *"I would enjoy operating the browser in this manner"*, *"It was easy to do the task"*, and *"The available features were effective for this task"*.

At the end of the study, users filled in a detailed questionnaire rating various features of XDBrowser. We recruited 15 participants (10 male, 5 female, median age 28) with high experience using touch as input (median 6 using 7-point Likert scale) and intermediate UI design skills (median 5). All participants indicated high familiarity with the five web sites that our versions resembled, and owned and used multiple devices on a regular basis. There was no time limit in the study and participants were encouraged to try out as many different designs as they wanted. The study took roughly one hour per participant.

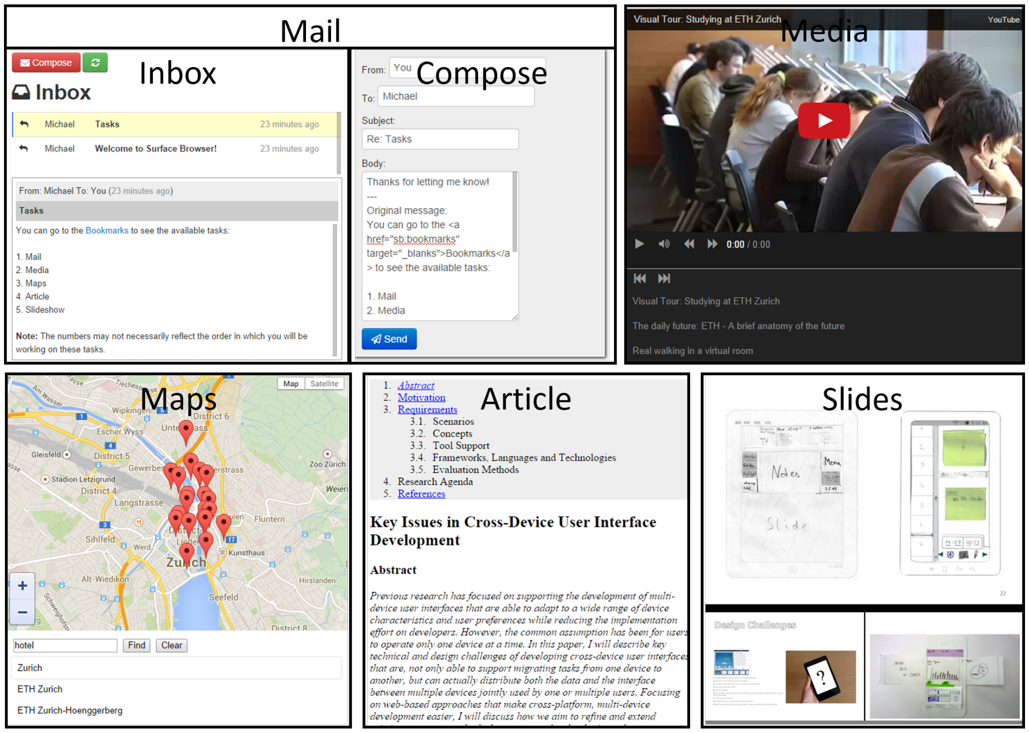
Below we describe the five interfaces used for customization: mail client, media player, maps viewer, article reader and slideshow presenter (Figure 2). These interfaces were commonly reported in the cross-device literature [4, 8, 11, 25, 36]. We believed that this selection would give us a variety of interfaces with different requirements and best allow us to compare the cross-device interfaces designed by our users to those designed by developers in previous work. All interfaces were based on real web sites; however, these are highly dynamic and personalized. Our versions ensured that the same content is shown to all participants. Rather than five specific existing sites, we wanted to test representative but simplified versions of them to focus users on core page elements.

Mail Interface

Similar to many email clients, the *mail* interface is composed of two pages: an inbox and a compose page. The inbox lists mails and shows the content of a mail selected from the list. The compose page for writing mails is opened when the compose button or the reply icon next to each mail is pressed. With this interface, we were interested in understanding how users would deal with synchronized pages, showing details of selected mails, and text input in a multi-device scenario.

Media Interface

Inspired from and implemented using YouTube, the *media* interface contains a video player and a playlist with controls for pausing/resuming playback, muting and unmuting, seeking



Inbox

Michael	Tasks	23 minutes ago
Michael	Welcome to Surface Browser!	23 minutes ago

From: Michael To: You (23 minutes ago)

Tasks

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

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

Note: The numbers may not necessarily reflect the order in which you will be working on these tasks.

From: You

To: Michael

Subject: Re: Tasks

Body:

Thanks for letting me know!

Original message:
You can go to the <a href="sb:bookmarks" target="_blanks">Bookmarks</a> to see the available tasks:

1. Mail
2. Media

Visual Tour: Studying at ETH Zurich

0:00 / 0:00

Visual Tour: Studying at ETH Zurich

The daily future: ETH - A brief anatomy of the future

Real walking in a virtual room

hotel

Zurich

ETH Zurich

ETH Zurich-Hoenggerberg

1. *Abstract*
2. Motivation
3. Requirements
   3.1. Scenarios
   3.2. Concepts
   3.3. Tool Support
   3.4. Frameworks, Languages and Technologies
   3.5. Evaluation Methods
4. Research Agenda
5. References

Key Issues in Cross-Device User Interface Development

Abstract

Previous research has focused on supporting the development of multi-device user interfaces that are able to adapt to a wide range of device characteristics and user preferences while reducing the implementation effort on developers. However, the common assumption has been for users to operate only one device at a time. In this paper, I will describe key technical and design challenges of developing cross-device user interfaces that are, not only able to support migrating tasks from one device to another, but can actually distribute both the data and the interface between multiple devices jointly used by one or multiple users. Focusing on web-based approaches that make cross-platform, multi-device development easier, I will discuss how we aim to refine and extend

Figure 2. The five customization tasks used for the study involved a mail client, a media player, a maps browser, an article reader and a slides viewer.

through the video, as well as rotating through the playlist. The playlist was restricted to three selected videos. This is an interesting interface for our study as it raises the question of which device(s) should play video and/or audio as well as issues of synchronizing media and controlling playback.

Maps Interface

Based on Google Maps, the *maps* interface embeds a map that can be panned and zoomed using gestures. It also has controls for switching between the map, terrain and satellite views as well as increasing/decreasing the zoom level. Located below the map is an input field for text search based on the Google Places service. For each place found in response to a query (*e.g.,* "hotel"), a marker will be added to the map. Markers can be clicked to show additional information. The *maps* interface also lists three places that center and zoom the map on the target location when selected. This interface is interesting due to the many possible configurations of the map across devices and in terms of what should and should not be synchronized.

Article Interface

Similar to many online newspapers, blogs and wikis, the *article* interface shows a table of contents with links to the respective sections in the text, an excerpt of the actual text, and a list of the references for citations in the text. This interface is interesting for exploring scrolling and navigation within larger pages and how the need for these may be reduced with cross-device use.

Slides Interface

Finally, to cover tasks common to gallery browsing and slideshow presentations, we included the *slides* interface. It consists of the current slide that can be clicked to go to the next slide, as well as thumbnail previews of the previous and next slides. Each preview can be clicked to set it as the current slide. In addition, the interface contained a textarea for notes. This interface is interesting as all page elements control and provide a different view on the state of the slideshow.

RESULTS

The main results of our study comprise the usage scenarios suggested by participants for parallel device use, the design strategies employed by participants when customizing each interface, and the design patterns emerging from the different designs created by participants. They also include participant feedback in terms of whether they would enjoy using XD-Browser, its ease of use when customizing each interface, and the effectiveness of its distribution and synchronization tools.

Cross-Device Interfaces and Usage Scenarios

In total, participants created 144 desirable designs with XD-Browser for the five interfaces using the two mobile devices. With an average of 9.6 different designs per participant and 1.92 per participant and task, participants could almost always think of at least two different ways of customizing and using each interface in the cross-device setting. We discuss the highlights of each customization task below, showing the variety of cross-device interfaces created using XDBrowser.

Mail Interface

For the *mail* interface, participants commonly customized for two primary tasks: reading mails and writing mails. A popular design for the first task was to use the phone for checking the inbox and selecting messages that participants then read on the tablet (Figure 3(a)). However, when they considered customizing for the second main task, writing mails, they

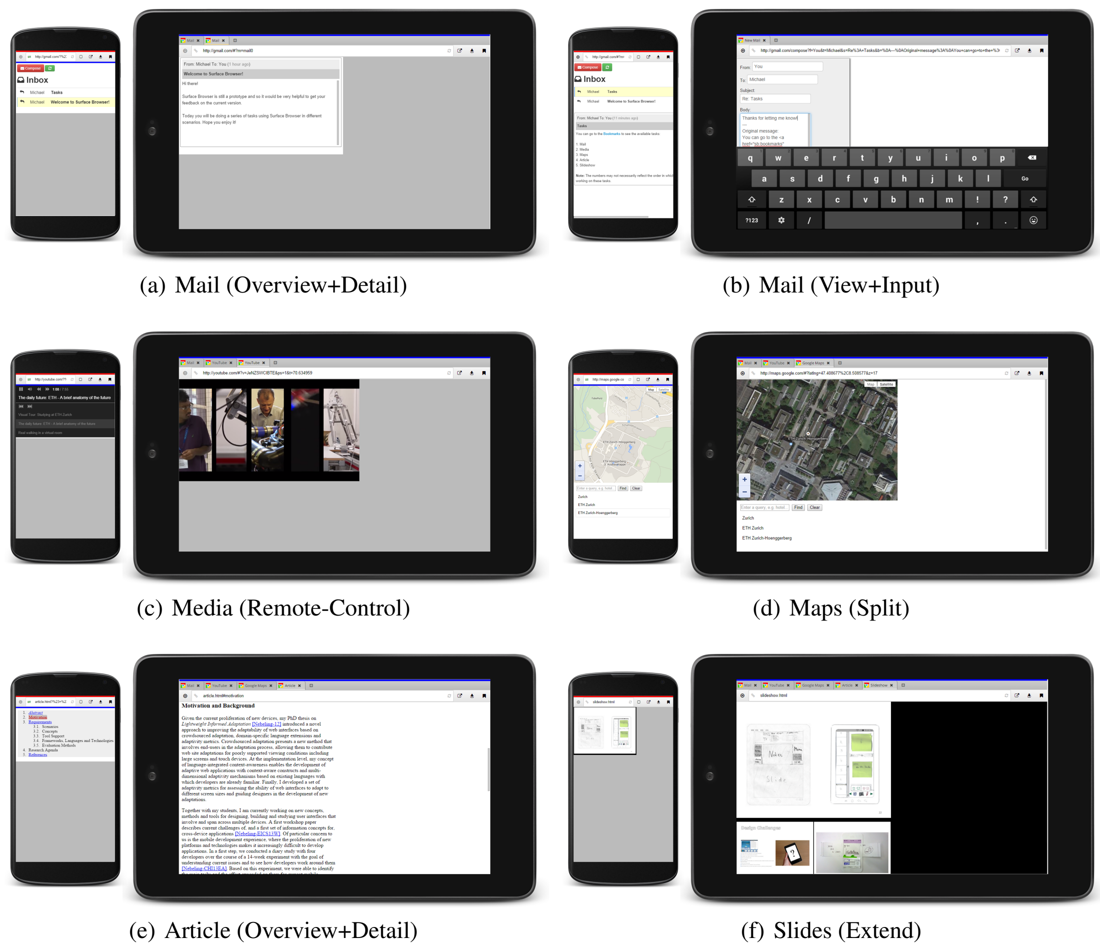

(a) Mail (Overview+Detail) (b) Mail (View+Input)
(c) Media (Remote-Control) (d) Maps (Split)
(e) Article (Overview+Detail) (f) Slides (Extend)

Figure 3. Examples of user-defined cross-device interfaces and the five cross-device design patterns that were most popular for these interfaces.

opted for a different cross-device design. In order to dedicate one device to input only, in this case, many participants used the tablet for composing mails and the phone for checking the inbox *and* reading mails (Figure 3(b)). They explained that the on-screen touch keyboard on the tablet is larger and more comfortable for typing. Since both tasks typically go hand in hand, participants wanted to quickly switch between the two different versions optimized for each task. To achieve this with our current prototype, participants opened two tabs with the *mail* interface, one for the customizations for reading and the other for writing mails. They then synchronized them and switched tabs according to the task.

Media Interface

The *media* interface was commonly customized for one task: playing media. The most frequent cross-device design showed the video in fullscreen on the tablet and the playback controls and the playlist on the smartphone (Figure 3(c)). However, participants distinguished between watching videos by themselves and showing videos to others, and made minor modifications in each case. In the first case, one participant copied one half of the video player on one device and the other half on the other device to make full use of the combined screen real estate. In the second case, another participant kept a smaller preview of the video on the smartphone for their own reference. In both cases, participants debated which devices should play sound. Participants often muted sound on the smartphone to limit audio to the tablet where speakers are usually better compared to those on smartphones. Some however played sound on both devices to have a better stereo effect when watching videos alone. One participant only played sound on the smartphone *via* headphones and watch the video on the tablet without having to worry about headphone cables. Participants also commonly suggested at some stage during the task that they might want to look up related information on the smartphone, *e.g.,* movie actors or lyrics for music videos. They could do this by disabling synchronization on the smartphone to use it independently from the tablet.

Maps Interface

The *maps* interface was customized for two primary tasks, namely for looking up a location on the map and for comparing maps. For looking up a location, the most popular cross-device design showed the map using a higher zoom factor on the tablet for a detailed view and a lower zoom level on the smartphone for an overview (Figure 3(d)) However, keeping the map views synchronized was a challenge since participants only wanted to synchronize the center of the map and not the zoom factor. This was tricky for XDBrowser's URL sync since the maps URL contains both parameters and hence synchronized both the center and the zoom level of the map view. For comparing maps, participants created different designs for comparing either the same location using two different map views or two different locations. In the first case, designs commonly showed the default map view on the phone and the terrain or satellite view on the tablet. These were considered useful for planning a bike trip and exploring the surroundings for walks and hikes. In the second case, the phone was used to zoom on one location and the tablet to zoom on another location. This was considered helpful for planning a road trip and deciding where to make a stop. For either task, participants found that the way XDBrowser currently synchronizes URLs was not ideal. They therefore only used synchronization up to the point of loading the *maps* interface on both devices and then disabled it to browse the maps independently on each device.

Article Interface

The *article* interface was customized for the primary task of reading the text and occasionally checking references. The most common cross-device design showed the article text on the tablet for reading and the table of contents and list of references on the smartphone (Figure 3(e)). When clicking on links in the table of contents or the references on the smartphone, participants commonly wanted XDBrowser to jump to the respective passage in the article on the tablet, rather than scrolling the page on the smartphone. To keep a lock on these parts of the page, participants cut the interface into two regions, one containing the table of contents and the other the references, to remove the article text from the smartphone. They explained that this design allowed them to use the smartphone for overview and reference purposes in support of the primary task of reading on the tablet.

Slides Interface

Participants customized the *slides* interface for a primary task, showing slides, and a secondary task, taking notes. For the primary task, the most common cross-device design showed the current slide on the tablet and the slide's previous and next previews on the smartphone (Figure 3(f)). For the secondary task, many wanted to exclusively use the smartphone for taking notes and show the rest of the interface on the tablet. Compared to the *mail* interface, however, they anticipated that switching between the two designs would not frequently occur while using the interface. They would either show slides or review slides and take notes. These two separate activities therefore did not create the same challenges as for the *mail* interface, where quickly switching between different customizations was commonly requested.

Design Strategies

In general, while customizing interfaces, participants considered and weighed the cost associated with using both devices, and necessarily having to switch between them, as opposed to just using one device for the task at hand. Although the vast majority of designs created by participants involved parallel use of both devices, not all of them did. If participants did not see the benefit in using both devices, they simply just used one of the devices for the task.

In observing participants customizing most interfaces for parallel multi-device use, we identified three common design strategies: *(1)* optimize for screen space, *(2)* optimize for input, and *(3)* minimize device switching.

The first strategy is motivated by participants' understanding that combining two devices provides more space for viewing content than using the devices in isolation. An obvious example of this strategy was the *media* interface with the player stretching across both devices. Other examples include the mail design for reading tasks where moving the inbox to the phone afforded more space on the tablet for reading messages, and the *maps* interface with overview and detail views on the tablet and phone. While we did not measure user performance during tasks, participants expected that optimizing along the screen size dimension reduces the time and effort required for between-page navigation (*e.g.,* inbox *vs.* compose mail), within-page navigation (*e.g.,* scrolling the article page), and panning/zooming (*e.g.,* viewing maps).

The second strategy was based on participants' preference for typing on the device with the larger keyboard where they felt they could type faster and with fewer errors. Clear instances of this strategy were observed for the *mail* and *slides* interfaces, where synchronization between the smartphone and the tablet allowed participants to view content on the smartphone while writing messages or taking notes on the tablet.

The third strategy was to keep the number of device switches during tasks low. This was exhibited through participants keeping the interface parts that they commonly required during a task together on the same device. On the other hand, the interface parts that helped them transition between tasks were commonly copied or moved to the other device. Again a good example is the *mail* interface where the tablet was used for the main task, reading messages, and the smartphone for opening messages on the tablet. This cross-device design only requires switching between devices when the user wants to select a different message for reading, which corresponds to starting a new task. However, when the larger task was writing messages, participants changed the design to keep the inbox and reading pane together on one device and used the tablet for writing.

Design Patterns

We reviewed each of the 144 designs created by participants, extracted common features and grouped the results into the largest classes. Seven multi-device patterns from three groups—*async patterns*, *copy and sync patterns*, and *move and sync patterns*—were identified in two separate passes and discussed with an independent cross-device researcher. Figure 4 gives an overview of the design patterns and how frequently they were employed in each customization task, while Figure 3 illustrates popular designs for the top five patterns. Below we present the groups and each pattern in turn.

1. Async Patterns

The first group consists of asynchronous multi-device use patterns, *i.e.,* using devices independently or not in full sync.

Single

While the vast majority were cross-device designs, three designs (2%) used only a single device. Three participants did not find it beneficial for the *media*, *maps* and *article* tasks

Pattern/Interface	Mail	Media	Maps	Article	Slides	Total
Async						
Single	0	1	1	1	0	**3**
Split	1	1	8	6	0	**16**
Overview+Detail	9	2	6	**12**	3	**32**
Copy and Sync						
Mirror	1	2	1	1	1	**6**
Extend	4	6	4	2	6	**22**
Move and Sync						
Remote-Control	4	**13**	**10**	4	**14**	**45**
View+Input	**13**	0	2	0	5	**20**
Total	**32**	**25**	**32**	**26**	**29**	**144**

Figure 4. Seven multi-device patterns, including five for cross-device use in parallel, and how often each was suggested in the five customization tasks. Highest-scoring patterns per task are indicated in gray shading.

to be able to use the two devices in combination and instead suggested using the original interface on just one of them.

Split

16 designs (11%) used the same interface on both devices in parallel, but in a not fully synchronized split view. This was popular for *maps*, *e.g.,* for route planning and exploring the surroundings using different map views (Figure 3(d)).

Overview+Detail

With 32 designs (23%), the second most popular pattern was to have only the parts of the interface that were useful for overview purposes on one device, and only the parts with details on the other. This was the most common proposal in the *article* task with 12 designs (8%) (Figure 3(e)).

2. Copy and Sync Patterns

The second group consists of patterns in which the page is copied and fully synchronized between devices.

Mirror

Six designs (4%) assigned equal roles to devices and duplicated the interface. Participants expected that the state of the interface would be fully synchronized between devices, *e.g.,* two suggested it in the *media* task for playback in stereo.

Extend

22 designs (15%) replicated parts of the interface on the second device so that the interface extended across both devices. For example, a common design for *slides* was to replicate the current slide on the second device (Figure 3(f)).

3. Move and Sync Patterns

The third group consists of patterns that move parts of the page from one device to the other and keep them in sync.

Remote-Control

The most popular pattern with 45 designs (31%) moved parts of the interface to another device for remote control. This was most common in the *media* and *slides* tasks where it was popular to have all controls only on the phone (Figure 3(c)).

View+Input

20 designs (14%) used one device for viewing content and the other for text input. This was most popular for *mail* with the phone used for viewing the inbox and the tablet for writing messages as input was perceived to be easier on the tablet given the larger virtual keyboard (Figure 3(b)).

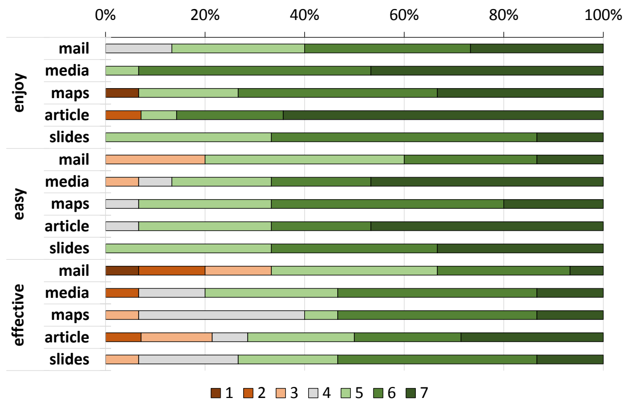


Figure 5. Post-task feedback on whether participants enjoyed operating XDBrowser, found the task easy to perform and provided tools effective.

User Feedback

When asked about the main benefits of XDBrowser, one participant said: *"It lets you take advantage of the strengths of each device, so that you have the smoothest overall experience"* (P1). It was considered useful for *"combining display and input devices for an improved user experience"* (P4) and for *"the fact that you can easily switch from one device to the other by means of sync"* (P9). Participants appreciated XDBrowser's *"distribution of the view area and the input area to maximize the view area"* (P13) and the ability to *"move typing tasks to the tablet"* (P12). Other aspects they highlighted included that *"less scrolling is necessary because I can navigate using a second device"* (P15) and that they could use XDBrowser *"to show to another person what I see on my screen"* (P14).

We also asked participants what they think are the main shortcomings of our prototype. In line with the challenges of creating cross-device interfaces for some tasks, it was considered necessary to *"make it easier to switch between tabs when combined"* (P1). Currently, *"too many steps are required for selecting content to be shared, should be automated for me"* (P3). For some interfaces, some participants were *"not sure all the time what gets synced and what doesn't"* (P15) and found that *"sync control could be more sophisticated"* (P5).

From the post-task feedback shown in Figure 5, we can see that participants enjoyed using XDBrowser for the tasks, found it easy to perform the tasks and the customization tools creating cross-device designs effective. Across all tasks, the median ratings were at or above 5 on a 7-point Likert-scale (7 = best).

Enjoyment

Overall, participants enjoyed using XDBrowser. Two participants answered that, while the tasks were easy and the features seemed effective, they would not enjoy using the two devices in combination for the *maps* and *article* interfaces and gave it a low rating for that reason.

Ease of Use

The *mail* task was not so easy for three participants who wanted to open the compose page directly on the other device without having to first open it on, and then push it to, the target device, which is currently required. The *media* task was not so easy for some participants due to the fact that autoplay is not available on mobile devices as enforced by the HTML5 standard, which required the video player to be started once on each device before it could be synchronized.

Effectiveness

The browser's features were generally deemed effective with some differences between tasks. Again, participants gave lower ratings in the *mail* and *media* task for the same issues noted above. The *maps* task received several slightly negative to neutral ratings mainly due to the fact that synchronizing the map's bounds between devices meant that also the zoom level remained in sync. This was contrary to some proposals of participants who wanted to maintain the same location, but have different views and zoom levels on the two devices. The lower ratings in the *article* task were given by participants who, as also requested for the mail application, wanted to open links in the text directly on the other device, rather than jumping to the anchor on the current device. Finally, the *slides* task received lower effectiveness ratings by participants who wanted to scale the current slide to fill the viewport. This was also often requested in the *media* task; however, there it was not an issue since YouTube supports fullscreen playback.

DISCUSSION

At the highest level, we differentiate distribution and synchronization, and distinguish two distribution methods, *copy/move*, and four synchronization options, *click/scroll/input/URL sync*. Many different designs were possible and a pattern analysis grouped the results into the seven largest classes. Our patterns can be seen as distinct but not disjoint, *i.e.,* a particular design can have elements in common with more than one pattern. Two pieces of information were used for disambiguation: *(i)* main function of distributed elements based on separation of content, navigation and presentation, *(ii)* participants' stated intent behind each distribution and role of each device. Figure 3(c) is a *remote-control* example because all navigation elements were moved to the phone with the intent to control video playback on the tablet. Figure 3(e) is *overview+detail* rather than *remote-control* although the table of contents with some navigation elements was moved to the phone because the intent was still to navigate the article primarily on the tablet.

Several recent works have presented cross-device designs for mail clients, media players, map browsers, document readers and gallery viewers similar to the five tasks we studied [4, 8, 11, 25, 36]. However, these were commonly designed by developers. Many of the designs proposed by developers can be classified as *mirror*, *extend*, *remote-control* and *overview+detail* patterns. While there is agreement in terms of classification, overall we found much less usage than expected for some of these patterns in our study. For example, *mirror* and *extend* designs from the *copy and sync* category were not very popular among participants. We suspect that this may have to do with our study only using two devices and that duplication or replication of interfaces may be more desired with additional devices. However, this is something that remains to be established.

Much more popular in our study were the *move and sync* patterns instead. Here, the examples shown in the literature missed some designs frequently created by our participants. For example, Duet [4], Conductor [11] and Panelrama [36] all present similar cross-device designs for the *mail* and *maps* interfaces. However, we find two major differences in the results of our study. First, our study showed a lot of variety with many different designs from the *move and sync* pattern category especially for these two interfaces. Second, none of them considered different tasks for each interface whereas our participants distinguished between primary and secondary tasks for which they wanted different cross-device designs.

The design strategies and patterns we extracted from user-defined designs establish the variety of desired designs and the data we collected on user preferences can guide designers in making informed decisions about choosing the right pattern for different tasks. Our study helps to provide a better picture of the mobile cross-device design space and can inform future design in this context. Our motivation to focus on two mobile devices was to generate a rich set of patterns for a setting that is increasingly common [17, 29, 30]. If existing web sites were to implement our patterns, users would immediately benefit.

Our study also raises new technical and design issues. First, while users often associated specific roles with devices, the same users frequently switched them depending on the task. Our attempt to tackle the issues at the browser rather than interface level is therefore a step in the right direction since the users themselves can use the tools to easily switch device roles making their own customizations without having to rely on developers to make the necessary changes for them. Second, an important question raised by participants is *what is* and *what is not* synchronized. XDBrowser currently supports four synchronization options that proved sufficient for customizing the interfaces for most tasks. However, in some cases, participants asked for more control while others wanted XDBrowser to make the decision for them. Our study therefore motivates further research into the means of providing fine-grained support and automation of different interface distributions with synchronization control at the content, navigation and presentation levels. It will also be interesting to study cross-apps tasks involving multiple web sites. Building on the results of this paper, these are important avenues for future research.

CONCLUSION

This paper has presented a user study around XDBrowser, a new kind of web browser we are developing to enable a better cross-device web browsing experience. XDBrowser extends previous end-user customization techniques to cross-device use. Our user study demonstrates its potential to support a wide range of common web browsing tasks and elicit desirable cross-device interfaces leading to seven design patterns.

The results generated so far are useful in at least three ways. First, they provide an overview of possible cross-device designs able to guide developers of distributed interfaces. Second, they aid classification of cross-device designs. Third, they provide a basis for comparison in future cross-device studies.

Acknowledgments

Michael Nebeling started this research with support from ETH Zurich. It was continued at CMU under mobility grant, P300P2_154571, by the Swiss National Science Foundation.

REFERENCES

1. Brian P. Bailey, Jacob T. Biehl, Damon J. Cook, and Heather Metcalf. 2008. Adapting paper prototyping for designing user interfaces for multiple display environments. *PUC* 12, 3 (2008), 269–277. DOI: http://dx.doi.org/10.1007/s00779-007-0147-2

2. Nilton Bila, Troy Ronda, Iqbal Mohomed, Khai N. Truong, and Eyal de Lara. 2007. PageTailor: Reusable End-User Customization for the Mobile Web. In *Proc. MobiSys*. 16–29. DOI: http://dx.doi.org/10.1145/1247660.1247666

3. Nicholas Chen, François Guimbretière, and Abigail Sellen. 2013. Graduate Student Use of a Multi-Slate Reading System. In *Proc. CHI*. 1799–1808. DOI: http://dx.doi.org/10.1145/2470654.2466237

4. Xiang 'Anthony' Chen, Tovi Grossman, Daniel J. Wigdor, and George W. Fitzmaurice. 2014. Duet: Exploring Joint Interactions on a Smart Phone and a Smart Watch. In *Proc. CHI*. 159–168. DOI: http://dx.doi.org/10.1145/2556288.2556955

5. Pei-Yu (Peggy) Chi and Yang Li. 2015. Weave: Scripting Cross-Device Wearable Interaction. In *Proc. CHI*. 3923–3932. DOI: http://dx.doi.org/10.1145/2702123.2702451

6. David Dearman and Jeffrey S. Pierce. 2008. It's on my other computer!: computing with multiple devices. In *Proc. CHI*. 767–776. DOI: http://dx.doi.org/10.1145/1357054.1357177

7. Eli Raymond Fisher, Sriram Karthik Badam, and Niklas Elmqvist. 2014. Designing peer-to-peer distributed user interfaces: Case studies on building distributed applications. *IJHCS* 72, 1 (2014), 100–110. DOI: http://dx.doi.org/10.1016/j.ijhcs.2013.08.011

8. Luca Frosini and Fabio Paternò. 2014. User Interface Distribution in Multi-Device and Multi-User Environments with Dynamically Migrating Engines. In *Proc. EICS*. 55–64. DOI: http://dx.doi.org/10.1145/2607023.2607032

9. Giuseppe Ghiani, Fabio Paternò, and Carmen Santoro. 2012. Push and Pull of Web User Interfaces in Multi-Device Environments. In *Proc. AVI*. 10–17. DOI: http://dx.doi.org/10.1145/2254556.2254563

10. Giuseppe Ghiani, Fabio Paternò, and Carmen Santoro. 2013. Interactive Customization of Ubiquitous Web Applications. *VLC* 24, 1 (2013), 37–52. DOI: http://dx.doi.org/10.1016/j.jvlc.2012.10.005

11. Peter Hamilton and Daniel J. Wigdor. 2014. Conductor: Enabling and Understanding Cross-Device Interaction. In *Proc. CHI*. 2773–2782. DOI: http://dx.doi.org/10.1145/2556288.2557170

12. 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*. 142–147. DOI: http://dx.doi.org/10.1145/2611009.2611026

13. Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proc. CHI*. 1247–1256. DOI: http://dx.doi.org/10.1145/2702123.2702215

14. 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*. 199–214. DOI: http://dx.doi.org/10.1007/978-3-319-11746-1_15

15. Heather M. Hutchings and Jeffrey S. Pierce. 2006. Understanding the Whethers, Hows, and Whys of Divisible Interfaces. In *Proc. AVI*. 274–277. DOI: http://dx.doi.org/10.1145/1133265.1133320

16. Brad Johanson, Shankar Ponnekanti, Caesar Sengupta, and Armando Fox. 2001. Multibrowsing: Moving Web Content across Multiple Displays. In *Proc. Ubicomp*. 346–353. DOI: http://dx.doi.org/10.1007/3-540-45427-6_29

17. Tero Jokela, Jarno Ojala, and Thomas Olsson. 2015. A Diary Study on Combining Multiple Information Devices in Everyday Activities and Tasks. In *Proc. CHI*. 3903–3912. DOI: http://dx.doi.org/10.1145/2702123.2702211

18. 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*. 722–735. DOI: http://dx.doi.org/10.1007/978-3-642-03655-2_79

19. 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*. 2009–2018. DOI: http://dx.doi.org/10.1145/1753326.1753631

20. Amy K. Karlson, Brian Meyers, Andy Jacobs, Paul Johns, and Shaun K. Kane. 2009. Working Overtime: Patterns of Smartphone and PC Usage in the Day of an Information Worker. In *Proc. Pervasive*. 398–405. DOI: http://dx.doi.org/10.1007/978-3-642-01516-8_27

21. Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa A. Lau. 2008. CoScripter: Automating & Sharing How-To Knowledge in the Enterprise. In *Proc. CHI*. 1719–1728. DOI: http://dx.doi.org/10.1145/1357054.1357323

22. James Lin and James A. Landay. 2008. Employing Patterns and Layers for Early-Stage Design and Prototyping of Cross-Device User Interfaces. In *Proc. CHI*. 1313–1322. DOI: http://dx.doi.org/10.1145/1357054.1357260

23. Jan Meskens, Kris Luyten, and Karin Coninx. 2010. DMacs: Building Multi-Device User Interfaces by Demonstrating, Sharing and Replaying Design Actions. In *Proc. UIST*. 129–138. DOI: http://dx.doi.org/10.1145/1866029.1866051

24. Jan Meskens, Jo Vermeulen, Kris Luyten, and Karin Coninx. 2008. Gummy for Multi-Platform User Interface Designs: Shape me, Multiply me, Fix me, Use me. In *Proc. AVI*. 233–240. DOI: http://dx.doi.org/10.1145/1385569.1385607

25. Michael Nebeling, Theano Mintsi, Maria Husmann, and Moira C. Norrie. 2014. Interactive Development of Cross-Device User Interfaces. In *Proc. CHI*. 2793–2802. DOI:http://dx.doi.org/10.1145/2556288.2556980

26. Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. 2013. CrowdAdapt: Crowdsourced Web Page Adaptation for Individual Viewing Conditions and Preferences. In *Proc. EICS*. 23–32. DOI: http://dx.doi.org/10.1145/2480296.2480304

27. Jeffrey Nichols, Zhigang Hua, and John Barton. 2008. Highlight: A System for Creating and Deploying Mobile Web Applications. In *Proc. UIST*. 249–258. DOI: http://dx.doi.org/10.1145/1449715.1449757

28. Antti Oulasvirta and Lauri Sumari. 2007. Mobile Kits and Laptop Trays: Managing Multiple Devices in Mobile Information Work. In *Proc. CHI*. 1127–1136. DOI: http://dx.doi.org/10.1145/1240624.1240795

29. Roman Rädle, Hans-Christian Jetter, Mario Schreiner, Zhihao Lu, Harald Reiterer, and Yvonne Rogers. 2015. Spatially-aware or Spatially-agnostic?: Elicitation and Evaluation of User-Defined Cross-Device Interactions. In *Proc. CHI*. 3913–3922. DOI: http://dx.doi.org/10.1145/2702123.2702287

30. Stephanie Santosa and Daniel Wigdor. 2013. A Field Study of Multi-Device Workflows in Distributed Workspaces. In *Proc. UbiComp*. 63–72. DOI: http://dx.doi.org/10.1145/2493432.2493476

31. Dominik Schmidt, Julian Seifert, Enrico Rukzio, and Hans Gellersen. 2012. A Cross-Device Interaction Style for Mobiles and Surfaces. In *Proc. DIS*. 318–327. DOI: http://dx.doi.org/10.1145/2317956.2318005

32. Teddy Seyed, Chris Burns, Mario Costa Sousa, Frank Maurer, and Anthony Tang. 2012. Eliciting Usable Gestures for Multi-Display Environments. In *Proc. ITS*. 41–50. DOI:http://dx.doi.org/10.1145/2396636.2396643

33. Desney S. Tan, Brian Meyers, and Mary Czerwinski. 2004. WinCuts: Manipulating Arbitrary Window Regions for More Effective Use of Screen Space. In *Proc. CHI EA*. 1525–1528. DOI: http://dx.doi.org/10.1145/985921.986106

34. Minna Wäljas, Katarina Segersthl, Kaisa Väänänen-Vainio-Mattila, and Harri Oinas-Kukkonen. 2010. Cross-Platform Service User Experience: A Field Study and an Initial Framework. In *Proc. Mobile HCI*. 219–228. DOI: http://dx.doi.org/10.1145/1851600.1851637

35. Heather Wiltse and Jeffrey Nichols. 2009. PlayByPlay: Collaborative Web Browsing for Desktop and Mobile Devices. In *Proc. CHI*. 1781–1790. http://doi.acm.org/10.1145/1518701.1518975

36. Jishuo Yang and Daniel Wigdor. 2014. Panelrama: Enabling Easy Specification of Cross-Device Web Applications. In *Proc. CHI*. 2783–2792. DOI: http://dx.doi.org/10.1145/2556288.2557199