Interactive Development of Cross-Device User Interfaces

Michael Nebeling
Institute of Information Systems, ETH Zurich
nebeling@inf.ethz.ch

Theano Mintsi
EPFL
theano.mintsi@epfl.ch

Maria Husmann
Moira C. Norrie
Institute of Information Systems, ETH Zurich
{husmann,norrie}@inf.ethz.ch

ABSTRACT
Current GUI builders provide a design environment for user interfaces that target either a single type or fixed set of devices, and provide little support for scenarios in which the user interface, or parts of it, are distributed over multiple devices. Distributed user interfaces have received increasing attention over the past years. There are different, often model-based, approaches that focus on technical issues. This paper presents *XDStudio*—a new GUI builder designed to support interactive development of cross-device web interfaces. XDStudio implements two complementary authoring modes with a focus on the design process of distributed user interfaces. First, *simulated authoring* allows designing for a multi-device environment on a single device by simulating other target devices. Second, *on-device authoring* allows the design process itself to be distributed over multiple devices, as design and development take place on the target devices themselves. To support interactive development for multi-device environments, where not all devices may be present at design and run-time, XDStudio supports switching between the two authoring modes, as well as between design and use modes, as required. This paper focuses on the design of XDStudio, and evaluates its support for two distribution scenarios.

Author Keywords
multi-device, distributed user interfaces; simulated authoring; on-device authoring.

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

INTRODUCTION
Current GUI builders provide a design environment for user interfaces that target either a single type or fixed set of devices. In particular, they provide little support for scenarios in which the user interface may be distributed over multiple devices. Distributed, multi-device user interfaces have received increasing attention over the past years [18]. There are different, often model-based, approaches that focus on technical issues, e.g. [12], while only some also address issues and challenges specific to the design process [11, 14].

This paper presents *XDStudio*—a new, web-based GUI builder specifically designed to explore interactive development of cross-device user interfaces based on two authoring modes. In the first, *simulated mode*, one device is used as the central authoring device, while target devices are simulated. In the second, *on-device mode*, the design process is also coordinated by a main device, but directly involves target devices. In addition, XDStudio distinguishes between *use mode* for normal interaction with an interface loaded into our editor and *design mode* for direct manipulation of the interface. XDStudio enables switching between these modes, allowing the designer to check the distribution at any moment and return to the design process for further editing as required. XDStudio makes the following contributions.

- We explore two scenarios that we developed as interesting use cases of multi-device, distributed user interfaces and used to drive the design and evaluation of XDStudio.
- Our notion of distribution profiles at the core of XDStudio extends existing context models—designed with a single device and user in mind—to multiple devices and users.
- We present cross-device authoring concepts and tools that cater for cases where not all devices and users are available, or where they are even different, at design and run-time.
- Finally, based on a first user study with 12 participants, we show that it is beneficial to have both modes for simulated and on-device authoring available, and that on-device authoring is preferred for device-centric scenarios.

We start with a discussion of related work in the next section. This is followed by two distribution scenarios that XDStudio aims to support. We then present the concepts and tools of XDStudio, and how they were implemented. Finally, our user study will be described, analysed and discussed in detail, before giving concluding remarks.

RELATED WORK
XDStudio draws from research on context-aware adaptation, multi-device design tools, and distributed user interfaces.

Context-Aware Adaptation
Previous research on model-based user interfaces has contributed a number of different models, languages and tools for the development of multi-device interfaces. Many solutions are based on the reference framework described by Calvary et al. [1]. The common goal is to allow designers to specify interfaces at a high level of abstraction based on the models and languages. The suggested authoring process typically unfolds

along a four-step, top-down approach, starting with domain concepts and task modelling, followed by subsequent transformation steps from abstract to concrete and the final user interfaces. Some approaches support automatic transformation between some of the levels [10], while others rely more or less on the designer to perform the mapping manually [19]. In combination with a context and adaptation model, it is possible to generate operational interfaces suitable for a variety of use contexts. Yet, context information taken into account for adaptation commonly concerns a single user and device at a time, and so would need to be extended to cater for the forms of cross-device user interfaces considered here.

Multi-Device Design Tools

The authoring of multi-device user interfaces has been the subject of extensive research [18]. Many different user interface description languages have been proposed. Two widely studied approaches include TERESA and MARIA that follow the aforementioned user interface abstraction model and also support web interface development [19]. While TERESA and MARIA can be regarded as authoring tools for the underlying user interface description languages, other examples closer to WYSIWYG include Damask [11] and Gummy [14].

Damask is a prototyping tool for creating sketches of multi-device web interfaces. The tool provides a fixed set of user interface design patterns that are optimised for desktop and mobile applications including support for speech input and output. User interfaces are created in layers to indicate the elements common to all platforms and those specific to a particular device and/or modality. While the pattern-based approach showed good results in a study with designers, scalability of the approach is questionable since the system provides a limited set of design patterns that are not extensible.

Gummy is a design environment for graphical user interfaces that allows designers to create interfaces for multiple devices using visual tools that automatically generate and maintain a platform-independent description of the interface. In addition, initial designs of the user interface for a new platform can be generated based on existing interfaces created for the same application. Gummy then maintains consistency between the different versions and, to ease development for richer or more limited platforms, it also adapts its authoring environment for different target platforms.

The original approach based on Gummy has seen a series of extensions with special tools for managing device-specific interfaces by allowing for cross-device copy-pasting and linked editing in support of consistency. Furthermore, design tool macros enable designers to record and share their design actions across devices and replay them for creating new interfaces. The latest extension is GUIDE2ux [13] which provides on-the-fly feedback about the design by showing whether preset design rules are violated. It also allows designers to inspect an interface live on the target device. XDStudio extends this work by introducing new modes for switching between live design previews and authoring directly on the devices.

Recent multi-device design tools take additional steps towards collaborative design and crowdsourced adaptation. Quill [4] supports collaboration between different stakeholders (developer, project manager, support team leader, etc.) by adopting a model-based design of cross-platform user interfaces. CrowdAdapt [17] uses crowdsourcing to collectively improve the adaptability of a web site under different viewing conditions. In contrast to Quill, CrowdAdapt builds on direct manipulation of the final interface without the need of using a particular web design method or model. XDStudio builds on this direct manipulation approach, making our tool suitable for designers with basic web development experience.

Distributed User Interfaces

Researchers have experimented with concepts and techniques to dynamically support partial or complete distribution of user interfaces [18]. Different approaches were explored with object-oriented frameworks such as ZOIL [9], data-oriented frameworks such as Shared Substance [7] and model-based frameworks such as [12]. However, compared to XDStudio, they provide little visual support for authoring distributed user interfaces. Rather, the focus is on powerful and expressive models and languages. As a result, the proposed solutions often tie in with specific programming paradigms, impose multiple different abstraction levels, and even require proprietary languages for programming and specification. In the case of web applications, most solutions build on HTML proxy-based techniques to dynamically push and pull user interfaces [5], a technique that was only recently extended for interactive customisation of the distribution at run-time [6].

XDStudio draws inspiration from all these works, but extends existing concepts for interactive cross-device development. Specifically, based on two different distribution scenarios, we develop the methods of simulated and on-device authoring and explore the benefits and tradeoffs of each of them.

SCENARIOS

This section presents two scenarios that XDStudio aims to support. The first takes place in a meeting room equipped with multiple interactive devices similar to various intelligent room projects known from the literature. The second scenario is based on a classroom setting adapted from an interactive collaborative learning method [2]. The two scenarios are different in that the first focuses on collaboration of multiple types of devices, while the second emphasises different roles of users participating in the distribution.

Scenario 1: Meeting Room

A team regularly meets up in a modern meeting room to discuss projects. Each team member brings along personal devices, such as smartphones, tablets, and laptops. Additionally, a large display is installed in the room, which can be used to project presentations, and a medium-sized touchscreen is present for displaying complementary information, such as the agenda, to the group. The team uses the large screen to present slides taking turns. The current presenter can control the slide show from their tablet. The other team members can submit questions and comments from their personal devices, which will also be shown on the presenter's tablet. A team member arrives a little later and their device needs to be integrated into the system while the presentation is already

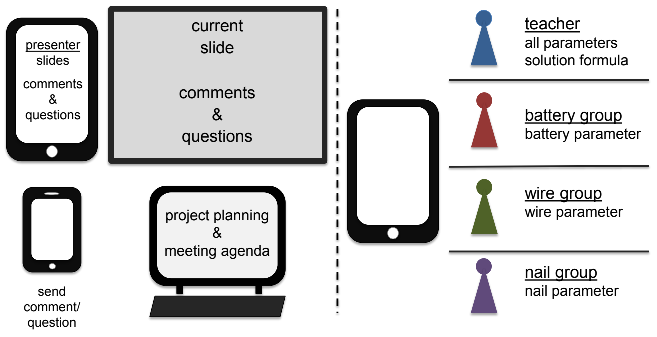


Figure 1: Characteristics of meeting room and classroom distribution scenarios

running. Another member is at a different location and would like to participate remotely. The current slide and the option to submit questions are available on their mobile phone.

Scenario 2: Classroom

In a classroom, where the teacher and all students are equipped with tablets, an educational application is used to teach physics to the students. The application lets the students change a set of variables of a simulated experiment, run it, and view the results. In order to foster interaction, the students are split into groups and each group only gets access to one variable and the result. Each group can manipulate the variable, but only the teacher has access to the whole experiment and also the underlying formula of the experiment. After the groups have experimented and discussed the results, the teacher decides to publish the formula to all the students.

We developed these two scenarios, not only because they illustrate interesting use cases for distributed user interfaces, but also because they exhibit different characteristics (Figure 1). The first contains a dynamic set of devices that may change as users enter or leave the meeting. Also the type of the device partly determines its role in the scenario, e.g. the large screen is used to present slides. On the other hand, in the second scenario, all involved devices are of the same type and the distribution would be based on the users' roles.

Based on these scenarios and also taking into account relevant literature, we have derived the following requirements for XDStudio. Compared to rather high-level requirements described in the literature, e.g. user interface distribution, migration and granularity [18] or context-awareness, portability, scalability and consistency [4], our requirements are more specific. Moreover, in addition to common design-time requirements, we explicitly distinguish run-time requirements.

Design-Time Requirements

(R1) Detection and integration of present devices
The system must allow the user to design for different device types [1] and detect devices connected at design-time.

(R2) Design for unknown devices
The system must provide the option to design a distributed user interface for devices that are not available at design-time [1].

(R3) User roles
The system must support the design of a distributed interface for diverse user roles [4]. For example, in the classroom scenario, a different view is created for the students and the teacher, while both use the same device type.

(R4) Support for designing and testing
The system must allow seamless switching between design and use of the interface in order to support an incremental and iterative design process [13].

(R5) Adaptation of user interface elements
The system must support changing the properties of user interface widgets such as size and position in order to accommodate different device types [14].

(R6) Adaptation of interaction between devices
The system must support changing how devices interact with each other [13]. For example, in the classroom, the students in one group should not be able to interfere with another group's experiment.

(R7) Reuse of existing interfaces and distributions
The system must support the reuse of an existing interface, or parts of it, as well as the definitions of devices and user roles so that they may be shared among multiple distributed interfaces [13].

Run-Time Requirements

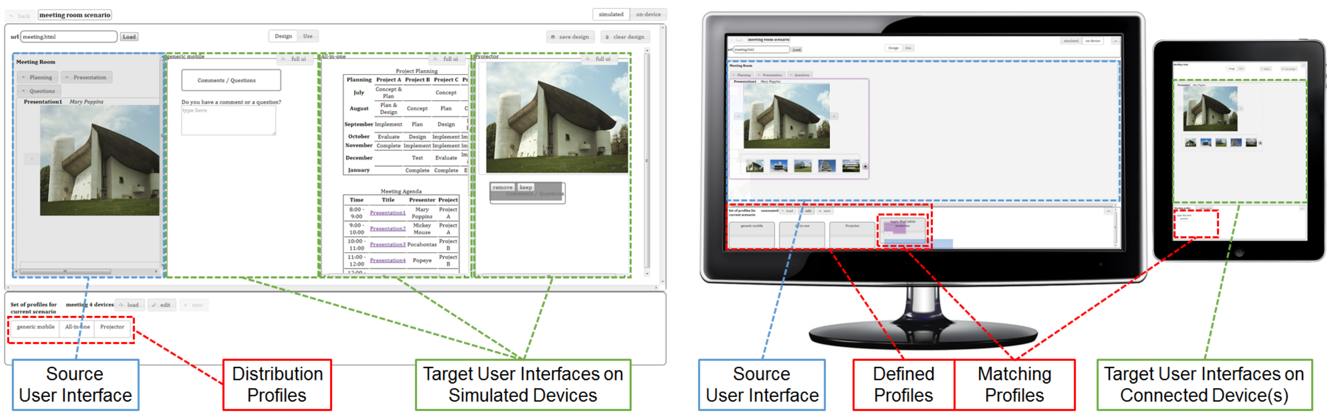
(R8) Dynamic detection and integration of available devices
The system must detect the devices and users present, providing each device with the corresponding view of the interface. The system must support devices joining and leaving at any point in time.

(R9) Update the distribution at runtime
The system must allow an authorised user to change the interface at run-time. For example, in the classroom scenario, the teacher updates the distribution during run-time, so that the students can access the formula only after having done the exercise.

(R10) Distribution Matching and Adaptation
The system must update the design-time distribution to match available devices and adapt accordingly to ensure that it is functional at run-time. For example, if the large screen in the meeting room scenario is missing, the system should adapt and show the presentation on another device.

XDSTUDIO

XDStudio was specifically designed for authoring distributed user interfaces based on three main features. First, it is based on a new concept of *distribution profile* that can be used to specify a distribution scenario in terms of involved devices, users and target user interfaces. Second, it supports two complementary *authoring modes* designed to support interactive development and make the design process more flexible. Third, it can further distinguish between *design and use modes* to develop and test distributions as required. Before discussing the concepts in more detail, we will first give a walkthrough of the system.



(a) Simulated Authoring (b) On-device Authoring

Figure 2: XDStudio supports a) *simulated authoring* on a central device and b) *on-device authoring* directly on target devices, as well as flexibly switching between both authoring modes to accommodate different distribution scenarios

System Walkthrough

Our walkthrough starts with the simulated, design mode shown in Figure 2a on the central authoring device. Later, we will alternate between design and use modes and also between simulated and on-device modes. As shown in Figure 2b, the latter directly involves connected target devices in the authoring process. We will refer to the meeting room and classroom scenarios presented in the previous section to illustrate how XDStudio may be used to design the distributed interface in each case. Current support for the requirements extracted from these scenarios will also be discussed.

Definition of Distribution Profiles

To get started, XDStudio requires the specification of at least one distribution profile. As illustrated in Figure 3a, the creation of a profile involves the definition of one or multiple devices and users involved in the distribution scenario.

As for devices, XDStudio distinguishes between different types of devices and specific devices based on string identifiers. The current implementation supports the device types desktop, tablet, mobile phone and other. Other may be used to describe types currently not known to the system. Examples of specific devices include “iPad” for tablet and “iPhone” for mobile phone. By default, “generic” is used to match non-specific devices of a given type, e.g. “generic tablet”. Similarly, XDStudio can associate a profile with user roles. For example, it is possible to define a role “presenter” or “teacher” to distinguish users in different roles. The user role is currently obtained based on the client name, but proper user identification could be added in the future.

Based on the scenario characteristics summarised in Figure 1, for the meeting room, a presenter profile could be defined as “presenter” tablet, a projector profile as other, a touchscreen profile as desktop, and a mobile profile as mobile phone. On the other hand, for the classroom, a teacher profile could be defined as “teacher” tablet, while the three student groups could be distinguished based on their role in the physics experiment, e.g. “battery-group” tablet for the first group of students controlling the battery parameter.

The set of profiles can be modified at any moment during the authoring process. XDStudio makes all defined profiles available as toggle buttons, enabling the user to select one or more profiles and load the associated target user interfaces into the editor. Figure 2a shows the target interfaces for the mobile, touchscreen, and projector profiles of the meeting room.

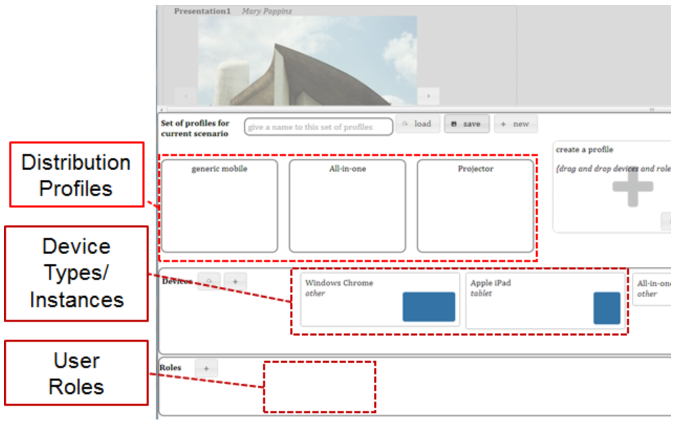
Tools and Modes for Authoring

XDStudio provides an editor that can load and display a source interface and essentially enable two ways of authoring the distribution in terms of target user interfaces (Figure 3b). For each profile, the user can select which user interface widgets and other elements of the source interface are to be included in the distribution by dragging and dropping them from the source interface to the corresponding target interfaces. Alternatively, they can click a button, Full UI, available for each target interface to insert the full version of the source interface as a starting point. They may then select specific interface elements and choose Keep/Remove in order to include/exclude them from the distribution. To clear the target interface, they can click the same button for Empty UI.

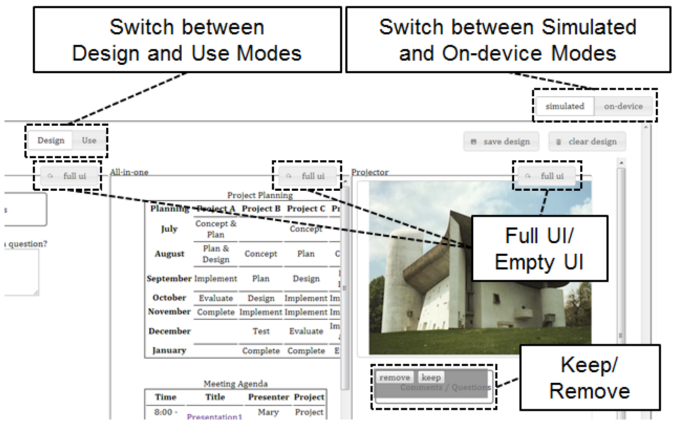
By default, XDStudio always starts with an empty target interface for a new profile. It may also be configured to always use the full interface as the starting point instead. The preferred configuration for this option may depend on the complexity of both the source interface and the distribution scenario in terms of the number of interface widgets that need to be distributed and the differences between profiles.

XDStudio starts in simulated, design mode. At any time during the authoring process, the user is free to alternate between simulated and on-device modes and between design and use modes. The selected mode is kept consistent between the central authoring device and all simulated or connected devices.

In simulated mode, the authoring process is fully controlled and carried out only on one device that we refer to as the central authoring device. All target devices are simulated and displayed on this device. After switching to on-device mode, the central authoring device is only used to show the source



(a) Distribution Profiles


Switch between Design and Use Modes
Switch between Simulated and On-device Modes
Full UI/ Empty UI
Keep/ Remove

(b) Authoring Modes and Tools

Figure 3: XDStudio manages distribution profiles for one or multiple devices and users and provides specific authoring tools

interface and defined profiles. The target interfaces are then instead displayed on the target devices themselves.

While in design mode, all widgets and interactive elements of the source and target interfaces are disabled. Instead, users can create and manipulate the target interfaces for the selected profiles as described above. On the other hand, in use mode, our authoring tools are disabled so that the source and target interfaces will again react to user input as if the interface was loaded directly in the browser rather than in our editor.

Note that, in on-device mode, the central authoring device is still used to control the authoring process. This is required to keep the distributed user interface functional in cases when the source interface is only partially distributed over target devices. Similar to simulated authoring, the Full UI/Empty UI and Keep/Remove authoring tools are available for the target interfaces. The distribution can also still be designed via drag-n-drop. However, this operation now requires dragging and dropping elements from the source interface to the profile icons at the bottom of the screen as a reference to the corresponding target interfaces running on the other devices.

To give an example, Figure 2b shows the on-device mode for the "presenter" tablet profile of the meeting room scenario on an iPad. Here, the user has just dragged the slideshow presentation widget and dropped it onto the iPad tablet "presenter" profile icon. The result of this design action is immediately shown on the iPad. Switching to use mode, if the user now clicks the Next button, the next slide of the presentation is simultaneously displayed on all connected devices that have this widget included in their target interface as well as on the central authoring device showing the source interface. If the user wants to somehow modify the distribution, they just need to switch back to design mode for further editing.

Support for Requirements

The features described above address the majority of requirements elicited previously. Our definition and use of distribution profiles during the authoring process fulfills $R1$ to $R3$ as well as $R8$. XDStudio automatically detects connected devices and integrates them into the authoring process ($R1$, $R8$). Device detection is based on the browser's user agent string and logic similar to MobileESP[1]. Unknown devices such as the projector in the meeting room scenario can be handled using XDStudio's generic devices ($R2$). Although user login is not yet properly managed, XDStudio provides basic support for user roles using the client name ($R3$). Flexible support for designing and testing ($R4$) and for updating the distribution at run-time ($R9$) is based on the design and use modes.

Currently, XDStudio only provides initial support for the remaining design and run-time requirements. The adaptation of user interface elements ($R5$) is a focus of many existing design tools that was out of scope of our proof-of-concept. For example, CrowdAdapt [17] implements a set of 7 design operations for adaptation to both handheld and large-screen devices. $R6$ for changing the interaction between devices is currently only implemented for the basic case of disabling certain interactions on certain devices in that the elements responsible for triggering the interaction may be excluded from the distribution based on the Remove UI Element tool.

Even without full implementation, we will explain below how $R5$ to $R7$ and $R10$ are covered based on the concepts behind XDStudio. We also formally show extensibility.

Concepts

The design of distributed user interfaces using XDStudio is based on a new concept of distribution profile. A distribution profile describes a distribution in terms of two primary components. First, it specifies the context in terms of the devices on which the distributed user interface will be displayed, the roles of users that may be handling these devices, as well as other context factors. Second, based on the source interface, it specifies the set of target user interfaces in terms of the elements that will be distributed and their adaptations with respect to different contexts, e.g. in terms of device characteristics and user requirements. The separation of context and target user interfaces makes it easier to define and handle different versions of a distribution and enables reuse between different user interfaces and distribution scenarios ($R7$).

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

Formally, context in our concept of distribution profile is based on the general context representation developed in [15]. XDStudio does not impose a particular context model. Rather, context is represented based on two sets, NAMES and VALUES, to denote legal context property names and values. Based on these two sets, a context dimension represents a group of context property names that may define a distribution of certain user interface elements. A *context dimension* D is defined as a set of names $D = \{n_1, n_2, \dots, n_k\}$ to distinguish k property names such that $D \subseteq \textsf{NAMES}$. A *context state* C of a context dimension D is then represented as

$$C(D) = \{(n, V) \mid n \in D \land V \subseteq \textsf{VALUES}\},$$

where $\forall\, n \in D : \exists\, (n, V) \in C(D)$. We further define C_* as the superset of all context states defined for a distribution.

While it could be extended with other dimensions, e.g. language, browser and location, to cater for other user, platform and environment factors, XDStudio currently distinguishes only two context dimensions: $D_{\mathit{Device}} = \{\mathit{type}, \mathit{identifier}\}$ and $D_{\mathit{User}} = \{\mathit{role}\}$. Based on these two dimensions, XDStudio can already represent the context states relevant to many scenarios as subsets of C_*. For example, the meeting room could be fully described as $C_*(D_{\mathit{Device}} \cup D_{\mathit{User}})$ = { (*type*, { *"desktop"*, *"tablet"*, *"phone"*, *"other"* }), (*identifier*, {*"generic"*}), (*role*, {*"presenter"* }) }.

Further, XDStudio describes a *source interface* S as a set of interface elements $E(S) = \{e_1, e_2, \dots, e_l\}$ where for $1 \leq i \leq l : \exists\, e_i \in S$. A *target interface* T for the source interface S is then defined as a set of *interface elements* $E(S)$ and *interface adaptations* A

$$T = \{(E(S)_1, A_1), (E(S)_2, A_2), \dots, (E(S)_s, A_t)\}$$

where for $1 \leq j \leq s : \exists\, E(S)_j \in S$ and $1 \leq k \leq t : A_k \subseteq A_* = \{\mathit{remove}\}$. Currently, Remove is the only adaptation supported by XDStudio. More complex adaptations in line with requirement $R5$ could be supported by extending A_* with a new operator $transform = \{E(S)_j, CSS_j\}$. Fairly complex adaptations can be performed using CSS3 (see [17]). Keep is just an empty adaptation $\emptyset$.

With the concepts above, we can define a *distribution profile* P as a set of context states C and target user interfaces T for source interface S

$$P = \{(C_1, T_1(S)), (C_2, T_2(S)), \dots, (C_q, T_r(S))\}$$

where for $1 \leq i \leq q : \exists\, C_i \subset C_*$ and $1 \leq j \leq r : T_j(S) \subseteq S$. Our definition of distribution profile thus allows all target interfaces to be defined for one or multiple devices and users, as well as potentially other context factors. Further, each target interface part of the distribution is described by a possibly empty set of elements and adaptations of the source interface. Note that our definition supports dynamic aspects, such as a user changing role at run-time or changing device, simply by changing the context state of the device or user dimension.

While distribution is currently restricted to elements contained in the source interface, the concept could be further extended in line with requirement $R6$ to enable substitution of existing, as well as definition of new, elements and support new interactions not originally part of the source interface. This could be based on transformations described for the graceful degradation approach [3] which could substitute incompatible widgets with new ones better supported by the platform of a more constrained device. Finally, it would also be possible to address run-time requirement $R10$, i.e. to support dynamic redistribution of target interfaces if the defined context states are only partially matched or not matched at all. The first case is possible if the run-time device and user characteristics are significantly different from the ones defined at design-time. The second case is the result if not all devices and users defined at design-time are present at run-time. Implementing the above extensions for the concepts in XDStudio, and combining them with the context matching and adaptation techniques described in [15], would make it possible for XDStudio to handle both cases in the future.

Implementation

XDStudio is a web-based authoring environment which makes the tool accessible from any web-enabled device using a web browser. On the main screen, the user can load an existing web site via local or remote URL, which will be used as the source interface for distribution. This is based on techniques implemented in [17] that included a study on 50 top web sites showing good compatibility. All parts of the distributed user interface are continuously monitored and updated if required. This is based on techniques implemented in [8] using an event replaying mechanism that transmits local DOM events to the server, which in turn propagates these events to the involved clients ensuring state consistency.

The server-side components use jQuery[2] on top of the Node.js platform[3]. For storage and retrieval of data, we made use of a MySQL database which communicates with the server through a pure Node.js JavaScript client implementing the MySQL protocol[4]. To push and pull information between the server and clients, we selected Socket.IO[5], a cross-browser transport mechanism for real-time communication.

The client-side of XDStudio is implemented using HMTL5, CSS3, jQuery and jQuery UI[6] in combination with Touch Punch[7] to support touch interaction. The display and handling of interfaces in the tool itself is realised through the use of iframes. For the source interface and for every target interface, separate iframes are created. When in design mode, a transparent overlay is attached over the iframe displaying the source interface. This disables the interaction with the actual interface and allows the user to select elements without triggering the original actions. The selected elements can then be dragged and dropped on an overlay on top of the iframe for the respective target interface in simulated mode, or directly on the respective profile icon in on-device mode. If the user selects Full UI to copy the source interface, all interface

[2]http://jquery.com
[3]http://nodejs.org
[4]https://github.com/felixge/node-mysql
[5]http://socket.io
[6]http://jqueryui.com
[7]http://touchpunch.furf.com

elements are copied by reference and inserted in the target interface. The overlay for the iframe of the target interface enables the user to select elements in the same way as before and exclude them using the Remove tool. When in use mode, the overlay is detached and DOM events on interface elements are again detected and handled by the browser.

Current Limitations

Elements excluded from the target interface using Remove are hidden using CSS. Similar to [17], XDStudio favours client-side adaptation, but could equally support server-side adaptation to manipulate the DOM before it is sent to clients. Whether or not widget behaviour is replicated or modified primarily depends on the granularity of widgets and which source interface elements were included in the copy. Distributed elements are currently referenced via HTML/CSS #id. Element selection is determined based on the position of the cursor on the click event. Selection is supported for all elements with an #id, thus the closest parent element with an #id is picked upon a click. It would also be possible to use jQuery selectors[8] that do not require an #id attribute for referencing. DOM similarity matching could be added in the future to make it more robust to web site evolutions.

Support for advanced forms of interfaces is currently subject to web standards and native browser support. Most device capabilities can be accessed through the evolving HTML5 Device API. Similarly, widget behaviour is subject to many aspects, e.g. browser support, plugins, use of HTML/CSS/JS. Libraries such as jQuery Mobile could be used to render widgets in their native look on mobile devices. More advanced widget behaviour could be introduced through additional adaptation operations. There are many frameworks compatible with XDStudio such as jQMultiTouch [16] which could be used to support multi-touch interaction modalities.

STUDY DESIGN

We conducted a small user study to experiment with simulated and on-device authoring for the two scenarios presented previously. Our experiment had two primary goals. First, we aimed to identify the advantages and limitations of each mode. Second, we wanted to learn what kind of distribution tasks may be better supported using each mode. A comparative evaluation with existing tools was not considered since XDStudio provides a set of unique features for cross-device development not present in other GUI builders.

Tasks and Procedure

As presented so far, XDStudio supports seamlessly switching between authoring modes. Yet, to be able to isolate effects, we designed the experiment around three conditions in which either only the *simulated*, only the *on-device*, or *both* authoring modes were available.

In the first condition, a Windows PC was used as the central authoring device for both scenarios. In the second condition, four additional devices (an iPad, an Android phone, an HP TouchSmart, and a second screen connected to the Windows PC simulating the projector) were used in addition to the central authoring device for the meeting room scenario. For the classroom scenario, an iPad was used in addition to the central authoring device simulating the roles of the teacher and the three different student groups of the classroom experiment. In the third condition, participants had the option of switching between simulated and on-device authoring mode.

[8]http://api.jquery.com/category/selectors

The experiment started with a pre-study questionnaire collecting background information including participants' experience with user interface design. The main part of the experiment used a 2x3 design for the controlled variables scenario (meeting room, classroom) and mode (simulated, on-device, both). We rotated and counterbalanced the order of scenarios and modes to reduce carryover effects. Each of the six design tasks concluded with a post-task questionnaire where users rated available XDStudio features as well as difficulty and efficiency of the task. Apart from the subjective feedback, a logging mechanism automatically recorded all actions such as when participants switched between design and use mode, or between simulated and on-device mode when both were available. Also automatically measured was the time needed to complete each task, as well as the times spent in simulated and on-device mode in the third condition. Finally, a post-study questionnaire asked users to rate each mode and express their preference with respect to each scenario.

Hypotheses

Although the two scenarios are of similar complexity in terms of the number of distribution profiles, they may require a different approach and allow us to better characterise the features of XDStudio. The meeting room involves a device-centric scenario around a source interface that needs to be altered for different types of devices, and the target interfaces are still interconnected. On the other hand, the classroom scenario is role-centric, where the source interface needs to be split for different user roles, and the target interfaces are no longer interconnected.

Accordingly, we formulated the following two hypotheses.

- **H1.** The scenario has an effect on how simulated and on-device authoring are used.
- **H2.** Having both authoring modes available is useful and makes the design task easier and more efficient.

In particular, we hypothesised that on-device authoring would be used more in the device-centric meeting room scenario. Also, participants would rate the condition with both authoring modes available as easier and more efficient in the post-study questionnaire.

Participants

We recruited 12 participants (2 female), aged between 24 and 38 (median 28). Most had a Computer Science background, basic user interface design experience (median 3, mode 4, on a 7-point scale from 1 for Novice to 7 for Expert) as well as some desktop or mobile development experience.

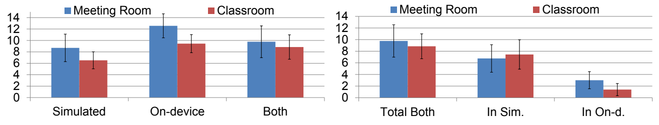


Figure 4: Average task completion times and distribution when both modes were available (in minutes; lower is better; error bars show standard deviation)

While it is still difficult to recruit participants that would consider themselves expert in cross-device development and distributed user interfaces, several of them had relevant experience in the area. Each participant owned at least two interactive devices, such as a mobile phone and a laptop, using them on a regular basis, i.e. minimum once per day. All participants were introduced to the tool and used it for the first time in the experiment.

RESULTS

To complete a task, participants were expected to create the distribution profiles required for the scenario, design the distributed user interface version for each of them, and finally test the target interfaces to verify the result. All participants successfully completed all six tasks within a 1 to 2 hour-session. We start with the analysis of the task completion times in the next section, before discussing user ratings and our takeaways from the experiment.

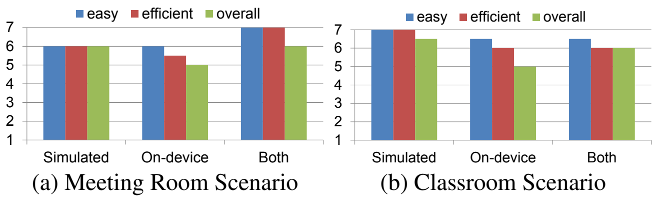
We note that the experiment was not fully controlled as it was primarily designed in a way that might lead to different authoring strategies of users to solve the tasks. For completeness, we include the results of any statistical tests we have performed. However, we are aware of the limited confidence given the small number of participants.

Times

Figure 4 shows the average time participants needed to complete each task. First, comparing times between scenarios, we can see that the meeting room demanded more time. This was not unexpected since it involves more devices to be handled and perhaps a more complex user interface, as some of the user interface elements to be distributed were initially hidden. This required switches between the design and use modes in order to interact with the user interface, toggle hidden elements, and then select them and assign them to the respective target user interface view. On the other hand, the classroom scenario involved only one device and perhaps a simpler distribution with fewer elements compared to the meeting room.

Both scenarios show similar time distributions between modes, with the simulated mode being the fastest, followed by the condition where both were available, and finally the on-device mode. On-device authoring was rather time-consuming, taking about 35% more time to complete the task compared to the simulated mode. Comparing between scenarios, the differences between the average times for the on-device and both condition seemed larger.

However, post-hoc RM-ANOVA tests showed no significant effects for scenario ($p > .09$) and mode ($p > .10$) for task



(a) Meeting Room Scenario (b) Classroom Scenario

Figure 5: Median ratings for task difficulty and efficiency as well as overall score (higher is better)

completion times. Still, an interesting observation with both modes available is that users required slightly more time for the meeting room, but much longer for the classroom, compared to when only one of the modes was available. This means that users worked more in on-device mode for the meeting room with both modes available.

Figure 4 (right) shows a breakdown of the average total execution times for the simulated and on-device modes when both were available in the third condition. Here, it is interesting that the average time spent in simulated mode is longer for the classroom than for the overall more time-consuming meeting room scenario. Users also spent twice as much time in the on-device mode when designing for the meeting room compared to when designing for the classroom scenario. This may mean that on-device authoring is more beneficial for device-centric rather than role-centric scenarios.

While post-hoc RM-ANOVA tests confirmed significant differences for mode ($p < .01$), the differences between scenarios were not significant ($p > .61$). Next we analysed the subjective feedback obtained from users.

User Feedback

This section analyses user ratings from the post-task and post-study questionnaires. In the post-task questionnaire, participants were asked how easy they found it to complete the task and how efficient they felt. In addition, they were asked to rate usefulness, ease of use and effectiveness of available features including profile creation tools, direct manipulation tools, the ability to switch between design and use modes, as well as between simulated and on-device authoring. The post-study questionnaire asked users to rate the authoring modes for the two distribution scenarios, which one they would prefer for each scenario, what other features they would like the tool to support, and if they have other comments. Ratings were collected using a 7-point Likert scale from 1 for the most negative and 7 for the most positive score.

Authoring Modes

Figure 5 shows the median ratings users gave after each task for the meeting room scenario and the classroom scenario, respectively. For the meeting room, the simulated and both condition have the same overall score, but users found it easier to do the task and felt more efficient when both simulated and on-device modes where available. The on-device authoring mode received a relatively lower score. However, for the meeting room scenario users still found it fairly easy to do the task, giving a slightly higher score compared to the simulated mode. For the classroom scenario, the overall grade of the simulated mode exceeds the other two, leaving the both condition in the second place and on-device last. The same applies for how easy participants found completing the task and how efficient they felt under the three conditions.

Post-hoc analysis using Friedman tests found no significant effect for the authoring mode ($p > .29$). Also the differences for task difficulty were not statistically significant ($p > .15$). However, the differences in task efficiency showed a significant effect ($p < .01$).

Follow-up pairwise Wilcoxon Signed Ranks Test of whether scenario has an effect confirmed that participants felt significantly more efficient in the classroom scenario ($p < .04$), where simulated authoring felt most efficient compared to on-device authoring ($p < .02$) or when both modes were available ($p < .03$). For the meeting room scenario, users felt significantly more efficient in the both condition compared to the on-device authoring ($p < .03$). Here, the other differences were not significant. However, despite this subjective feedback about efficiency and the slight edge for the classrom scenario, as stated earlier, there were no significant differences in mean task performance times between scenarios.

Finally, the average user ratings in terms of whether users found alternating from one mode to the other useful, easy and effective were all fairly positive (all between 5 and 6). While the meeting room received slightly higher scores compared to the classroom scenario, the differences were not significant.

Design and Use Modes

The feature for switching between design and use mode obtained fairly positive ratings for useful, easy to use, and effective. While a Friedman test found no significant differences between scenarios, the feature received slightly higher scores in the classroom (all medians 6) compared to the meeting room scenario (all medians 5). Given that not all elements were always visible in the meeting room, users repeatedly had to go into use mode to switch tab panes in order to view target elements, and then return to design mode and assign them to the respective profile. This demanded effort that most users stated they would preferably avoid. In contrast, the classroom scenario required fewer switches, and the use mode was mostly employed to check the final target user interfaces.

Distributon Profiles

As for distribution profiles, users rated the creation of devices and roles as well as the selection of profiles for designing the distribution. On average, users were very positive about these three features giving at least a rating of 6 for useful, easy to use and effective.

Four participants commented that they preferred on-device authoring when it comes to profile creation, as information about connected devices saves times and effort. In this case, manual creation would only be necessary for devices that are not available at design-time.

Other Design Tools

The ratings collected for element selection and drag-and-drop as well as the Full/Empty UI insertion feature were similarly positive for useful (all medians above 6), easy to use (all medians 5) and effective (all medians above 5.5). Here, it was interesting to see how the tools were used for the two scenarios. In the meeting room scenario, users found it more efficient to design using element selection and drag-n-drop as opposed to starting from the full user interface version, where many hidden elements had to be first uncovered through toggling visibility in the use mode, and then removed. For the classroom scenario, creating distributions using the Full UI tool and then removing not required elements was preferred to manually dragging and dropping each required element, which seemed less efficient to users. But also drag-n-drop was used for corrections when required elements had mistakenly been removed after the Full UI tool was used to copy the source interface.

CONCLUSION

Based on these results and our observations during the experiment, we can confirm H1 in that the scenario had a strong effect on how simulated and on-device authoring were used. While average task times were not significantly different, participants used on-device authoring more in the meeting room scenario, making it better suited to device-centric scenarios. However, in cases where the distribution only requires small changes, e.g. excluding certain elements from the source interface, or if fewer devices are involved, simulated authoring may still perform better.

Also H2 can be accepted. Our experiment separated simulated and on-device authoring, but based on the results, we can now say that it is beneficial to have both modes available. One participant explicitly stated that they would prefer to do the design using the simulated mode, but also see the changes on the connected target devices, which was only available in the on-device mode. While the differences between authoring modes and ease of design were not statistically significant, users felt significantly more efficient with both modes available in the meeting room scenario.

Authoring Strategies

Based on our observations and the log files produced during the experiment, we were able to extract three main themes of how participants commonly authored cross-device user interfaces using XDStudio.

All Profiles First, Widget-driven Distribution

Although participants were completely free in how they would solve each task, they commonly used the approach of first creating all distribution profiles before designing the target user interfaces. However, rather than designing the distribution for one profile after the other, users often focused on the separate views of the source user interface and distributed one view after the other. This seemed especially efficient in the meeting room scenario, where some target user interfaces shared the same user interface elements such as the projector and the “presenter” tablet.

Use Mode to Develop Overall Strategy

Another common behaviour of our participants was to initially switch to use mode to first develop an overall strategy of the distribution. This was interesting because participants were first introduced to each source interface, giving them time to explore it before they were given instructions about the task. Use mode was also commonly used at the end of tasks in which on-device mode was not available in order to verify the distribution.

On-device Mode for Testing and Final Check

We also noticed that when having the option to alternate between simulated and on-device authoring, the majority of participants (8 out of 12) performed the task in at least one of the scenarios using primarily the simulated mode. They then switched to the on-device mode mostly to check how the target user interface was actually displayed on each device.

Limitations

Finally, considering the nature of the scenarios, we note that the emphasis was on designing for the given distribution and less on experimenting with different designs of the user interface and distribution according to different goals. Therefore, all participants focused on performing the design tasks according to the task specification and only shortly tested the target user interfaces to verify the tasks were fulfilled.

To thoroughly investigate how users would interact with the tool to explore different distributions, as well as testing and debugging them, additional tasks would have been required. While we believe this would lead to interesting findings for XDStudio, we refrained from doing so in our initial user study due to the already long sessions of one to two hours.

Acknowledgements

The authors would like to thank Pierre Dillenbourg for helpful discussions on this research. Theano Mintsi was supported through the Act "I.K.Y Bursary Program process with individualized assessment, academic year 2011–2012" by the O.P. resources "Education and Lifelong Learning" of the European Social Fund (ESF) and the NSRF 2007-2013.

REFERENCES

1. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. A Unifying Reference Framework for Multi- Target User Interfaces. *IWC 15* (2003).
2. Dillenbourg, P., and Hong, F. The mechanics of cscl macro scripts. *CSCL 3*, 1 (2008).
3. Florins, M., and Vanderdonckt, J. Graceful Degradation of User Interfaces as a Design Method for Multiplatform Systems. In *Proc. IUI* (2004).
4. Genaro Motti, V., Raggett, D., Van Cauwelaert, S., and Vanderdonckt, J. Simplifying the Development of Cross-platform Web User Interfaces by Collaborative Model-based Design. In *Proc. SIGDOC* (2013).
5. Ghiani, G., Paternò, F., and Santoro, C. Push and Pull of Web User Interfaces in Multi-Device Environments. In *Proc. AVI* (2012).
6. Ghiani, G., Paternò, F., and Santoro, C. Interactive customization of ubiquitous web applications. *VLC 24*, 1 (2013).
7. Gjerlufsen, T., Klokmose, C. N., Eagan, J., Pillias, C., and Beaudouin-Lafon, M. Shared Substance: Developing Flexible Multi-Surface Applications. In *Proc. CHI* (2011).
8. Husmann, M., Nebeling, M., and Norrie, M. C. MultiMasher: A Visual Tool for Multi-Device Mashups. In *ICWE Workshops* (2013).
9. Jetter, H.-C., Zöllner, M., Gerken, J., and Reiterer, H. Design and Implementation of Post-WIMP Distributed User Interfaces with ZOIL. *IJHCI* (2012).
10. Limbourg, Q., and Vanderdonckt, J. Multipath Transformational Development of User Interfaces with Graph Transformations. In *Proc. HCSE*. Springer, 2009.
11. Lin, J., and Landay, J. A. Employing Patterns and Layers for Early-Stage Design and Prototyping of Cross-Device User Interfaces. In *Proc. CHI* (2008).
12. Melchior, J., Vanderdonckt, J., and Roy, P. V. A Model-Based Approach for Distributed User Interfaces. In *Proc. EICS* (2011).
13. Meskens, J., Loskyll, M., Seibetaler, M., Luyten, K., Coninx, K., and Meixner, G. GUIDE2ux: a GUI Design Environment for Enhancing the User eXperience. In *Proc. EICS* (2011).
14. Meskens, J., Vermeulen, J., Luyten, K., and Coninx, K. Gummy for Multi-Platform User Interface Designs: Shape me, Multiply me, Fix me, Use me. In *Proc. AVI* (2008).
15. Nebeling, M., Grossniklaus, M., Leone, S., and Norrie, M. C. XCML: Providing Context-Aware Language Extensions for the Specification of Multi-Device Web Applications. *WWW 15*, 4 (2012).
16. Nebeling, M., and Norrie, M. C. jQMultiTouch: Lightweight Toolkit and Development Framework for Multi-touch/Multi-device Web Interfaces. In *Proc. EICS* (2012).
17. Nebeling, M., Speicher, M., and Norrie, M. C. CrowdAdapt: Enabling Crowdsourced Web Page Adaptation for Individual Viewing Conditions and Preferences. In *Proc. EICS* (2013).
18. Paternò, F., and Santoro, C. A Logical Framework for Multi-Device User Interfaces. In *Proc. EICS* (2012).
19. Paternò, F., Santoro, C., and Spano, L. MARIA: A Universal, Declarative, Multiple Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments. *TOCHI 16*, 4 (2009).