Understanding the Challenges of Designing and Developing Multi-Device Experiences

Tao Dong
Google Inc.
Mountain View, CA, USA
dongtao@acm.org

Elizabeth F. Churchill
Google Inc.
Mountain View, CA, USA
churchill@acm.org

Jeffrey Nichols
Google Inc.
Mountain View, CA, USA
jwnichols@google.com

ABSTRACT

As the number of computing devices available to users continues to grow, personal computing increasingly involves using multiple devices together. However, support for multi-device interactions has fallen behind users' desire to leverage the diverse capabilities of the devices that surround them. In this paper, we report on an interview study of 29 designers and developers in which we investigate the barriers to creating useful, usable, and delightful multi-device experiences. We uncovered three key challenges: 1) the difficulty in designing the interactions between devices, 2) the complexity of adapting interfaces to different platform UI standards, and 3) the lack of tools and methods for testing multi-device user experiences. We discuss the technological and business factors behind these challenges and potential ways to lower the barriers they impose.

Author Keywords
Multi-device experiences; multichanneled services; crossmedia services; cross-device interactions; design issues; developer experience, interviews

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

INTRODUCTION

The ability to seamlessly connect multiple devices of varying screen sizes and capabilities has always been an integral part of the vision for distributed user experiences and Ubiquitous Computing. Recent studies [7,14] have shown that this future vision is beginning to arrive, and that many users are now engaging with multiple devices in parallel everyday. Use cases range from the mundane and simple, such as checking the translation of a word in a subtitle while watching TV, to more complex and long-form tasks, such as following instructions on a phone for installing and configuring software on a computer.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored.

Copyright is held by the owner/author(s).
DIS 2016, June 04-08, 2016, Brisbane, QLD, Australia
ACM 978-1-4503-4031-1/16/06.
http://dx.doi.org/10.1145/2901790.2901851

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs International 4.0 License

Unfortunately, this same research shows that these multi-device use cases are rarely supported by software and that users must act as the bridge connecting their devices. This can substantially increase cognitive load for the user and greatly increase the difficulty of even relatively simple tasks. The lack of high quality multi-device user experiences despite the availability of the hardware devices that should enable them implies that there exist design and development challenges that are not yet fully understood. To better understand this, we chose to conduct a study to answer two research questions:

- *What barriers and challenges are there to creating useful, usable, and delightful multi-device experiences?*
- *What tools and methods would be helpful to simplify the design and development of such experiences?*

Our focus is to understand what factors might be complicating and undermining the work of the designers and developers who are trying to bring multi-device experiences to users. Our long-term goal is to inform the design of tools that aid in the design and development of such experiences.

To this end, we conducted in-depth interviews with 29 professionals who are actively designing or building multi-device user experiences, revealing a number of common challenges that are unique to the design and development of such experiences. In this paper, we present the three most critical challenges they discussed:

1. The difficulty in designing interactions between devices
2. The complexity of adapting user interfaces to different platform UI standards
3. The lack of tools and methods for testing multi-device user experiences

These three challenges affect different activities in building a multi-device experience. The first challenge complicates the functional design of a system, while the second challenge burdens visual and interaction design. The last challenge can cause the implementation of the experience to be costly and hard to manage.

Our work builds on past research (e.g., [4,11]) that has predicted the contemporary landscape of multiple, potentially connected devices, and forewarned of some of the difficulties inherent in creating seamless connected device experiences, such as maintaining UI consistency across devices and eliciting user feedback on early prototypes. As this multi-device computing environment has now become a reality, it is important to again consider these issues and identify which challenges have actually arisen and which have turned out to be less important than originally anticipated.

The rest of this paper is organized as follows: We first present an overview of related work on the design of multi-screen, multi-device experiences and the unique challenges of designing for such products. We then describe our research method and processes, and present detailed results that illustrate three key challenges identified. Finally, we discuss the technological and business factors behind these challenges and potential ways to lower the barriers they impose.

RELATED WORK

Our research is related to and informed by two bodies of work: studies on multi-screen and multi-device design principles, and studies seeking to understand the challenges faced by designers and developers of such multi-screen and multi-device user experiences. To contextualize these two bodies of work, we first introduce two main types of multi-device experiences identified in prior work. We have found this distinction helpful in understanding different design goals and the challenges associated with them.

Multichanneled Services and Crossmedia Services

Wäljas et al. identified two main types of multi-device experiences: multichanneled services and crossmedia services [19]. Though it is possible for a service to be both multichanneled and crossmedia, we found this distinction useful for analytical purposes.

Multichanneled services allow the user to access the same service via different kinds of devices and platforms (i.e., channels) [6,18]. The features available on different devices might differ due to input and output constraints and usage expectations, but the core functionalities are accessible from any of the supported devices. An example of a multichanneled service is the music streaming service Spotify, which offers largely the same set of features via its Web-based player, Mac application, iOS app, and Android app. The user's data between these channels are synchronized to allow the user to move between these channels to consume the service.

Crossmedia services split functionality across different devices according to those devices' strengths and affordances [19]. A commercial example of a crossmedia service is Google Chromecast[1], which allows the user to use a mobile device or a Web browser as the remote control for a large-screen viewing device, such as a TV. In this case, each type of device is used for its strength, and functionality is largely non-overlapping between them. The experience as a whole would not be possible without at least one of each type of device being present and functioning.

Multi-device Design Principles

To understand multi-device designers' challenges, it is important to understand the unique design requirements of multi-device services in the first place. This will help us to understand the gap between what designers can achieve with the resources available and the ideal user experience they want to provide. Multichanneled services and crossmedia services have different design and development considerations and thus we consider them separately.

For multichanneled services, inter-device consistency is the most fundamental principle. Extending the conventional concept of usability to multi-device systems, Denis and Karsenty [3] coined the term *inter-usability,* which refers to the overall ease of switching to other devices for a given functionality. Through interviewing 10 users who used multichanneled services such as emails, diaries, and address books, they identified two main dimensions of inter-usability: *knowledge continuity* and *task continuity*. Knowledge continuity allows the user to apply her existing knowledge about a service learned from using a previous device to a new one, while task continuity allows the user to easily resume the task he previously worked on when he switches devices. Denis and Karsenty further argued that these two types of continuities could be achieved by implementing different levels of inter-device consistency.

The key issue for crossmedia services is the question of how to distribute functionalities to different devices in the system. Segerståhl [15] conducted a field study of a fitness tracking system that consisted of a wearable activity tracker and a Web service. Based on the results, Segerståhl argued for the benefits of specializing different devices within the system with different functionalities to reduce the complexity of individual components of the system. However, she also warned that such distribution of functionality must match the structure of the target activity and maintain a certain degree of flexibility and feature redundancy (which she referred to as functional modularity) because human activities are constantly fluctuating.

The temporal dimension of multi-device use is also helpful in understanding the multi-device design space. In a diary study of multi-device use, Jokela et al. [7] found that both sequential use and parallel use of multiple devices were practiced by users. Similarly, in the 4C framework proposed by Sørensen et al. [16], the authors make a distinction between sequential use and simultaneous use. They emphasize data synchronization and migration of the user's activity state for sequential use and stress the importance of making sure different devices play to their individual strengths and complement one another for simultaneous use.

[1] Chromecast. https://www.google.com/chromecast

In search of an overarching design principle for multi-device experiences, Wäljas et al. [19] propose the notion of service coherence, based on insights gathered from a diary study of 3 different multi-device services, including both multichanneled services and crossmedia services. They define coherence as follows: "Coherence is influenced by composition, flow of interactions and content as well as continuity." In a similar vein, Kim et al. [8] argue that an ideal multi-device experience should have "locally optimized consistent UI & globally unified coherent UX." Levin offers a more holistic approach in her 3C framework [9], which argues that *consistent*, *continuous*, and *complementary* approaches should be considered as building blocks of a multi-device ecosystem rather than mutually exclusive paths of design.

Multi-Device Prototyping and Development Issues

Previous work has also examined the challenges inherent in prototyping and building multi-device user experiences.

To motivate a pattern-based cross-platform design tool, Lin [11] interviewed nine UI designers who worked on applications targeting two or more different types of devices, including desktop computers, PDAs, and WAP phones. The findings from his interviews highlight the difficulty of maintaining UI consistency across devices, especially the consistency of menu order, terminology, and colors and graphics. Furthermore, Lin noted the lack of design tools specialized for handling multiple devices and the scarcity of cross-device design patterns despite designers considering them to be useful.

Around the same time (pre-2006), Dow et al. [4] conducted an interview-based study with eleven designers who had experience "designing and prototyping off-the-desktop applications," which included multi-device experiences but also ubiquitous computing systems in general. They identified three challenges, including design tools inadequately supporting communications between different design roles, designers lacking knowledge about the capabilities and constraints of new hardware, and technical difficulties in producing Ubicomp prototypes. Furthermore, they argue for creating tools that can enable designers to employ multiple representations, including storyboards, diagrams, and simulations, to examine and express application design ideas.

More recently (pre-2011), Antila and Lui [1] interviewed seventeen professionals whose work included "components, which are interconnected with some level of measured usability." Based on those interviews they identified four challenges. First, it was difficult to address implementation constraints imposed by specific platforms at the early stage of a project, because the tools used by designers and the tools used by developers lacked integration. Second, some domains, such as healthcare, restricted deployment of certain technologies for organizational reasons. These restrictions can negatively affect the continuity and consistency of cross-platform experiences. Third, it was difficult to get user feedback on multi-device experiences until a functional prototype can be deployed in the field due to the lack of research frameworks and methods. Last, targeting multiple platforms/devices was complicated because of the tension between the assumed need for a unified look-and-feel across devices and the inherent differences in the capabilities and user interaction metaphors of each device.

Our research extends this prior work in three important ways:

1. Prior work focused mainly on the (still important) challenge of designing UIs that are consistent across devices that have very different input and output capabilities (e.g., [1,11]). We found this challenge was complicated by the additional tension between following native UI standards imposed by platform vendors (e.g., Google's Material Design[2] and Apple's Flat Design[3]) and maintaining consistency of design across platforms.
2. Prior work (e.g., [1,4]) noted that practitioners have difficulties getting useful feedback without deploying functional prototypes in the field due to the lack of research frameworks and analysis methods to explore a realistic cross-section of contextually appropriate usage scenarios. We extend this finding by unpacking the design complexity imposed by the many, often unanticipated ways devices can interact with one another in a multi-device experience.
3. Prior work (e.g., [1]) focused on the difficulty of user testing for multi-device experiences. Our work reveals that software testing is equally difficult. Functional testing, compatibility testing, and GUI testing are challenging for multi-device developers and represent an important barrier to development. Our interviews shed light on the unique challenges of conducting these tests in the implementation process of multi-device experiences.

STUDY DESIGN

In order to identify challenges related to designing and developing multi-screen and multi-device experiences, we conducted semi-structured interviews with 29 professional designers and developers who are currently building multi-device experiences.

Participants

Participants were recruited from an existing database of prospective participants maintained by our organization, and also through mailing lists, personal referrals, online development communities, and snowball sampling.

[2] Material Design. https://www.google.com/design/spec/material-design/

[3] iOS Human Interface Guidelines: Designing for iOS. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/

Of our 29 participants, 16 were UX/UI designers, 11 were software developers, and 2 were product managers. Our participants came from 13 organizations, but about half were affiliated with one large technology company that offers several popular multichannel and crossmedia products. Twenty worked on multichanneled experiences, and 9 worked on crossmedia experiences. The dominance of multichanneled experiences in our sample reflects the nascent nature of crossmedia experiences in the market.

Interview Procedure

All but three of our participants were interviewed in person; the remainder through video conferencing tools. The interviews were semi-structured and issues were discussed on a number of themes: the cross-device aspects of the participant's current and past projects; tools currently and previously in use; preferred, prescribed and discarded workflows; examples of adapting UIs to different platforms and form factors; aspects of prototyping and testing designs; within-team and cross-organization and development function communication issues; and attitudes towards automating certain aspects of the design process. Each interview lasted about an hour.

All interviews were audiotaped, summarized, and partially transcribed. Affinity diagrams were created to organize the data and identify common and key themes. Analytic memos on key themes were generated immediately post-interview and iteratively during analysis to provide a foundation for synthesizing the viewpoints of different participants.

RESEARCH FINDINGS

Our research identifies a number of challenges in designing and developing multi-screen and multi-device user experiences, confirming that the nature of multi-device interactions have brought about unique problems for designers and developers. Although a number of themes are apparent in our data, in this paper we present the top three challenges that have deep implications for the practice of design and development of multichanneled and crossmedia user experiences. Table 1 gives an overview of the challenges identified and explored.

Designing Interactions between Devices

Some of our participants worked on features that allow multiple devices to collaborate either in sequence or in parallel to help the user accomplish a task. Sequential use of multiple devices often emerges naturally when a multichanneled service provides the ability to synchronize data across devices, while parallel use is usually more deliberately designed by assigning different roles to different devices involved in an interaction. Both types of interactions between devices bring about new design challenges as we describe below.

Maintaining Consistent Information Architecture

From a design perspective, maintaining consistent information architecture is critical to help users achieve task continuity and knowledge continuity when they switch to a different device that might have a distinct form factor or different input and output constraints. P16, a designer working on a travel search application said his team was aware that some users would plan their trip across several devices in multiple sessions, as he described:

"You might start [planning a trip] at work on desktop. You might do more searches along the subway on your phone. And then in the evening with your tablet."

In order to make this kind of multi-session, multi-device activity as cohesive as possible, P16 emphasized the

Activities	Identified Challenges
Functional Design	**The difficulty in designing interactions between devices** • *Maintaining consistent information architecture* • *Handling task continuity with uncertain user intentions* • *Designing and communicating complex conceptual models and business logic*
Visual & Interaction Design	**The complexity of adapting user interfaces to different platform UI standards** • *Business biases and unrealistic design ideals* • *Fuzzy boundary between platform standards and product identities* • *Unclear cost of deviating from the standard* • *Unverified assumptions about multi-device use*
Implementation	**The lack of tools and methods for testing multi-device user experiences** • *Too many distinct devices to test* • *Inadequate emulators* • *Interdependency between components running on different devices* • *Difficulty of automating UI tests*

Table 1. Selected Challenges Grouped by the Activities in the Design and Implementation Process

importance of maintaining consistent information architecture across devices. In particular, he explained:

"When you transition between devices, 'Oh yeah! That was the thing I was looking at before.' So really as much as the layout as we can keep the same and consistent I'd love to keep it the same and consistent. But the most important thing to me is that we use the same words to describe the same things and the flow of information in the unit is consistent."

An additional challenge brought by multi-device computing to information architecture design is that some of the most familiar building blocks of information architecture start losing their meanings in a multi-device world. Designer P14 provided an example of this in an e-book app that attempts to always open to the last page a user was reading across different devices:

"The challenging thing is because of the different resolutions and screen sizes of different devices... you might have 1,000 characters here [a large-screen Android phone]. On the iPhone, you'll have 600 characters. So making sure that when you open it up, you map exactly to the right location. Page numbers don't really mean anything."

Handling Task Continuity with Uncertain User Intentions

Having consistent information architecture across devices is important to support task continuity, but that is often not enough. Designers sometimes find themselves uncertain about user intentions and making uninformed guesses when they wanted to create continuous user experiences across devices.

For instance, P23 worked on an application designed for sales professionals that has both a mobile version and a desktop version. The app supports two sequential use cases. In one case, users can save sales leads on their smartphones when they are in the field and then continue researching or following up on these leads on their computers when they get back to the office. In the second case, users can collect potential sales leads on their desktop computers and then retrieve related information before or during customer meetings on their smartphones.

Eager to harness the potential of multi-device computing, P23's team initially wanted to implement a "dream" scenario, as they referred it to, in which the mobile app would automatically present sales leads the user saved in the desktop version at the moment the user opens the mobile app. They hoped that this feature would help the user re-access saved information and provide a continuous experience. However, it proved to be more complicated than they thought, as P23 explained:

"Let's say you send it but you never opened up the app and then like three days later you opened up the app. It's telling you to go to this thing that's no longer relevant."

P23's account reflects the difficulty to answer several critical design questions due to the lack of tools and frameworks for designing such cross-device experiences. The first question they could not answer, in our view, was: what is the user's intention when he/she saves a sales lead on a device? The action of saving a lead does not necessarily imply an unfinished task that the user intends to get back to. Though they could have requested the user to specify his/her intention, but P23 said they worried it would make it "a heavyweight action." The second question was: what is the user's intention when he/she opens the app later on a different device? The time elapsed between these two sessions might provide some signal but it seemed not enough. The last question was: how proactively the system should help the user to get back to a saved task? Will that reminder get in the way of another task? Without a means to properly answer these three questions, implementing that "dream" scenario would be "sometimes useful, sometimes annoying [to the user]," said P23. His team ended up just providing the user with a separate section of saved leads, which, he admitted, did not take full advantage of the multi-device ecosystem but was a safe option.

Designing and Communicating Complex Conceptual Models and Business Logic

Another aspect of the challenge was related to the complex conceptual models and business logic governing crossmedia experiences. All apps embody business logic in that they take users through a flow, but in a crossmedia experience, business logic also specifies how and when a device should respond to another. It turned out to be quite difficult for designers to understand, inspect, and communicate how multiple devices would work together under different circumstances.

To designers and developers, it is particularly difficult to grasp the application behaviors in unusual or infrequent scenarios. Nevertheless their users may find themselves in such scenarios–and it is well known that a single negative experience can sour a user's relationship with a product. P26, who was familiar with issues in developing Chromecast apps, said that properly handling how the mobile client (i.e., the sender app in a typical Chromecast experience[4]) disconnects from and reconnects to the Chromecast device was one of the hardest problems for app developers. If they were not careful about the business logic they encoded into their apps, they might put their users in awkward positions. He told us an example:

"If you don't get it right, then you leave for the day, your kids watching whatever when they come home from school. You come home. Your device sees that network and reconnects to the network, and then reconnects that app [to

[4] Chrome Sender App Development. https://developers.google.com/cast/docs/chrome_sender

Chromecast] potentially. And you blow away what someone else is watching."

In this particular case, it would be easy to attribute the fault to the developer who made its app overly aggressive on reconnecting itself to the Chromecast device. But there are two issues inherent to designing crossmedia experiences. The first issue was the inability to determine the user's intention when they get disconnected from one of the devices involved in the experience. The father in this example made a so-called "implicit disconnect" by going out of the range of the WiFi network. It was unclear to the mobile client whether he wanted to reconnect to the Chromecast later or wanted to end the experience. The second issue was that it could be onerous for the product designer to walk through all the edge cases to understand the implications of the business logic he/she designed for the application. The challenge of designing sound business logic for applications involved in a crossmedia experience was crystallized in the analogy P10 made:

"When you think about design involving all these devices, it's a conversation...How do you ping those things in a way that it's an ongoing conversation, and they speak to one another in a synced way and in a logical way?"

Designing business logic for crossmedia experiences is hard, as is communicating that logic. One example is the Smart Lock[5] technology shipped with newer Android devices and Chromebooks. One of the features offered by the technology is to automatically unlock a device based on its proximity with another device that belongs to the same user and is currently unlocked. For example, when your phone is near your Chromebook and your phone has been unlocked, Smart Lock will automatically unlock your Chromebook. P10, a designer who were familiar with the technology, told us some users found it difficult to understand how it worked and its security implications:

"We need to do a better job at explaining to people the overall mental model. I think this idea of unlock phone and the unlocked phone represents you and your phone and it lets us know that you're there… Also just understanding what unlocks what and how the system works."

To summarize, participants identified three main reasons why anticipating and designing multi-device interactions could be challenging. First, the information architecture needs to be consistent across different form factors of devices to support task continuity. Second, it is hard to provide explicit support for task continuity across devices, because the user's intention is often unclear. Third, as devices start to collaborate in a variety of ways and under many different circumstances, it is hard for the designer to anticipate all the edge cases and design application behaviors appropriate for them.

[5] Google Smart Lock. https://get.google.com/smartlock/

Adapting User Interfaces to Platform UI Standards

Multichanneled experiences need to be accessible from different computing platforms, which often have distinct UI standards and design languages imposed by the platform's vendor, e.g., Apple's Flat Design, Google's Material Design, and Microsoft's metro-style UI. Though these UI standards are useful for platform vendors to enforce a coherent user experience within their respective device ecosystems, they also have become a source of contention, as multichanneled experiences often need to simultaneously pursue two different kinds of consistencies. For example, an application is expected to be consistent with the UI standard of the platform it runs on. Furthermore, an application is expected to be consistent with its sibling applications on other platforms. How should the designer accommodate the differences between UI standards while maintaining the consistency between different channels of the same service?

Based on our interviews, many experienced designers have taken a pragmatic approach to address this question. For example P10 expressed such a view as follows:

"The native guidelines need to be respected, but there is still a lot of room for freedom and creativity within your specific product."

This appears to be a sensible way to strike a balance between the potentially conflicting requirements of platform standards and product identities. However, a closer look at our interview data reveals that achieving that balance involves a tremendous amount of work to defend design decisions against business biases, refrain from pursuing unrealistic design ideals, and clarify ambiguities between platform standards and product identity. In addition, this approach is also loaded with risky assumptions about how users may cross ecosystem boundaries. We address each of those issues below.

Business Biases and Unrealistic Design Ideals

Though none of the designers we interviewed believed that every aspect of UI design should or can be consistent between platforms, some of their business stakeholders apparently did. For example, P07 lamented that business stakeholders often prefer consistency across platforms without understanding the reasons that differences may be necessary. He said:

"Way too often they [the stakeholders] are asking wrong questions. They're usually asking, 'why are they not the same' instead of 'why are they different?' Because usually there is a reason to be different."

To pursue such design ideals can lead to a compromised user experience. For instance, P13 said:

"We were very resistant to [UI differences between apps] at the beginning. We thought, well because on one screen you scroll this list horizontally, then on the Web you should scroll horizontally as well. We tried for months to make that work until we said no one wants to scroll horizontally on

the Web...We can't stick to a rule that compromises the quality and the experience for the user."

Fuzzy Boundary between Platform Standards and Product Identities

As stated above, many designers wanted to follow the UI standard of the platform but they also wanted to maintain the consistency of those aspects of the design that defined the character of their products across platforms. However, there was little consensus nor any deep rationale for what needed to be consistent across platforms.

When we asked about consistency choices, some participants made reference to very abstract motivations and concepts. For example, P09 talked about keeping the "spirit" of the product consistent across platforms while not worrying about specific UI elements. What "spirit" is was not defined, but his strategy was to stick to a set of design principles for the particular product. Similarly, P10 argued that consistency across platforms should apply to the differentiating factors in your product. P02 and P16 were more specific about visual design and information architecture, respectively, but their answers seemed highly dependent on the project they were thinking about at the time. For instance, P16's emphasis on information architecture was related to a travel planning application that allows multi-session and multi-device searches.

Unclear Cost of Deviating from the Standard

Though our participants generally were able to find ways to accommodate both platform UI standards and product-specific designs, they did occasionally run into conflicts, in which they found little guidance to help them properly evaluate the tradeoff. For example, P12 described a situation where the brand font of their product, a shopping app, was different from the system font of Android. They ended up giving up the brand font to make the UI look closer to other native apps, because the team believed that Android users would leave negative reviews if they saw a non-standard font. However, he was never sure whether that was the right decision:

"We respected the font for Android. We used Roboto. I might be wrong... On one hand, you are alienating your brand. On the other hand, you are alienating your users."

Unverified Assumptions about Multi-Device Use

Throughout our interviews, we found that the perceived importance of respecting native UI standards among many designers might be based on unverified assumptions about how users use multiple devices in the real world. Relying on such assumptions can be especially risky in the fast-changing area of multi-device interaction design.

Some participants provided seemingly contradictory accounts about the assumptions guiding their design decisions. For example, P14 initially said she believed that users rarely cross ecosystem boundaries (e.g., an iOS user will rarely use Android devices, and vice versa). Following this conviction, she said,

"It's just a guiding principle I think [that] we don't violate the OS native experiences."

However, she was also aware of the possibility of users sharing devices that have different platforms in a household, as she later said:

"And also if you're in a family, you guys are sharing the same library, you may not necessarily have the same OS or device."

The problem was that neither assumption had been validated through user research. Leaving such design decisions to intuition and guesswork is problematic when between-platform consistency can sometimes conflict with within-platform consistency.

To summarize, designers often find themselves managing a delicate balance between respecting platform UI standards and maintaining the identity and coherence of the product across platforms. Their practice is complicated and undermined by the lack of information and methods to assess tradeoffs between different kinds of consistency.

Testing Multi-device User Experiences

Testing multi-device experiences is much more complex than testing single-device experiences. Our participants reported a number of issues, including too many distinct devices, inadequate emulators, interdependency between components running on different devices, and the difficulty of automating UI tests in cross-device systems.

Too Many Distinct Devices to Test

Since the original iPhone launched in 2007, mobile device models have proliferated. According to a report [5], there were 24,000 distinct models of Android smartphones worldwide in 2015. It is infeasible for developers to sufficiently test their applications to cover all the possible devices in use.

A common strategy to deal with this explosion of devices is to test with a handful of representative devices from each category of form factor, and then test with a few known "trouble makers." For example, P18 said:

"Everyone has a small device, a medium-sized device, and a tablet. We also have a lot of other tablets to pass around... We use some little devices that are particularly known crashers."

Some modifications made by device vendors add to the burden of testing. For instance, P18 told us:

"They [a large technology manufacturer] were using the classic browser on Jelly Bean, even though Jelly Bean was supposed to use the Chrome-based browser."

One of the consequences of having to test on many real devices is that UI design issues are often discovered late in the development process. Designer P14 told us that on-device testing was usually done during the coding process and sometimes as late as in the QA stage. She said:

"Engineers can't catch everything for every device, then QA test and say, 'this is not working on this device,' and they'll take screenshots and file bugs."

Unable to discover design flaws on certain devices early on can lead to difficult decisions and compromises on user experience. For example, P14 said:

"What really is a bummer is that we have to make a design compromise, because it's not gonna work on this one set of devices. So it looks crappy on 40% of devices, because we have to fix for the other devices. Then I looked at our metrics, and say, 'Okay. Well, you know people using that on this OS and on that device is less than a greater N, and then I just have to make a call."

For smaller app makers, the problem is likely to be more acute, since they often cannot afford buying many different devices or hiring dedicated QA engineers.

Inadequate Emulators

Emulators are supposed to help developers examine their apps when the target physical devices are not available, but the developers we interviewed rarely used emulators. The main reason was that they were too slow to be useful. Emulator makers are fully aware of this and they have started making dramatic improvements to the speed of emulators [17].While a faster emulator is certainly helpful for many developers, it does not solve the problem for those who need to access low-level hardware features for inter-device communication and data transmission.

We learned of one such problem from P20, an engineer who worked on a system that uses Bluetooth and other radio technologies to transmit data from one device directly to another. He and his colleagues had to test with more than 120 different devices to make sure that their system would work well, because different devices often implement radio technologies differently. His team could not simply test their system on emulators because:

"All they [the emulators] will be doing is forking out to whatever Bluetooth stack was installed on your desktop. For that, you are not testing what actually gonna happen on the phone, you're just testing some arbitrary Bluetooth stack. Maybe it works great, [but] that still doesn't tell you anything about the devices. Maybe it doesn't work great, but your debugging problem might not exist in the real world. So it's a problem. ***The emulator doesn't emulate hardware.*** *[emphasis ours]"*

The key deficiency of emulators, as P20 pointed out, is that they are designed to mimic the software environments of mobile devices rather than different devices' hardware properties and features. Thus, they are often not helpful for testing direct device-to-device communications, which are highly dependent on the compatibility and performance of the hardware.

Interdependency between Components Running on Different Devices

Another factor that makes testing crossmedia experiences challenging is the interdependency between software components running on different devices. How to test each component independent of the others is an important productivity question for developers who are building these components in parallel. Being in this situation when his company was developing a Chromecast app, P27's solution was to create a dummy sender application to pass fake custom messages to invoke test cases of the receiver app (the software component running on the Chromecast device that renders content on the TV). He said:

"It's extremely challenging when you're developing the mobile apps, and you have a developer developing the receiver app, and they're not far enough along to do some of the testing. You need that sort of mock workflow...to work through scenarios when the mobile apps were not even done yet."

Similarly, P29 also created a dummy sender app to test his receiver app.

Difficulty of Automating UI Tests

Also related to the distributed nature of crossmedia experiences, some participants found it difficult to automate UI tests that require triggering events on different devices in a coordinated manner. P26 expressed his concern about this issue:

"It basically is manual testing, which is really time-consuming, really hard to do consistently."

The challenge of testing was often brought by the increased physicality of multi-device experiences. For example, one manual test case we learned from P26 requires the tester to have 2 or more Chromecast units set and available, and then tap the Cast icon in the sender application running on a smartphone. The test case further stipulates that the expected results should be two or more Chromecast device names appearing on a list. As this test case shows, to successfully execute such a simple test requires manual manipulations of physical devices by the tester.

Developers would want to reduce their reliance on physical manipulations of devices in tests. Here is an extreme but illustrative example:

"We would do silly things like wrap the Chromecast in tinfoil to trigger disconnect. You know, things like that. If there were tools available that allow you to manually trigger things like that, it would've been beneficial." (P27)

To summarize, testing multi-device experiences is often inadequate, inefficient, and inconsistent for four main reasons:

1. Many software and hardware differences between devices cause applications to appear and function differently.

2. Emulators are not very useful because of their performance limitations and their inability to simulate low-level hardware stacks critical to inter-device communication.
3. The interdependency of devices in a crossmedia service makes it hard to test each component independently when the components are being developed in parallel and potentially by different teams.
4. UI tests are difficult to automate due to the need to physically manipulate devices in many test cases.

DISCUSSION

In this section, we discuss the contributions of our research, two change-drivers behind the three key challenges we identified, and the implications for supporting designers and developers of multi-device experiences.

Research Contributions

Through interviewing designers and developers who are creating multi-device user experiences, we have identified a number of challenges for design and development. These challenges were either not addressed or under-examined in past research. For example, past studies has reported the issue of adapting UIs to devices with differing input and output constraints, but they do not address the impact of platform UI standards. Our research shows that expecting designers to make the UI consistent across platforms and simultaneously follow the native UI standard of each target platform has posed both usability and organizational challenges to designing quality multi-device experiences, raising questions such as:

- How often do users cross platform boundaries?
- Will users tolerate a design that deviates from the platform conventions in order to optimize for cross-platform consistency?
- What aspects of cross-platform consistency are users like to find most important?

Moreover, we have showed that designing sound business logic governing how devices interact with one another is extremely difficult due to uncertain user intentions and the lack of frameworks and tools to allow designers to foresee disruptive scenarios. This complexity for designers has not been revealed and detailed in prior work as far as we know.

In addition, our interviews have shown that testing multi-device experiences is often an onerous effort. Researchers previously warned that user testing a multi-device experience could be difficult [1]; our research reveals that this challenge goes beyond user testing but includes several important types of software testing such as functional tests, compatibility tests, and GUI tests.

Change-Drivers Behind the Challenges

We believe the three challenges we identified reflect two broader changes in the technological landscape, which we discuss below.

The Complexity Introduced by Collaborations between Devices

Research has shown that users want their devices to collaborate with one another [7,14], and our interviews reveal that some designers and developers have started experimenting with features that involve sequential or parallel uses of multiple devices. These new types of interactions have brought additional complexity that designers and developers struggle to understand and address. There are several dimensions of this complexity. First, a device's action often depends on the properties and states of other devices. Second, devices in a multi-device system often initiate actions with one another on behalf of the user, as in the case of the Smart Lock technology. Last, designers often need to design for task continuity under a great amount of uncertainty about the user's state and intention. To their disappointment, the design tools and frameworks they use today are not well equipped to address such uncertainty in multi-device experience design.

The Continued Diversification and Standardization of the Device Ecosystems

Paradoxically, the device ecosystems appear to be undergoing both diversification and standardization at the same time. As a result, multi-device designers and developers are caught between these two parallel processes.

On the one hand, there are strong technological and business factors driving the trend towards diversification. The barrier to entering hardware design and manufacturing has been drastically lowered over the past few years due to the rise of the maker movement, crowd funding platforms, and the open manufacturing paradigm started in Southern China [10]. As more hardware vendors join the competition, they all want to differentiate their devices from others by adding unique features such as Apple's 3D Touch[6], Amazon's Dynamic Perspective[7], and YotaPhone's dual displays[8], not to mention numerous vendor customizations to the Android operating system. The second driving force is the aging of mobile ecosystems, which has already led to co-existence of many old models and new models that have very different capabilities and designs. Supporting both new and old devices is hard but often necessary to retain market share. The last driving force is the likely rise of modular phones being developed by efforts such as Google's Project Ara[9]. When such technologies become commercially viable, there will be a new wave of device customization driven by consumers and makers.

On the other hand, the OS makers have relentlessly pursued an eco-system strategy that seeks to deliver unified

[6] 3D Touch. http://www.apple.com/iphone-6s/3d-touch/

[7] How 'Dynamic Perspective' Brings 3D to the Amazon Fire Phone. http://mashable.com/2014/06/18/amazon-fire-phone-3d-dynamic-perspective/

[8] YotaPhone. https://yotaphone.com/us-en/

[9] Project Ara. http://www.projectara.com/

experiences across the products within their respective ecosystems. Standardization of UI design has been an important part of that strategy. Our data suggest that this trend complicates multi-device design and development in several ways. First, it might create a perception among designers that users have become less tolerant of UI designs that deviate from the platform standard they are accustomed to. Second, as a substantial part of the application needs to be designed specifically for each target platform, synchronizing changes across platforms become an onerous effort. Last, it is likely to become harder to design experiences that are not only cross-device but also cross-ecosystem, since standardization *within* ecosystems often means diversification *between* them.

Key Implication: Better Multi-Device Simulation

Based on our data and analysis, we believe that the ability to simulate multi-device interactions at early stages of the design and development process is key to addressing challenges related to both designing and testing those interactions.

Not only should such a simulator be able to emulate properties and traits of individual devices, it also needs to incorporate other factors crucial to producing an accurate preview of the multi-device experience being designed. These factors include but are not limited to data flows between devices, timing of actions, usage contexts, performance of the devices and the infrastructure, availability of the devices, and their security settings. Such simulations can help designers better anticipate the behaviors of the application, shorten the feedback loop, and quickly iterate the design. Furthermore, those simulations may also help designers better communicate their design ideas to stakeholders including users and get their feedback earlier in the design process. An additional use of simulation is in testing. It can potentially reduce the need for physical manipulations of devices and paves a road to more automated cross-device tests.

Some research prototyping systems, such as the Weave IDE [2] and XDStudio [12], provide features to preview multi-device interactions within their respective design and prototyping environments. Unfortunately, they lack the ability to simulate usage scenarios with authentic data and contexts. We suggest tool developers and researchers consider adopting the “capture and replay” approach proposed by Newman et al. [13], who developed a tool called Replay to support the design and testing of location-based services. Replay allows the developer to playback episodes of GPS traces captured from real users’ mobile phones to examine the behaviors of their applications. There are many ways this approach might be adapted to multi-device development. For example, traces of a device’s presence in specific environments might be captured and made available in a tool to allow designers to more easily explore real-world scenarios.

Future Work

In addition to building simulation tools for multi-device experiences, it is critical to update and deepen our understanding about how people use multiple devices in a variety of settings, especially as support for such experiences gradually enhances over the time. Based on what we have learned from our interviews, we suggest future work explore two particularly important questions:

1. When do users cross platform boundaries, especially within the same task? How their experiences might be affected by the differences between platform UI standards?
2. How do social dynamics, device ownership, and timing play a role in shared crossmedia experiences? For example, should a Chromecast device treat all clients equally or prioritize connecting requests from certain members of the household?

It is also important to keep track of emerging tools and practices adopted by designers and developers. As tools and toolkits for developing multi-device experiences become more mature and available, there is an opportunity to gain insights from conducting surveys and analyzing online discussions about multi-device design and development.

CONCLUSION

Support for multi-device interactions has fallen behind users’ increasing desire to leverage the diverse capabilities of the devices that surround them. Through interviewing 29 practitioners working in this area, we identified three key challenges that have prevented designers and developers from building usable multi-device systems despite of growing user demand. In particular, our work highlights the following challenges that have not been extensively examined before:

- The difficulty in designing interactions between devices
- The complexity of adapting user interfaces to different platform UI standards
- The lack of tools and methods for testing multi-device user experiences

Based on these findings, we suggest further research into simulations of multi-device experiences as a way to lower the barrier to designing and developing truly useful, usable, and enjoyable multi-device experiences.

ACKNOWLEDGMENTS

We thank our participants and our colleagues for their help with the study.

REFERENCES

1. Ville Antila and Alfred Lui. 2011. Challenges in Designing Inter-usable Systems. In *Human-Computer Interaction – INTERACT 2011*, Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque and Marco Winckler (eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 396–403. Retrieved

September 20, 2015 from http://link.springer.com/10.1007/978-3-642-23774-4_33

2. Pei-Yu (Peggy) Chi and Yang Li. 2015. Weave: Scripting Cross-Device Wearable Interaction. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 3923–3932. http://doi.org/10.1145/2702123.2702451
3. Charles Denis and Laurent Karsenty. 2003. Inter-Usability of Multi-Device Systems – A Conceptual Framework. In *Multiple User Interfaces*, Ahmed Seffah and Homa Javahery (eds.). John Wiley & Sons, Ltd, 373–385. Retrieved September 21, 2015 from http://onlinelibrary.wiley.com/doi/10.1002/0470091703.ch17/summary
4. Steven Dow, T. Scott Saponas, Yang Li, and James A. Landay. 2006. External Representations in Ubiquitous Computing Design and the Implications for Design Tools. *Proceedings of the 6th Conference on Designing Interactive Systems*, ACM, 241–250. http://doi.org/10.1145/1142405.1142443
5. David Gilbert. Android smartphone boom sees over 24,000 distinct models in use worldwide. *International Business Times UK*. Retrieved January 10, 2016 from http://www.ibtimes.co.uk/android-smartphone-boom-sees-over-24000-distinct-models-use-worldwide-1514342
6. Hanna-Maria Halkosaari, L. Tiina Sarjakoski, Salu Ylirisku, and Tapani Sarjakoski. 2013. Designing a Multichannel Map Service Concept. *Human Technology* Volume 9, 1: 72–91. http://doi.org/10.17011/ht/urn.201305211723
7. Tero Jokela, Jarno Ojala, and Thomas Olsson. 2015. A Diary Study on Combining Multiple Information Devices in Everyday Activities and Tasks. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 3903–3912. http://doi.org/10.1145/2702123.2702211
8. Sung Woo Kim, Han Kyung Jo, and Da Yun Ha. 2011. Different UI, Same UX: A Design Concept for Implementing a Locally-Optimized and Globally-Unified User Experience. In *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, Aaron Marcus (ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 440–448. Retrieved September 20, 2015 from http://link.springer.com/10.1007/978-3-642-21708-1_50
9. Michal Levin. 2014. *Designing Multi-Device Experiences: An Ecosystem Approach to User Experiences across Devices*. O'Reilly Media, Beijing.
10. Silvia Lindtner. 2014. Hackerspaces and the Internet of Things in China: How makers are reinventing industrial production, innovation, and the self. *China Information* 28, 2: 145–167. Retrieved October 1, 2015 from http://cin.sagepub.com/content/28/2/145.short
11. James Lin. 2005. Using Design Patterns and Layers to Support the Early-stage Design and Prototyping of Cross-device User Interfaces.
12. Michael Nebeling, Theano Mintsi, Maria Husmann, and Moira Norrie. 2014. Interactive Development of Cross-device User Interfaces. *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 2793–2802. http://doi.org/10.1145/2556288.2556980
13. Mark W. Newman, Mark S. Ackerman, Jungwoo Kim, et al. 2010. Bringing the field into the lab: supporting capture and replay of contextual data for the design of context-aware applications. *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM, 105–108. http://doi.org/10.1145/1866029.1866048
14. Stephanie Santosa and Daniel Wigdor. 2013. A Field Study of Multi-device Workflows in Distributed Workspaces. *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, 63–72. http://doi.org/10.1145/2493432.2493476
15. Katarina Segerståhl. 2009. Crossmedia Systems Constructed around Human Activities: A Field Study and Implications for Design. In *Human-Computer Interaction – INTERACT 2009*, Tom Gross, Jan Gulliksen, Paula Kotzé, et al. (eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 354–367. Retrieved September 21, 2015 from http://link.springer.com/10.1007/978-3-642-03658-3_41
16. Henrik Sørensen, Dimitrios Raptis, Jesper Kjeldskov, and Mikael B. Skov. 2014. The 4C Framework: Principles of Interaction in Digital Ecosystems. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, 87–97. http://doi.org/10.1145/2632048.2636089
17. Nate Swanner. 2015. The new emulator in Android Studio 2.0 is 50 times faster than before. *The Next Web*. Retrieved January 17, 2016 from http://thenextweb.com/dd/2015/11/23/the-new-emulator-in-android-studio-2-0-is-50-times-faster-than-before/
18. Tyler Tate. 2011. The Rise of Cross-Channel UX Design. *UX Matters*. Retrieved from http://www.uxmatters.com/mt/archives/2011/10/the-rise-of-cross-channel-ux-design.php
19. Minna Wäljas, Katarina Segerståhl, Kaisa Väänänen-Vainio-Mattila, and Harri Oinas-Kukkonen. 2010. Cross-platform Service User Experience: A Field Study and an Initial Framework. *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, ACM, 219–228. http://doi.org/10.1145/1851600.1851637