XDSession: Integrated Development and Testing of Cross-Device Applications

Michael Nebeling[1]*, **Maria Husmann**[2], **Christoph Zimmerli**[2], **Giulio Valente and Moira C. Norrie**[2]
[1] Human-Computer Interaction Institute, Carnegie Mellon University
[2] Department of Computer Science, ETH Zurich
nebeling@cmu.edu, { husmann, zimmerli, norrie }@inf.ethz.ch

ABSTRACT
Despite the recent proliferation of new cross-device application frameworks, there is still a lack of sophisticated tools for testing new applications during their development. This paper presents *XDSession*—a framework for cross-device application development based on a concept of cross-device sessions, not only useful for managing distribution and synchronisation, but also for logging and debugging. Integrated with XDSession are two new tools specifically designed for cross-device testing. First, the *session controller* supports management and testing of cross-device sessions with connected or simulated devices at run-time. Second, the *session inspector* enables inspection and analysis of multi-device/multi-user sessions with support for deterministic record/replay of cross-device sessions. We show the utility of XDSession based on a case study of a semester-long course project in which our tools were used by students to reimplement an existing application and extend it with cross-device capabilities.

Author Keywords
multi-user/multi-device applications; cross-device development platform; session concept.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Input devices and strategies, Interaction styles.*

INTRODUCTION
For many years, both research and industry have focused on improving input and output capabilities of available types of devices as well as developing new and more powerful forms of devices. While this will continue to be important, a new direction is to try and improve the interaction between devices so that multiple devices can more easily be used in combination and form a uniform interaction space.

Cross-device development requires design decisions and implementation effort on different levels, involving the adaptation, distribution and synchronisation of both the user interface and data across devices [6]. Many different tools have been developed to address the issues of distributing and synchronising parts of a user interface across multiple devices [2, 5, 8]. However, little has been done to investigate the differences between traditional and cross-device development processes. In particular, testing becomes more difficult as it has to consider the use of multiple devices by multiple users in co-located, remote or mixed usage scenarios.

To address this problem, we have developed *XDSession*, a lightweight and extensible framework for cross-device development with support for integrated development and testing of cross-device applications. It is based on cross-device session concepts from [6] for managing the distribution of both the user interface and information. This paper explores how the concepts can be exploited for logging and debugging, and makes the following contributions:

- extended cross-device session concepts with new operations for sharing data between multiple sessions;
- *XDSession*, our reference implementation of the concepts supporting the full set of operations and serving as a platform for research into cross-device testing;
- the *session controller* and *session inspector* as two novel tools specifically designed to support cross-device development and testing based on the new concepts.

We also present a case study of an independent semester project in which XDSession was used by students to enhance an existing application with cross-device capabilities.

CROSS-DEVICE SESSION CONCEPTS & OPERATIONS
The literature often informally speaks of sessions to refer to information interchange between a user and a device or between two or more communicating devices. We wanted to formalise this concept and use it as a construct both for grouping users and devices as well as associated information. This section defines the concepts and available operations.

Figure 1 shows three users with four devices participating in two sessions sharing data. Users 1 and 2 with their respective devices are participating in Session A, while User 3 with Devices 3 and 4 is in Session B. Changes made to the shared data overlapping both sessions are sent to all four devices, while

* Michael Nebeling conducted this research while at ETH Zurich. He is now affiliated with Carnegie Mellon University and funded by a Swiss NSF Advanced Postdoc.Mobility grant, P300P2_154571.

EICS'15, June 23 - 26, 2015, Duisburg, Germany

http://dx.doi.org/10.1145/2774225.2775075

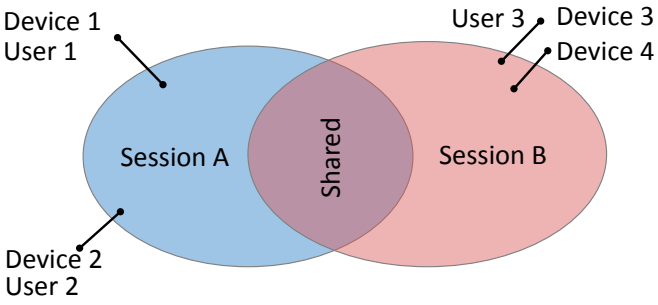
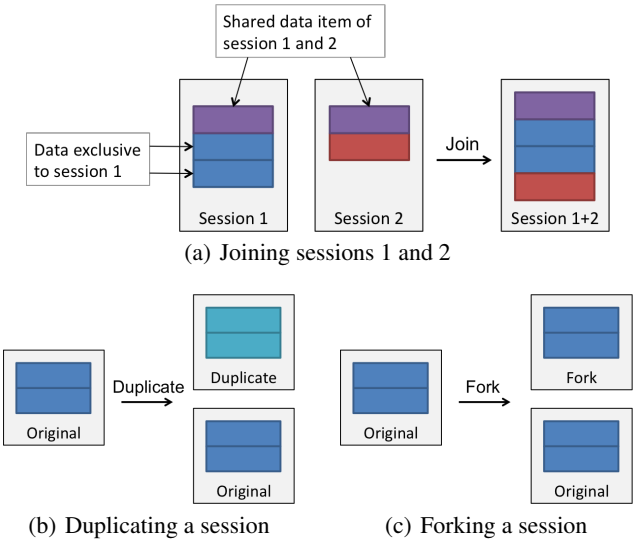


Figure 1. Cross-device sessions example

changes to data contained in a single session are sent only to the directly connected devices.

Our concept of a Session S is defined as $S =< U, D, I >$ linking the concepts of User U, Device D and Information I represented as data and metadata. Data refers to the information managed in a session, while metadata describes the interactions and how the data was manipulated during the session.



(a) Joining sessions 1 and 2

(b) Duplicating a session (c) Forking a session

Figure 2. Session operations

Based on these concepts, we have defined seven session operations: *create*, *edit*, *delete*, *join*, *leave*, *duplicate*, and *fork*. The first three cover basic CRUD operations to manage information *within* sessions. The remaining four operations illustrated in Figure 2 are more interesting as they control how information is managed and shared *between* sessions.

A device can join multiple sessions. On join, the device receives the whole or the missing part, i.e. the part of the newly joined session which does not overlap with any other session the device is already part of, of the data belonging to the selected session. When a device leaves a session, data of that session which is not shared with any other session is removed from the device. We define a fork as a dependent copy of a session, where a new session is created maintaining the references to the data of the existing session being forked. On the other hand, duplication creates an independent copy of an existing session, i.e. a new session with duplicated data.

XDSESSION

Based on the session concepts, we designed XDSession as a framework for cross-device application development. Figure 3 shows the main components of our framework. Our implementation of XDSession is based on a client/server architecture. The server is responsible for managing and distributing both the data and the UI of an application. The client provides two specific components for handling sessions, namely the *session controller* and the *session inspector*. The session controller enables users to manage devices and sessions in an easy way. It also allows developers to simulate different types of devices for testing. The session inspector provides a logging system and introduces a method for reproducing events logged as metadata stored in a session. The session management components are shared by the client and server.

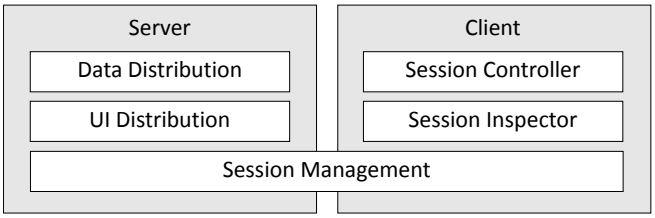


Figure 3. XDSession framework components

XDSession applications are written in HTML5 and JavaScript. Web developers can therefore write applications using technologies with which they are already familiar. XDSession's server takes care of both persistence and synchronisation. The developer can focus on the client implementing the UI and logic of the application itself. Developers who only want to build cross-device applications can directly benefit from the framework support without having to understand the underlying mechanisms, while tailoring and extending the framework is also possible for more advanced applications.

XDSession uses an event-driven architecture for session management based on callback functions to inform about performed actions. For most operations, the response contains a reference to the data, or is `null` if an error occurred. Data created, updated or deleted from other devices is received in realtime from a listener function. The received object contains a reference to the respective data as well as metadata including the user, device and sessions involved in the action. In addition, developers can define custom events in order to send data between devices. Unlike the previous operations, custom operations may not be logged as part of the sessions they concern if the developer chooses not to do so. This feature is designed to stream data and support the handling of interactions in DUIs, while, at the same time, providing means for optimisation if data or metadata do not need to be stored.

The remainder of this section presents the session controller and session inspector as two tools based on these concepts specifically designed for testing XDSession applications.

Session Controller

The session controller is a visual tool that can be used for managing cross-device sessions in XDSession applications.

Beyond that, our goal was to allow the session controller to be used as a prototyping tool enabling the rapid creation of virtual sessions not associated with any connected device, thus for simulating and testing different multi-user/multi-device scenarios facilitating the cross-device development process.

While developers may choose to implement their own session controller using the XDSession API, a default session controller can be integrated in the HTML body of an XDSession application via a custom `session-controller` HTML tag. It essentially embeds a toggle menu giving access to user, session and device properties at run-time. From this menu, it is possible to create new and manage existing users as well as selecting the user(s) in control of the current session. It is also possible to specify which users are allowed to interact with a device. A new session can be created and the device then automatically joins the new session. As explained before, a device can join multiple sessions. Shared data is automatically synchronised between devices by the framework. When a session is removed, it is first deselected from all the devices and then the data that is not shared is removed. While the framework is able to automatically detect the device type, the properties can also be edited. This enables simulation of a different device type for testing purposes.

Session Inspector

The session inspector is a visual tool for inspecting data and metadata of cross-device sessions in XDSession applications. In combination with the session controller, it can be used for recording and replaying sessions and therefore actions triggered on connected or simulated devices. Similar to the session controller, it can be integrated via a custom `session-inspector` HTML tag.

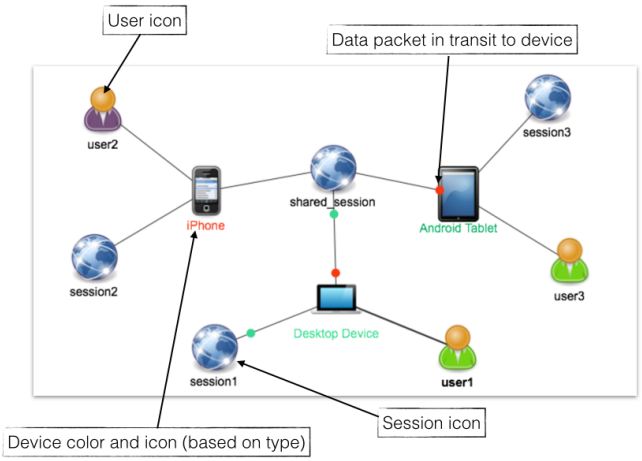


Figure 4. Overview tab in Session Inspector

The overview shown in Figure 4 visualises the state of an XDSession application in terms of the user, device and session properties. Changes to the data and metadata are visualised as packets flowing from the source device to all devices participating in the respective session(s). The graph can be panned and zoomed and via double-click it is possible to inspect the selected entities. With this tool, it is possible to monitor the flow of information and trace messages between devices and sessions, which supports keeping an overview and helps verifying that a session was indeed configured correctly.

All collected information can be inspected in more detail via the logger shown Figure 5 with which it is possible to browse all past events and see new events in realtime. Each event triggered by the user or generated by the system is automatically added to the log. The log can be filtered by device type(s), i.e. smartphone, tablet, desktop, by session(s) and by device(s). A double click on a log entry enters the debug mode and jumps to the respective event on the timeline.

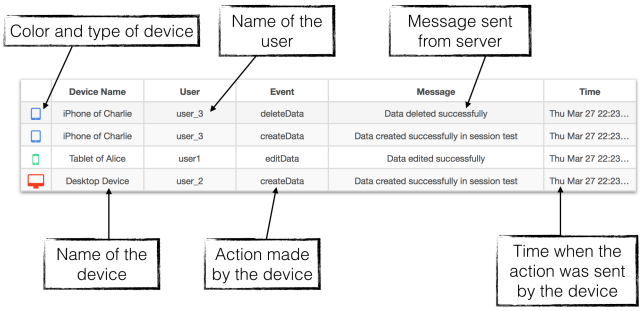


Figure 5. Logger in Session Inspector

The timeline shown in Figure 6 is the core of the session inspector as a debug tool allowing to navigate the timeline of recorded events. It enables deterministic record/replay similar to [1], a technique that gives the possibility of reproducing the actions that took place at a certain time, without altering the course of the events [4]. The session inspector can reproduce session events and therefore undo/redo create, modify and delete actions of data based on the stored metadata. Events can be replayed by changing the log time either via the forward/backward controls or by clicking an event.

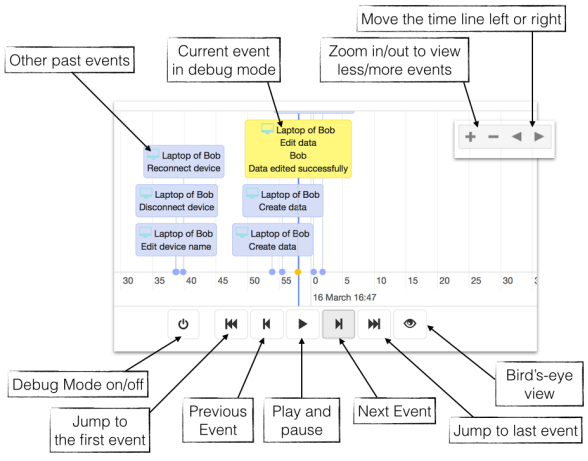


Figure 6. Timeline interface in Session Inspector

To replay events, it is necessary to enter the debug mode. This action synchronises the devices with the log time, which is then maintained and propagated to all participating devices. Sessions that are replayed are processed independently of any

active sessions in XDSession. The user can combine and review recorded sessions on the fly. Our technique allows seamless testing on the current and participating devices, and exiting the debug mode at any time to resume the active sessions.

With the remaining tabs, it is possible to obtain detailed information on users, devices and sessions. The three views are connected with each other: the first shows registered users and their devices, the second shows the devices and sessions to which they are connected, and the third shows the sessions and contained information. In each session it is possible to see both active and deleted data, where deleted data is marked in red. In addition, it is possible to see with which session(s) the data is shared. Finally, for each data entry, it is possible to see collected metadata in terms of the performed action, the user and the device that carried out that action as well as a short text summarising the action.

IMPLEMENTATION

XDSession consists of a server-side and a client-side component. The server is built in Java and handles 1) data distribution, 2) persistence, and 3) UI distribution. For data distribution, the server keeps track of all users, devices, and their sessions. When a device joins a session, the server sends it all necessary data. When a data item is changed on a device, the server propagates the changes to all devices that share the same session. To provide such updates from server to client, bidirectional realtime communication is needed, which we realise with Socket.IO. On the server-side we use an open-source Socket.IO implementation[1] based on the Netty[2] server framework. Persistence is implemented based on the db4o[3] object-oriented database. The server stores a string representation of each data item. Additionally, metadata such as sessions, devices, users, and logs are also made persistent. XDSession includes a Jetty[4] web server and provides a basic UI distribution mechanism that allows to deliver differing html-files tailored to the device type. Device type detection is done with MobileESP[5].

The client is written in TypeScript[6], a typed superset of JavaScript. TypeScript code compiles to JavaScript; hence, developers can opt to use the JavaScript version of XDSession and are not forced to use TypeScript. The client logic consists of modules for 1) device detection, 2) persistence, 3) session management and 4) logging. The device detection relies on the client-side implementation of MobileESP. The device information is available to the developer on the client side, but also transmitted to the server for the UI distribution. The client saves information and settings of the current device in local storage, so that they persist across browsing sessions. In particular, the current device, sessions, and user are stored. The session management module administrates information of users, sessions, and devices present in the system. It communicates with the server to keep up to date. The log module provides access to the log and is mainly used by the session inspector, but could also be used directly by a developer. On top of the client modules, the session inspector and the session controller are implemented in AngularJS[7]. In addition, the session inspector uses the CHAP Links Library[8] for the graph and timeline visualisations. The session controller relies on Alertify[9] for dialogs and notifications.

IDEA ARCHIPELAGO CASE STUDY

This section presents a case study on how XDSession was used to support the Idea Garden EU FP7 research project[10]. The research project running from October 2012 to September 2015 involves eight academic and industrial partners from all across Europe to design an interactive learning environment fostering creativity. In the first year, a first version of an application called *Idea Archipelago* was designed around an interactive whiteboard installation and implemented using Windows technologies. In the second year, it was decided to switch to web technologies in order to expand the support for different devices and enable cross-device use of the application. XDSession, not only enabled a rapid switch of technologies and cross-device support, but also facilitated the development process as more devices were supported.

Original Idea Archipelago

The original application was implemented for the Windows .NET platform using Windows Presentation Foundation[11] technologies. The application is targeted at an interactive whiteboard setting with digital pens as input devices. It allows users to manage information related to projects. After creating and opening a project, users can import files (images, videos, documents) from the file system and via image web search (e.g. Flickr, Google images) onto a fixed-size grid map. On the map, files can be clustered into groups. These groups can be named and moved as one entity. In addition to importing existing files, the application also allows to create new content in grid cells by launching external applications for editing or creating small post-it note-like scribbles right in the cell. Items on the map can also be duplicated and deleted.

Development Process

In a semester-long (14 weeks) lab project we set students the task of re-implementing the existing Idea Archipelago application using web technologies and extending it with cross-device capabilities. The group consisted of four computer science master students (2 males + 2 females, age 23-26, final year MSc Computer Science, average 2 years web programming experience). Each of them committed an average of ten hours per week to the project.

The lengths of the different periods of the semester project are shown in Table 1. The first part of the students' work concentrated on re-implementing the existing application's features

[1] https://github.com/mrniko/netty-socketio
[2] http://netty.io
[3] http://www.db4o.com
[4] http://eclipse.org/jetty
[5] http://mobileesp.com
[6] http://www.typescriptlang.org
[7] https://angularjs.org
[8] http://almende.github.io/chap-links-library
[9] http://fabien-d.github.io/alertify.js
[10] http://idea-garden.org
[11] http://msdn.microsoft.com/en-us/library/ms754130%28v=vs.110%29.aspx

Task	Length
Re-implementation of Archipelago functionality	5 weeks
Scenarios for multi-/cross-device work	1 week
Adaptations to different device characteristics	2 weeks
Familiarisation with XDSession	1 week
Architecture change to MVC	1 week
Integration of XDSession	1 week
Notifications of other users' actions	1 week
Preparation for presentations	2 weeks

Table 1. Lengths of the semester project's phases

with web technologies. The focus was on single device/user settings on desktop computers with mouse and keyboard as input devices. This part was finished after 5 weeks.

Next, the students shifted their focus from development work to envisioning multi-device/multi-user scenarios for the application. These ideas were implemented in two parts, carried out in parallel by different students: (1) adapting to other devices and (2) adding cross-device functionality.

Adaptation to other devices mainly meant that the target device range was extended to mobile devices (tablets and smartphones). The idea was to make the application's full functionality also available on these devices. This required adaptations to much smaller screen sizes and touch input. Figure 7 shows the resulting UI running across different devices.

Figure 7. Idea Archipelago XD

The students were provided with the XDSession framework to use it as a basis for adding cross-device functionality to the application. The familiarisation with, and actual integration of, XDSession each took roughly one week. The new web-based, cross-device version was named *Idea Archipelago XD*.

Integration of XDSession

When XDSession is adopted in order to add cross-device functionality to an application under development, the main design choice is how to map application concepts onto XDSession's session concepts. Here, the goal was to support users with collaborative work on the information related to a project. Multiple users should be able to employ multiple devices (even per user) to manipulate the project's data at the same time. They should be aware of the other users and devices who are collaborating and of their actions. Therefore, the decision was made to map projects onto XDSession sessions. Creating a new project translates to creating a session. Opening a project from the application's project overview means joining that session. Manipulating the project's data means editing the session data, which leads to these changes being propagated to all other devices currently in the session and therefore the project. These other devices can then show notifications to make the users aware of the changes.

In addition to notifications about applied changes, it was also decided to show information labels on cells whose content is currently being edited on another device, as well as on cells and groups that are currently being moved on another device. The original application did not need such a mechanism, since the collaborative whiteboard setting it targets means that other users can always see directly if one user is editing a certain cell. The information labels are, however, no locks. If a second user decides to edit an item that is already being edited anyway, XDSession's *last-update-wins* semantics would decide which version of the content would result at the end. However, XDSession's history functionality allows the application to display who has edited an item and when, thus tracing lost updates due to concurrent edits.

Observations and Feedback

The feedback on XDSession we received from the students is encouraging: the students appreciated its features as they allowed them to save precious development time and focus their efforts on the UI adaptations and application-specific cross-device support without having to face all the fundamental challenges of providing cross-device distribution and synchronisation support. For us, it was particularly promising to see that only little time was required to understand and integrate the framework. Our reviews revealed high-quality code and effective use of XDSession's major features.

Finally, we want to highlight three examples of how the students made innovative use of the framework and tool support and two ways how they could be beneficial to the project in the future.

First, they carefully prepared a project and therefore a session for a live cross-device demo of both the application's features and the use of the framework. This session was useful for rehearsal and tests using the students' own mobile devices. Second, the session inspector was used to verify that the correct devices were connected to the system and that they joined the correct sessions. For this, the graph in the overview tab was especially useful. Third, when debugging issues with the new notification functionality of Archipelago XD, the logging facility of the session inspector was used to determine if events were actually produced by the source or not processed by the target device.

In the future, the session inspector could be used to analyse the need for collaborative editing support and determine the best possible user experience that could be achieved. Session logs could be produced by simulating realistic multi-user/multi-device scenarios and studied using the different views available in XDSession's inspector. For example, the number of lost updates could be determined and be used as a basis for deciding on a strategy for concurrency control. Finally, the session controller and inspector could be

used together to produce a number of recordings of meetings in which Idea Archipelago XD is used to discuss the next steps planned as part of the European research project. These recordings in the form of sessions could then be shared with the other project partners to be reproduced and replayed interactively for them to get a better understanding of the ideas.

RELATED WORK

This paper proposes a set of tools for testing and debugging specifically designed for cross-device applications. The common approach of using browsers' in-built tools such as Firebug[12] or Chrome Developer Tools[13] in combination with device emulators or remote debugging via USB helps to catch errors in the code, but is insufficient to deal with problems that may arise in a multi-user/multi-device context.

Browser-independent tools include PhoneGap Emulator[14] and Weinre[15]. PhoneGap Emulator is a browser plugin that integrates a browser-based device emulator. The tool emulates PhoneGap's core APIs and can be used alongside Chrome Developer Tools for debugging UI and application logic. Weinre is a remote debugger for mobile web applications. Similar to Chrome Developer Tools, it provides a JavaScript console, system profiler and DOM inspector. While being effective tools for emulation and remote inspection, they are still oriented towards single-device applications.

Another important shortcoming of these tools is that they are not able to reproduce errors that have occurred during testing. Advanced debugging support has therefore been the subject of recent research. Mugshot [4] is a tool for capturing events from a JavaScript programme. It provides deterministic record/replay of a web application, making it possible to reproduce and analyse certain behaviours. The authors identified a number of advantages of using this technique, such as failure analysis, performance evaluation, and even usability analysis of a GUI. Timelapse [1] is a similar tool that allows developers to record and replay interactive behaviours in web applications. It is possible to browse, visualise, and seek within the recorded programme, while using common debugging approaches such as breakpoints or reading the logger. With XDSession, we explore a simplified technique, but one that is better suited to multi-device/multi-user environments.

A number of promising projects have been launched outside the research community. Firebase[16] is a cloud-based service that stores and synchronises JSON data in realtime across devices. Basic tool support for testing and debugging is provided in the manner of an inspection tool that allows to observe and manipulate the data in the cloud. However, only the current state is presented and no logging is offered. In contrast, Meteor[17] is a full stack framework. Data synchronisation across clients is supported based on a collection concept. It offers a publish-subscribe mechanism to control how the data is shared among clients. While both Meteor and Firebase have a concept of a user, there is none for devices or the notion of grouping data in sessions. PubNub[18] is another realtime framework, which provides synchronisation of data across multiple devices. While it does track the online status of devices, the devices or users are not associated to changes in the data, which is the case in XDSession.

[12] `http://getfirebug.com`

[13] `https://developers.google.com/chrome-developer-tools`

[14] `http://emulate.phonegap.com`

[15] `https://people.apache.org/~pmuellr/weinre`

[16] `https://www.firebase.com`

[17] `https://www.meteor.com`

CONCLUSION

In this paper, we have presented XDSession—a new framework for developing cross-device applications with integrated support for testing and debugging. The framework implements the concept of cross-device sessions introduced in [6]. Cross-device sessions are treated as data containers used by multiple devices whose purpose is to record the data exchange between multiple devices and support session playback for data sharing between sessions.

To aid development and testing of cross-device applications based on these concepts, XDSession integrates two novel tools for managing and analysing cross-device sessions. The session controller can be used to manage devices and group them into sessions. Devices may actually be connected to the system, or they can be simulated. Both are useful for testing. The session inspector facilitates inspection and analysis of cross-device sessions. It adapts an existing method for deterministic record/replay to cross-device sessions and supports monitoring of data exchange between devices and users.

REFERENCES

1. Burg, B., Bailey, R., Ko, A. J., and Ernst, M. D. Interactive Record/Replay for Web Application Debugging. In *Proc. UIST* (2013).
2. Frosini, L., and Paternò, F. User Interface Distribution in Multi-Device and Multi-User Environments with Dynamically Migrating Engines. In *Proc. EICS* (2014).
3. Heikkinen, T., Goncalves, J., Kostakos, V., Elhart, I., and Ojala, T. Tandem Browsing Toolkit: Distributed Multi-Display Interfaces with Web Technologies. In *Proc. PerDis* (2014).
4. Mickens, J. W., Elson, J., and Howell, J. Mugshot: Deterministic Capture and Replay for JavaScript Applications. In *Proc. NSDI* (2010).
5. Nebeling, M., Mintsi, T., Husmann, M., and Norrie, M. C. Interactive Development of Cross-Device User Interfaces. In *Proc. CHI* (2014).
6. Nebeling, M., Zimmerli, C., Husmann, M., Simmen, D., and Norrie, M. C. Information Concepts for Cross-Device Applications. In *Proc. DUI@EICS* (2013).
7. Paternò, F., and Santoro, C. A Logical Framework for Multi-Device User Interfaces. In *Proc. EICS* (2012).
8. Yang, J., and Wigdor, D. Panelrama: Enabling Easy Specification of Cross-Device Web Applications. In *Proc. CHI* (2014).

[18] `http://www.pubnub.com`