SurfaceFleet: Exploring Distributed Interactions Unbounded from Device, Application, User, and Time

Frederik Brudy† [1,2], David Ledo† [1,3], Michel Pahud[1], Nathalie Henry Riche[1], Christian Holz[1,4], Anandghan Waghmare[1,5], Hemant Bhaskar Surale[1,6], Marcus Peinado[1], Xiaokuan Zhang[1,7], Shannon Joyner[1,8], Badrish Chandramouli[1], Umar Farooq Minhas[1], Jonathan Goldstein[1], William Buxton[1], Ken Hinckley[1]

[1]Microsoft Research, Redmond, WA, United States; [2]University College London, London, UK; [3]University of Calgary, AB, Canada; [4]ETH Zürich, Switzerland; [5]Georgia Institute of Technology, GA, USA; [6]University of Waterloo, Canada; [7]Ohio State University, OH, USA; [8]Cornell University, NY, USA
f.brudy@cs.ucl.ac.uk & david.ledo@ucalgary.ca; surface-fleet@microsoft.com

ABSTRACT
Knowledge work increasingly spans multiple computing surfaces. Yet in status quo user experiences, content as well as tools, behaviors, and workflows are largely bound to the current *device*—running the current *application*, for the current *user*, and at the current moment in *time*. SurfaceFleet is a system and toolkit that uses resilient distributed programming techniques to explore cross-device interactions that are unbounded in these four dimensions of device, application, user, and time. As a reference implementation, we describe an interface built using Surface Fleet that employs lightweight, semi-transparent UI elements known as *Applets*. Applets appear always-on-top of the operating system, application windows, and (conceptually) above the device itself. But all connections and synchronized data are virtualized and made resilient through the cloud. For example, a sharing Applet known as a *Portfolio* allows a user to drag and drop unbound *Interaction Promises* into a document. Such promises can then be fulfilled with content asynchronously, at a later time (or multiple times), from another device, and by the same or a different user.

Author Keywords
cross-device interaction; distributed systems; mobility

CSS Concepts
• Human-centered computing~Ubiquitous and mobile

INTRODUCTION
Modern information work increasingly relies on multi-device workflows and distributed workspaces [66]. Mobility of this sort implies the need for an ecosystem of technologies [22] that transition user activity from one *place* to another, whether that "place" takes the form of a literal location, a

UIST '20, October 20–23, 2020, Virtual Event, USA
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7514-6/20/10…$15.00
https://doi.org/10.1145/3379337.3415874

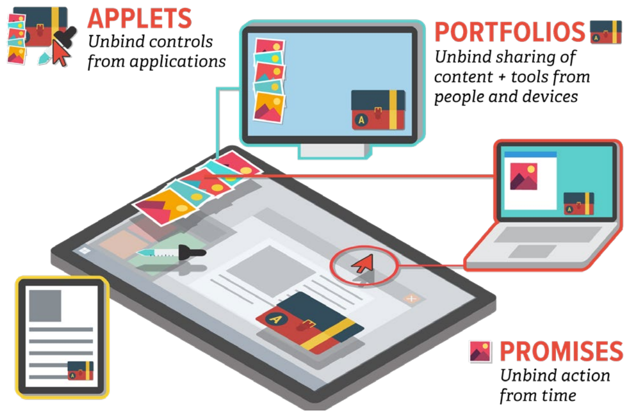
APPLETS
Unbind controls from applications
PORTFOLIOS
Unbind sharing of content + tools from people and devices
PROMISES
Unbind action from time

Figure 1. SurfaceFleet unbinds UI elements from not only the device but also the current application, user, and time. In the visible UI, *Applets* unbind controls from applications. *Portfolios* unbind tools, inputs, behaviors, and content from the current device and user. *Promises* unbind actions from time.

different device form-factor, the presence of a collaborator, or the availability of the pieces of information needed to complete a particular task. But the problem is that—if we consider *place* in such a general manner—these transitions come at a high cost, in the currencies of both application development and user experience.

SurfaceFleet (Fig. 1) is a working system that addresses these challenges by de-coupling UI elements and operations from a particular device. This system (for Microsoft Windows) includes a resilient distributed systems foundation [24], a preliminary toolkit, and a reference implementation of interaction techniques that unbind interaction across multiple dimensions of mobility in information work. So at a high level our work contributes a new way of thinking about, designing, and building distributed interactive systems.

Yet, once one decouples user interface mechanisms from the current *device*, this also has interesting carry-on implications for unbinding interaction from the current *application*, the current *user*, and the current *time*, as well. SurfaceFleet handles transitions in place—bridging the resulting gaps—across all four of these dimensions.

† The first two authors contributed equally to this work

While these gaps have long been established in Ubicomp, CSCW, and "Society of Devices" research [4, 11, 22, 26, 42, 59, 78], SurfaceFleet provides a unified interface around a small number of concepts to bridge device, application, user, and time—all at once. This integrative contribution—part design probe, part reference implementation, and part systems proof-of-concept—suggests that our approach can address a variety of cross-device usage scenarios that entail crossing one or more of these four bridges.

While some of our scenarios involve collaboration, the type of computer-supported cooperative work afforded by current on-line document sharing systems is not our core contribution. Indeed, instead of sharing entire documents, our techniques instead focus on supporting distributed user operations—tools, inputs, content, and behaviors—that can be combined with or act upon documents.

For example, interactive tools and inputs include the system clipboard, a color picker, a camera stream, or a mouse telepointer. Individual pieces of content include images, passages of text, or color palettes that can be dragged and dropped into documents. But even groupings of multiple objects—for example, placeholders for three images—can be collected in a *Container*, which visually resembles a splayed-out sheaf of papers. Any object—content, tools, or Containers—can be shared across one's own devices (or with a collaborator) via a *Portfolio* (Figure 7). A Portfolio is a distributed-interface object akin to an art portfolio case—a place where an one carries work of mixed media, of various sizes and types, and their tools of the trade. Content, Containers, and Portfolios are all reified in the UI through *Applets*—draggable, semi-transparent UI elements that remain always-on-top of the window manager, available as the user switches between different programs or web pages.

Finally, *Interaction Promises* offer novel behavior by acting as a reference to future content, such as a placeholder for an on-site photo that has not yet been taken. These are encapsulated as cross-device SurfaceFleet objects that users can likewise collect, share, and drag & drop into documents.

Our main contribution is the concepts behind the SurfaceFleet system itself, which leverages a robust and performant distributed system foundation [24], raising novel implications for migration of user experiences across multiple dimensions of "place." To illustrate the potential of this approach for a variety of usage scenarios, we explore novel interaction mechanisms including Applets, Portfolios, and Interaction Promises. We also present a preliminary toolkit for authoring SurfaceFleet applications in C# without deep prior expertise in distributed systems. Overall, our work offers a unifying conceptual contribution through its framing of mobility as transitions in place in terms of device, application, user, and time—and the resulting exploration of techniques that simultaneously bridge all four of these gaps.

FOUR CORNERSTONES OF UNBOUNDEDNESS

Here, we further unpack these four dimensions of device, application, user, and time, showing how they can act in tandem via direct manipulation, such as in the concrete Usage Scenario that follows.

#1. Device Unbound. Applets run on individual devices, but the underlying system preserves all updates to program state in a durable log via the Azure cloud [24]. Thus, migration of user activity from one device to another is a special case of fail-over to a new machine. But the same durable logging mechanism affords highly performant synchronization, also enabling parallel (multiple device) experiences. This allows the unbinding of interface elements from a particular device.

#2. Application Unbound. Rather than replacing users' existing applications, our strategy is to interoperate with (and span across) them, via Applets layered above other content. But unlike techniques such as ToolGlass [12], Tracking Menus [21], or translucent patches [49], *Applets* are independent executables that are not bound to a particular program, window, or the walled garden of a web browser; rather, they float above the operating system shell, and its applications—and conceptually, even the device itself.

#3. User Unbound. Since *Applets* roam across devices, they afford connections between people—multiple users on multiple devices—as well. Yet in current practice, many of the tools one uses for collaboration differ sharply from the everyday tools used for individual work. Hence, we adopt the design stance that *individual tools are collaborative tools, and vice versa*. One can set aside an ephemeral piece of information for later use in one's individual work—via exactly the same interaction mechanisms used to pass a screen grab to a co-located (or even remote) collaborator.

#4. Time Unbound. Another consequence of our approach is support of both synchronous and asynchronous interactions. Once there is a deterministically replayable log of distributed state, application logic does not necessarily have to "fail-over" immediately. Likewise, finer-grained interactions can also be left latent, to migrate or synchronize at a later *time*.

Acting in Tandem. These four cornerstones of unboundedness can act in tandem, allowing users to selectively share objects, or defer actions, until opportunity arises on the device with the desired resources, at the right time and the right place. For example, a user could work with a collaborator to gather the desired information, from a suitable application, on another device, and at a later time. In addition—and similar to Koorsgard's description of a place-centric approach to computing [48]—this affords fluid transitions between different configurations in everyday situations and evolving changes over time.

Via Direct Manipulation. Graphical user interfaces rely heavily on direct manipulation of visually-represented objects on a single device. Yet for distributed work, in current practice the necessary documents or pieces of information are often invisible and out of sight—lost and fragmented [13, 66] across a Borgesian labyrinth of synced folders, downloads, devices, and web services. This sharply diverges from natural human ways of organizing—such as collecting reference images in a sheaf nearby, or sharing a clipping from an article just by passing it to a colleague. This suggests that it might be fruitful to tackle some aspects of distributed interaction by reifying them as objects and instruments [7, 8] for "drag & drop" direct manipulation.

Our work also resonates with Beaudouin-Lafon's compelling (but brief) *Unified Principles of Interaction* that support ubiquitous sharing, distributed interaction, and pieces of content, while elevating interactions (i.e. tools) to first-class objects [9]. We argue that by unbinding interaction from the aforementioned cornerstones, we provide a flexible environment where users can work across devices, migrating tools and content between applications and tasks—by working collaboratively, and at a time that suits them best. Ultimately, users choose a configuration that fits their workflow, allowing them to fulfill their tasks.

Usage Scenario

To give a more concrete impression of how SurfaceFleet looks and feels, the following usage scenario (Figure 2) illustrates how Applets combine (see also our video figure).

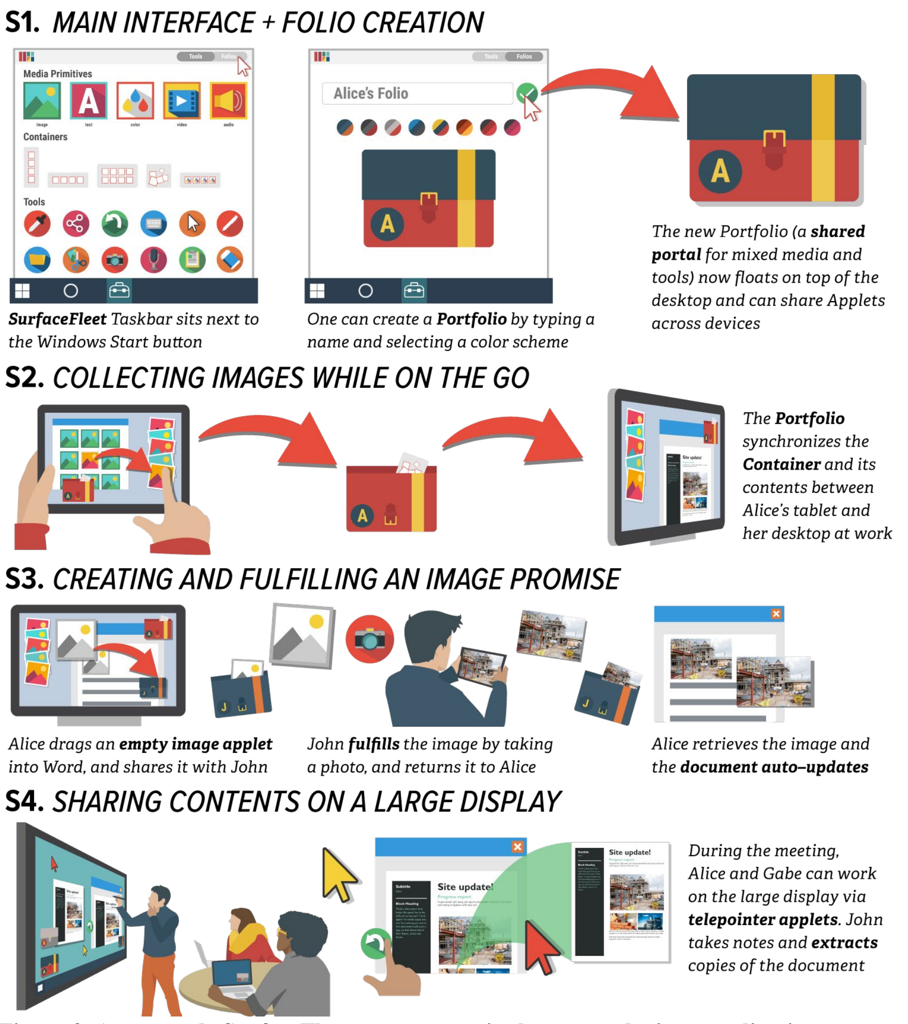
S1. MAIN INTERFACE + FOLIO CREATION
Media Primitives
Containers
Tools
Tools
Folios
Alice's Folio
SurfaceFleet *Taskbar sits next to the Windows Start button*
One can create a ***Portfolio*** *by typing a name and selecting a color scheme*
The new Portfolio (a ***shared portal*** *for mixed media and tools) now floats on top of the desktop and can share Applets across devices*
S2. COLLECTING IMAGES WHILE ON THE GO
The ***Portfolio*** *synchronizes the* ***Container*** *and its contents between Alice's tablet and her desktop at work*
S3. CREATING AND FULFILLING AN IMAGE PROMISE
Alice drags an ***empty image applet*** *into Word, and shares it with John*
John ***fulfills*** *the image by taking a photo, and returns it to Alice*
Alice retrieves the image and the ***document auto–updates***
S4. SHARING CONTENTS ON A LARGE DISPLAY
Site update!
During the meeting, Alice and Gabe can work on the large display via ***telepointer applets****. John takes notes and* ***extracts*** *copies of the document*

Figure 2. An example SurfaceFleet usage scenario that spans devices, applications, users, and time—via *Portfolios* (S0), *Containers* (S1), *Interaction Promises* (S2), and *Tools* (S3).

Alice is an architect at a small firm, currently working on a critical report ensuring that her building design is being constructed properly at a far-flung construction site:

S1. Create a Portfolio. *Alice launches the SurfaceFleet taskbar from the system tray, where she can create Media Primitives, Containers, Portfolios, and Tools. She creates a Portfolio—an Applet that passes content across devices. The Portfolio floats on top of her window manager, always accessible via drag & drop from other programs or Applets.*

S2. Collect Images On the Go. *On the train, using her tablet, Alice creates a Container from the SurfaceFleet taskbar. She looks through her folder of site photos and drags & drops the images she wants into the Container. Alice drags this Container into a Portfolio that she created. Later, in her office, she opens it from her desktop, giving access to the shared Container with the images collected on her tablet.*

S3. Create & Fulfill a Promise. *At her desktop, Alice realizes she doesn't have a photo of the new building's entrance. Alice creates an empty placeholder and inserts it into her document. Through a Portfolio, she shares it with John, a co-worker on-site at construction. That afternoon, John takes a photo of the new entrance to fill the placeholder. When Alice retrieves the image from the Portfolio, her document updates, fulfilling the Promise.*

S4. Share Contents on a Large Display. *In a meeting room with a large display, Alice discusses the report with two colleagues. From a social distance, they can each pass their mouse pointer to the large display through a Portfolio. Alice then stands to present, while her colleagues tele-point to indicate parts of the document they have questions about. Alice uses the Extract tool to take a snapshot of the current page. Using a pen, she then annotates the document as her colleagues indicate areas of concern. She captures the mark-up by Extracting the page once again, and for comparison she drags the two snapshots back into a shared Portfolio.*

RELATED WORK

SurfaceFleet builds on systems and techniques for transitions across devices, applications, and individual vs. collaborative work—or for deferring actions in time. We then contrast our approach with existing on-line sharing services (Figure 3).

Devices Unbound: Cross-Device Interaction

In the cross-device design space of Brudy et al. [15], SurfaceFleet supports one or many people and devices;

synchronous or asynchronous interactions; fixed and ad-hoc dynamics; personal and social scales; and remote or co-located use. Many previous cross-device techniques depend on spatial engagement [54], such as the spatially updated drag-and-drop regions offered by Relate Gateways [23]. But this requires a "smart room" with full spatial sensing, excluding remote users—or indeed anyone who steps outside the room. Our work forgoes spatial sensing to more flexibly support co-located *and remote* interactions [5, 27, 46].

Conductor [32] supports ad-hoc chaining of devices and cross-device relationship management, in single-user / multi-device scenarios with small tablets. Panelrama [80] partitions the 'panels' of a web UI (such as the elements of a YouTube video player) across devices. Treating multiple devices as a sort of multi-monitor [29] offers another approach [2]. SurfaceFleet's Applets adopt this notion of distributing small, self-contained pieces of functionality, but does so in ways that also afford multi-user/multi-device collaboration.

Cloud-Capable UI Elements as First-Class Objects

While SurfaceFleet's Applets provide useful functionality even when running on a single device, they inherently support migration of program state across devices, if and when desired. Existing cross-device HCI systems and toolkits tend to emphasize spatial proximity [37, 53, 69], gesture input / recognition [19, 38], or testing [57, 58] for the toolkit's level of abstraction. In SurfaceFleet we focus on providing shared state abstractions for distributed systems.

Other toolkits explore how to persist information across clients, often making use of a shared dictionary (e.g. GroupKit [64]) or a shared Document Object Model (DOM) on the web, as realized by Webstrates [46]. Under these representations, UI elements can update their information when a shared model changes. We apply a similar technical notion, but bring it to the level of user interface elements in native applications, with state shared at the C# language-binding level through a principled and scalable database architecture [24]. Hence, in SurfaceFleet, cloud-capable UI elements and behaviors are first-class objects.

Apps Unbound: Across the OS and Existing Programs

HCI systems research often leverages and repurposes existing infrastructures. This empowers end-users to combine tools, customize, and achieve new effects [25, 50, 61]. Similar considerations arise from field studies of knowledge work, such as the observation that document management tools should be "integrated in the current working environment of the user" [14]. SurfaceFleet uses Applets that float above the window manager, allowing them to co-exist with the everyday applications and OS features that knowledge workers already use for productivity. By contrast, a web solution is bound to a single application—the browser—and hence largely walled off from rich OS features and other running programs.

Xerox's classic Rooms metaphor [35] supports multi-tasking across sets of applications, including carrying certain windows across Rooms as *Baggage*. More recently, activity-centric systems explore multi-tasking support for activity roaming, suspension and resumption, and activity sharing [6, 76]. SurfaceFleet's Applets share some similar motivation but focus on mechanisms to extend simple actions across devices, while unbinding them from users and time, as well.

Users Unbound: from Individual to Collaborative

Mobility is a key attribute of collaboration [51]—both in terms of *space*, and the social notions of *place* that people make of it [34]. People shift between group and individual activities, needs for information-sharing change, and small groups come and go [72]. Yet the technological tools for collaboration differ from those used for individual work, making transitions from Human-Computer to Human-Human interaction costly [18] —in part because the tools are less familiar. Single display groupware [73] offers an example of reducing such costs: a session that starts on a single user's display readily transitions to multi-user activity.

In collaborative systems, the dimension of *user* is of course implicit, since they address multi-user and not individual work. But remote collaboration often requires asynchrony, and hence ways to unbind *time*. For example, the MATE collaborative writing system [33] allows one user's mark-up gestures to be acted upon by a colleague at a later time. Likewise, Portholes [20] provide background awareness of when remote users are present, affording notions such as meeting "when everyone is available" [18].

More generally, SurfaceFleet expands the notion of *place* in the classic time/space matrix of collaboration [4, 42]. For example, mixed-presence groupware [74] addresses the *same time / same place* and *same time / different place* quadrants of this matrix. But SurfaceFleet calls out the dimension of *user*—individual or collaborative—while also raising *device*, *application*, and *time* as cornerstones of unboundedness. This shows how interactions and system abstractions that unbind activity from a particular device can also serve to unbind other dimensions of *place* in mobility.

Time Unbound: Deferred Action, Multiple Fulfillment

Beyond the *same time / different time* distinction of the time/space matrix [42], other work explores going back in time [62, 79]. But the ability to defer certain actions to future time also could be valuable because knowledge workers often must cope with uncertainty, or missing information.

For example, people often can't sensibly file new materials because their future role or utility is still unknown [44]. Our *Container* and *Portfolio* Applets in particular extend designs for gathering pieces of encountered information [56] during active reading, via multi-object visual clipboards [36, 63, 75], to span devices and multiple users. But in particular, SurfaceFleet includes ways to defer actions (user decisions) to a later point in time—or even *more than one point in time*, such as through multiple fulfillment of Interaction Promises.

Existing Online Sharing Apps and Web Services

Current sharing apps (Fig. 3) address aspects of distributed work, but these solutions are siloed (in a single app or the browser)—and focus on folder sync, or sharing entire files.

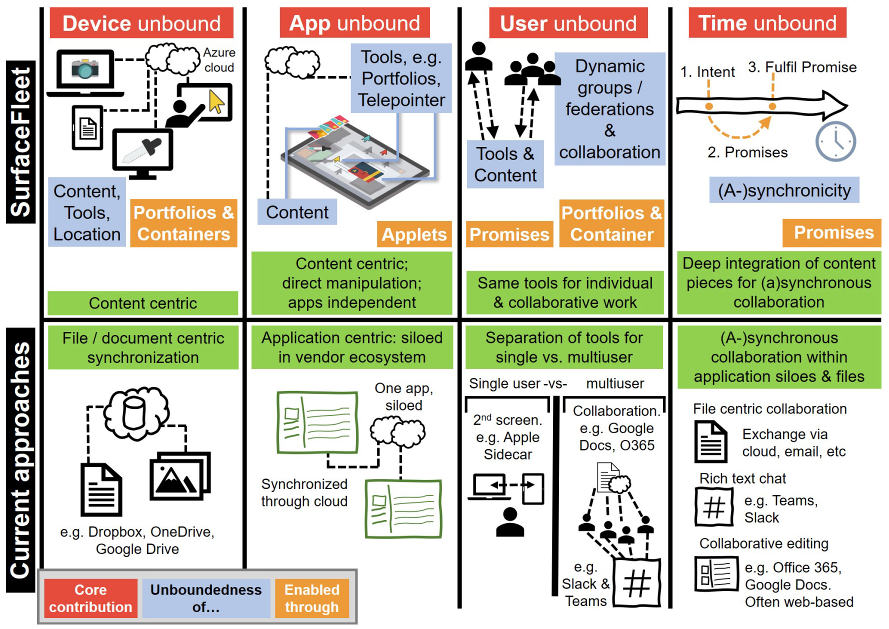
SurfaceFleet
Current approaches
Device unbound
App unbound
User unbound
Time unbound
Azure cloud
Content, Tools, Location
Portfolios & Containers
Content centric
Tools, e.g. Portfolios, Telepointer
Content
Applets
Content centric; direct manipulation; apps independent
Dynamic groups / federations & collaboration
Tools & Content
Promises
Portfolios & Container
Same tools for individual & collaborative work
1. Intent
3. Fulfil Promise
2. Promises
(A-)synchronicity
Promises
Deep integration of content pieces for (a)synchronous collaboration
File / document centric synchronization
e.g. Dropbox, OneDrive, Google Drive
Application centric: siloed in vendor ecosystem
One app, siloed
Synchronized through cloud
Separation of tools for single vs. multiuser
Single user -vs- multiuser
2nd screen. e.g. Apple Sidecar
Collaboration. e.g. Google Docs, O365
e.g. Slack & Teams
(A-)synchronous collaboration within application siloes & files
File centric collaboration
Exchange via cloud, email, etc
Rich text chat
e.g. Teams, Slack
Collaborative editing
e.g. Office 365, Google Docs. Often web-based
Core contribution
Unboundedness of...
Enabled through

Figure 3. Surface Fleet unbinds Device, App, User, and Time in a way that unifies all four of these cornerstones, and that complements existing on-line sharing tools.

Our contribution is integrative, addressing multiple aspects with a few UI concepts. For example, Dropbox, OneDrive, and GoogleDrive are document-centric archives. They focus on cold storage of entire documents, synchronizing established folders and files, rather than transient pieces of content (or tools) in active use. Slack and Teams focus on messaging, with files or images dragged into threads. But these are still silos: one must switch to Slack/Teams to share. So sharing with collaborators requires a different interface than passing individual work to one's own devices. By using Applets that float on top of the window manager, SurfaceFleet keeps the same sharing affordances always available, even as the user switches between their familiar productivity applications, documents, or web pages.

Apple Continuity [3] supports features such as handoff to another device, using a device as a second screen, or Continuity Camera to insert a picture taken by another device. SurfaceFleet contributes technical means to build these type of experiences, and shows how this can generalize to operations across four cornerstones of unboundedness.

Finally, our system is not a client with simple views into the data on each device. Rather, SurfaceFleet hosts "rich clients" where the shared layer plugs directly into application state at the C# language binding level. Hence, durable shared state via Azure, in a principled distributed system architecture, is an integral part of our functionality.

Summary

Each Applet in the SurfaceFleet system is an independent executable, with a cloud connection to log shared model updates in a robust and durable manner. Giving each Applet a concrete visual representation that floats above the window manager reifies these concepts [8] for rich instrumental interaction [7]. Activity starts on a single device, yet individual tools can serve as collaborative tools—and vice versa—while also affording the deferral of actions in time. Unbinding Device, Application, User, and Time each have precedents, but SurfaceFleet is the first distributed system to put all four of these cornerstones into action simultaneously, for both tools and pieces of content, using just a few cross-device toolkit abstractions and interactions.

SURFACEFLEET SYSTEM & TOOLKIT

Before discussing SurfaceFleet's interaction techniques in more depth, we first detail the technologies that comprise our system and toolkit. SurfaceFleet runs on Windows and is implemented in C# using the .NET Framework and Windows Presentation Foundation (WPF). Major components include a shared model, robust logging of updates, and OS and application-interop features (Figure 5).

Device Unbound: Migration as Cross-Device Fail-Over

SurfaceFleet is founded on principled distributed systems techniques, so resiliency is built into our system—unlike HCI toolkits and demos that use ad-hoc TCP connections, or UDP streaming, for example. We use a distributed client-server architecture, where each client keeps a local copy of shared state. Changes to the local state synchronize in real-time, with the state updating (recovering) on a new device as soon as a connection is established. Clients within a federation can communicate through the Azure cloud, which they access via a federation-specific connection string needed for Shared Key authorization.

We built a custom infrastructure on top of Ambrosia [24]. Ambrosia (available via open source) uses declarative database techniques to persist data, providing virtual resiliency by capturing state changes in a deterministically replayable log. This is done at the C# language level in a durable, failure-resilient, and performant manner via Azure.

Ambrosia utilizes a component known as CRA (also open-source) [65] that virtualizes connection management. Virtualization of inter-device connections combined with virtual resiliency of state changes makes our system not only robust to IP address changes, but also facilitates migration of user interface operations from one device to another. As long as clients can access the cloud they can (re)synchronize application state—essentially turning a transition from one machine to another into a *cross-device fail-over*.

Developers don't have to write extra program logic to handle complex distributed failure cases. For Serializable data types, SurfaceFleet wraps these robust foundations for shared state at the C# language level through C#'s Attributes feature, which enables querying of program entities at runtime without the need for any compiler modifications or additional tools. Developers simply need to annotate variables as `[Synchronizable]` (Figure 4). The robust sharing of state across multiple devices is entirely managed on behalf of the developer.

```
[Synchronizable]
public Color CurrentColor
{
    get
    {
        return (Color)this.Connection.SharedModel["CurrentColor"];
    }
    set
    {
        this.Connection.SharedModel["CurrentColor"] = value;
    }
}
//subscribe to update events of remote variable
Connection.SharedModel.ModelUpdated += OnModelUpdated;

private void OnModelUpdated(object sender, ModelChangedEventArgs e)
{
    if (e.PropertyName == "CurrentColor" && this.isDisplayOnly)
    {
          this.Container.Background =
                     new SolidColorBrush(this.CurrentColor);
    }
}
```

Figure 4. Tagging a variable with the `[Synchronizable]` Attribute makes it available across devices. Subscribing to `ModelUpdated` events triggers callbacks for changes in value.

Developers can subscribe to update events for state changes on remote variables—including single objects, lists, or dictionaries—and receive a callback in response (Figure 4). The SurfaceFleet toolkit supports many basic data types, images, colors, lists, dictionaries, and so forth, but developers can extend these mechanisms to arbitrary objects by annotating their own classes with a C# `DataContract`.

Application Unbound: Cross-Application Functionality

Knowledge work is not siloed within any single "sharing" or "messaging" app. To achieve Application Unboundedness, SurfaceFleet's UI adopts strategies that meet knowledge workers where their activity occurs, even as they task-switch among many applications.

Semi-Transparent, Always-on-top Applets: Existing on-line sharing apps tend to be siloed in the web browser, or a single application. But SurfaceFleet's semi-transparent Applets float above the window manager, making them *always visible* and *always available* as drag & drop targets, no matter the current application, web page, or file system window.

Multiple Formats in Clipboard. The Windows Clipboard can hold information in multiple formats. SurfaceFleet takes advantage of this by simultaneously placing multiple standard formats—as well as internal data formats—on the system clipboard. This allows SurfaceFleet to share rich objects across internal components—or standard formats with external applications—using the same mechanisms.

Drag & Drop Events: Users can drag & drop by mouse, pen, or direct-touch to pass these rich data formats amongst Applets and unmodified applications such as Word or Adobe Illustrator. Dragging a SurfaceFleet image primitive, for example, adds data in three formats: *(i)* the path to a copy of the image in the file system; *(ii)* a bitmap in multiple formats to enable rich feedback and compatibility with unmodified applications; and *(iii)* an internal format that passes an ID to native SurfaceFleet components such as Portfolios and Containers—or for use in behaviors such as Interaction Promises.

SurfaceFleet Plugins & Component Object Model (COM): When the user drops a SurfaceFleet object onto an unmodified external application, we check whether we can access its COM APIs. This requires building a SurfaceFleet plugin to handle the COM interfaces for each external application; we currently implement plugins for Word, Illustrator, Photoshop, and PowerPoint integration. But the SurfaceFleet toolkit includes generalizable abstract classes that enable developers to add support for new applications in a straightforward manner. If SurfaceFleet supports the external application, it performs an action appropriate for the given data type, such as inserting a photo, filling the selected shape with a color, or creating an Interaction Promise. Alternatively, developers can modify an external application

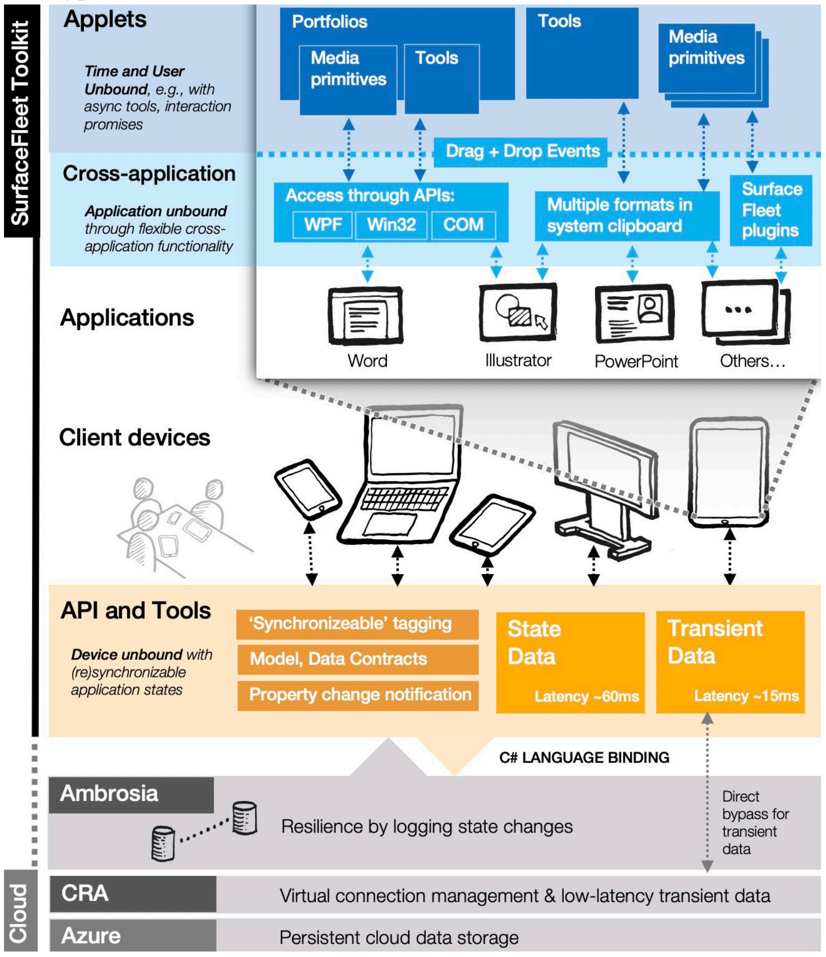
SurfaceFleet Toolkit
Applets
Time and User Unbound, e.g., with async tools, interaction promises
Portfolios
Media primitives
Tools
Tools
Media primitives
Drag + Drop Events
Cross-application
Application unbound through flexible cross-application functionality
Access through APIs:
WPF
Win32
COM
Multiple formats in system clipboard
Surface Fleet plugins
Applications
Word
Illustrator
PowerPoint
Others…
Client devices
API and Tools
Device unbound with (re)synchronizable application states
'Synchronizeable' tagging
Model, Data Contracts
Property change notification
State Data
Latency ~60ms
Transient Data
Latency ~15ms
C# LANGUAGE BINDING
Ambrosia
Resilience by logging state changes
Direct bypass for transient data
CRA
Virtual connection management & low-latency transient data
Azure
Persistent cloud data storage
Cloud

Figure 5. System, toolkit, and underlying technical components of SurfaceFleet.

to access our custom clipboard data format directly. Meanwhile, for legacy applications we default to clipboard formats such as file paths, bitmap images, or text strings.

Time Unbound: Interaction-Driven Promises

Unbinding actions from time is fundamental to our system's technical underpinnings. For example, when a device joins a federation, it deterministically replays shared model updates from the session. Hence, SurfaceFleet can migrate activity to another device immediately, or at a later time, or even revisit past states so long as they remain available in the log. But at present, we expose Time Unboundedness in the user interface through Interaction Promises, which allow users to insert placeholders for content that is not yet available.

To realize Interaction Promises with external applications, we use COM APIs to insert & fulfill Promises. For example, when creating a Promise in Word, we use COM APIs to insert an invisible bookmark (with a unique ID) spanning a range selection including the placeholder image. At a future time, we can scan the document for the ID and replace the content, again using COM APIs. Here, we simply use the caret position within the document's text-flow to insert the invisible bookmark, plus a placeholder image, which we can then replace later. Alternatively, developers can implement their own plugins to check for Promise fulfillments—even when our application is not running, or the document has been closed. For example, we created a custom plugin for Word, which checks opened files for an inserted Promise, with replacement of the content if needed. This would further allow insertion of longer blocks of rich content that can be manipulated within the document—as well as externally.

SurfaceFleet Tools offer another example of app-native support for time-unbound Interaction Promises. For example, the color picker can accept a selection (shape) dragged from Adobe Illustrator. The system keeps a reference to this shape, and updates the selection's fill color whenever the user samples a new color with the color picker.

User Unbound: Social Protocol and Organizational Trust

Conceptually, SurfaceFleet's interactions are unbound from any particular user. For example, the Promise replacement noted above also works in a collaborative scenario, where one user can insert an image placeholder, then share with a collaborator via *Portfolio*. The collaborator can then fulfill it (e.g. by taking a photo with their device's camera), and share back through the Portfolio. We rely on social protocol for users to learn of shared Portfolios, or to avoid conflicts such as sharing one physical keyboard to two different devices.

But as stated previously, our core contributions do not revolve around collaboration, and as such we presently do not implement the rich heritage of known people-centric interaction and feedback techniques available in the CSCW literature. For example, our system currently does not show which users are connected, or provide feedback of who has collaborative access. Rather, our goal here is to show that our technical architecture and the interface mechanisms we explore have implications for collaborative scenarios as well.

At a technical level, we require SurfaceFleet clients to run on an organizationally approved version of the operating system image. This ensures that untrusted, unknown, or possibly malicious devices outside of an organization cannot join SurfaceFleet federations. To verify and enforce this, we implemented Windows device attestation via the CPU's built-in Trusted Platform Module (TPM) hardware to cryptographically ensure device compliance. Devices that pass this attestation receive an authorization key, enabling access to a federation's shared state on the Azure cloud.

INTERACTION TECHNIQUES IN SURFACEFLEET

At a high level, the various concepts realized in the user interface of SurfaceFleet are intended as technology probes [40]. We believe our toolkit affords many new opportunities for cross-device interaction, which we explored by building and reflecting-in-action [67] upon a set of interdependent techniques. Each of these techniques (or "probes") represent meaningfully distinct examples that go beyond single instances by collectively exploring a class of techniques—and that illustrate the compound, integrated [17] workflow of knowledge work that spans multiple surfaces [66, 68].

At present, we make no strong claims as to whether SurfaceFleet and the particular probes realized in its Applets make for "better" distributed work or not. And the level of development is not yet such that we can deploy our techniques longitudinally for real work. Rather, these techniques are intended to probe and demonstrate some of the interesting technical and interaction possibilities afforded when one approaches Society-of-Devices experiences from a principled distributed systems foundation.

Applets and the SurfaceFleet Taskbar

Applets let SurfaceFleet offer user interface objects across the window manager and other applications. Applets are independent executables, visible as compact regions that float above the window manager. The user can reposition them as desired, or dismiss when no longer needed.

Applets are semi-transparent by default. This allows partial visibility of underlying content or program windows. But when the user touches an Applet (or hovers over it with the mouse or a pen-tip), it fully materializes, becoming opaque; moving away then fades back to a semi-transparent state. Hence Applets are always visible, always on top, and always available for drag & drop from any program or web page.

Through these Applets, SurfaceFleet UI strives to make cross-device interactions *visible* and *local*. The legacy of Xerox Star conventions such as folders, icons, and generic verbs [43] shows the power of directly manipulating visual metaphors [28, 41, 70]. The *visibility* and *locality* [77] of techniques such as Local Tools [10], KidPad [39] and HabilisDraw [16, 71] show how making a tool's meaning, state, and parameters apparent provides awareness and affordances [30, 60] for interaction. Related techniques such as Tracking Menus [21], Translucent Patches [49], and ToolGlass [12] all use floating tools to interact with content in rich ways. Such reification [8] allows domain objects to combine with interaction instruments [7]—an approach also well-suited to multi-surface environments [47].

SurfaceFleet's Applets adopt these strategies to interoperate with the OS, window manger, application windows, and one another through instrumental interaction. The user can launch Applets from the SurfaceFleet taskbar (Figure 6). But further interactions with floating Applets and Tools often yield Applets directly, without having to revisit this taskbar.

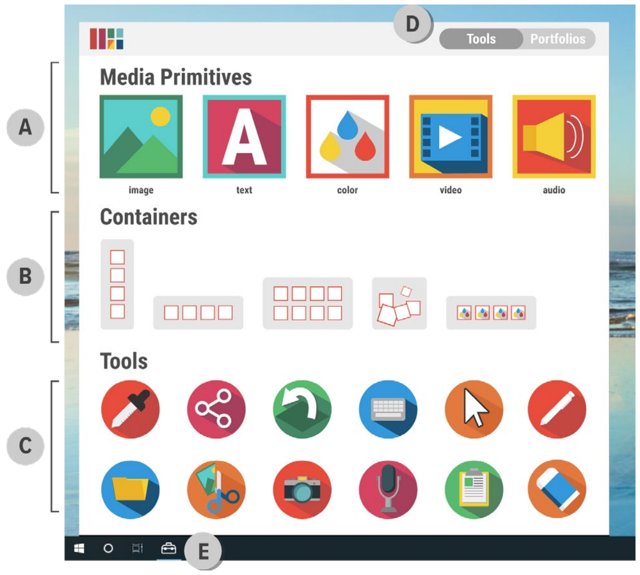

Figure 6. The SurfaceFleet taskbar can create Applets for (a) Media Primitives, (b) Containers, and (c) Tools, and (d) Portfolios. The taskbar icon (e) appears near the Start menu.

Five Mechanisms for Cross-Device Interaction

SurfaceFleet consists of five main distributed-interface mechanisms that users can employ in combination to unbind content and tools from device, application, user, and time:

1. ***Portfolios:*** Akin to an art portfolio case, this Applet functions both as a cross-device *portal*, and a creator's travel *folio*—a place to stash mixed-media content (Primitives, Containers) as well as interactive Tools.
2. ***Interaction Promises*** are placeholders for content, allowing deferral of select actions or decisions. For example, users can drag empty Media Primitives, where the contents of the images are not yet available, to create and share Promises for future fulfillment.
3. ***Tools:*** SurfaceFleet encapsulates a number of user interface operations, inputs, and interactive behaviors in special Applets known as Tools. Users can drag Tools onto content (or use them as drop targets) to achieve various effects via instrumental interaction [7, 8].
4. ***Media Primitives*** are pieces of content, such as images, that are elevated to floating Applets to make them directly actionable for cross-device use in SurfaceFleet.
5. ***Containers:*** Visually represented as a splayed-out sheaf, this Applet offers an always-available, multi-object visual clipboard that users can dock to any edge of the screen for convenient collection and curation of content.

The following sections discuss each of these in more depth, starting with perhaps the most interesting ones, Portfolios and Interaction Promises.

Portfolios: Unbinding from Devices and People

The Portfolio Applet acts like a mobile travel case, holding mixed-media objects and tools while transporting them from one "place" to another. Part portal, part stash, and part teleporter, Portfolios enable convenient transfer of SurfaceFleet objects—Media Primitives, Containers, Tools, and Promises—across devices through a common drag and drop operation. Portfolios are unbound from *devices*—once a Portfolio is instantiated, other devices in the same federation can access it. And they're unbound from *users* as well—allowing people to build habits for moving personal objects across devices for individual work, that then also apply to moving shared objects across users for collaborative work. Hence SurfaceFleet offers consistent interactions such that *individual tools are collaborative tools, and vice versa*.

When an Applet (such as an Image Primitive) is dragged over a Portfolio (Figure 7), the Portfolio wiggles to indicate that it can accept the content. Releasing the Applet places it into the Portfolio, which shows the Applet sticking out. This feedback of available items is echoed across all devices that have access to that particular shared Portfolio. People can place or retrieve contents from a Portfolio in a manner similar to how people hand off physical documents or even tools. When a person takes an item out of a Portfolio, by double-tapping, it is elevated to an independent Applet, now floating on their screen. Users can customize the label and color scheme of each Portfolio to make them distinct.

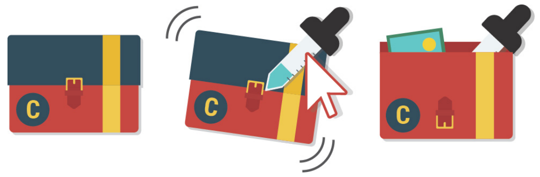

Figure 7. Portfolios share content and tools. An empty Portfolio appears closed (left). Dragging content or tools over a Portfolio causes it to *wiggle*, signaling that the Portfolio can accept the object (middle). Portfolios partially reveal their contents to provide awareness (right). A double-tap retrieves the contents.

Interaction Promises: Unbinding from Time

Interaction Promises are one of the more interesting new concepts probed by SurfaceFleet. They afford asynchronous workflows where people can delegate *pieces of content* to other devices, or collaborators, for fulfillment in the future. That is, in combination with Applets, SurfaceFleet uses these as proxies for the as-yet unavailable contents of a Media Primitive. They can also be fulfilled one or more times, such as replacing an initial image with a better option that a collaborator shares back later. Interaction Promises even support multiple fulfillment, i.e., they can return a Container with a plurality of objects as options, from which the user who initiated the promise can decide which one to select.

These types of dilemmas are common in knowledge work, where pieces of information may be ambiguous, undecided, too distracting to deal with immediately, or unavailable as work begins [44, 68]. In other cases, users may need to defer decisions to a later time, such as when they are on another device with the right resources (e.g. a camera) or content (photo collection)—or when a collaborator is ready to help.

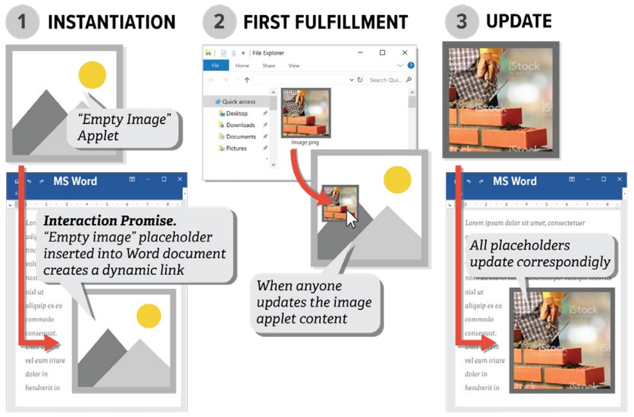
1 INSTANTIATION
2 FIRST FULFILLMENT
3 UPDATE
"Empty Image" Applet
MS Word
Interaction Promise. "Empty image" placeholder inserted into Word document creates a dynamic link
File Explorer
When anyone updates the image applet content
MS Word
All placeholders update correspondingly

Figure 8. Interaction Promises: *1)* the user drags an empty placeholder into a document. The user can then *2)* populate it with an initial image, or 3) receive updates when the linked Applet is fulfilled from other devices, or by other users.

For example, to insert a provisional image into a Word document (Figure 8), SurfaceFleet lets users drag an empty placeholder Image Primitive Applet into their document. By also sharing this Applet with a collaborator, via a Portfolio, the collaborator can later fulfill this placeholder with image content. SurfaceFleet links all placeholders within documents to their Applet source, allowing the image to be updated when the corresponding distributed-interface object changes, whether locally, on another device, or by another user. Likewise, upon fulfillment, a single Promise can propagate to multiple placeholders in a document, as currently implemented in Microsoft Word through invisible bookmarks that contain the Applet's internal object ID (for the details of our approach see the earlier technical description of "Interaction-Driven Promises").

Tools—User Operations Across Devices & Applications

Tools are Applets that reify and encapsulate functionality for generic verbs [43], input streams, or OS-level commands that apply across multiple applications. As appropriate for instrumental interaction [7], the effect of a Tool depends on what it is applied to—hence, by drag and drop to different programs, content types, or Applets, the user can achieve various effects with a small set of Tools. For example, clicking a full Color Picker Tool over Illustrator "squeezes out" the color, onto Illustrator's selected objects.

Tools always produce a visual manifestation, either by generating a new Applet (Media Primitive) as a result, or by having a visual representation of the command applied locally. Users can then drag these representations to share the Tools (i.e. their results or behaviors) across devices via Portfolios. In this manner, not just content but also tools can be unbound from devices and passed to collaborators.

Examples of Tools as Generic Commands

Tools can have special functions that encapsulate cross-application as well as operating system behaviors. All Tools can be shared across devices via Portfolios (e.g. Figure 7).

Color Picker Tool. Resembling an eyedropper, this tool retrieves the color of a screen pixel from any device. When full, the color can be squeezed out onto other Applets or running programs. For example, a designer can use the Color Picker to drop multiple colors into a Container, which creates color chips, e.g. to curate the color palette for a brand design.

Camera Tool. A device's camera can connect to a Media Primitive or Container. When the user snaps a picture, the photo replaces the Primitive, or adds to the Container.

Tele-point & Tele-type. In co-located, shared screen scenarios, users can pass their mouse cursor to another device. Each user's telepointer appears with a distinct color. For clicks, SurfaceFleet injects multi-touch events, allowing each user to drag objects and interact. Likewise, users can pass the Tele-type Tool to a device lacking a keyboard.

Screen Grab. The user can lasso a portion of the screen using a pen, touch, or mouse, resulting in a nonrectangular Image Primitive. This makes it easy to grab a piece of encountered content [56] on one device and share it back to another.

Extract Tool. The Extract tool is similar to Screen Grab, but grabs rectangular content within an application window (such as to collect the currently visible page of a document for mark-up), and elevates it to an Applet for collection and sharing via Containers and Portfolios.

Clipboard. Users can link this Tool to an Applet, such as a Container, to collect objects copied to the system clipboard (further detailed below). The Clipboard Tool's icon remains visible, providing feedback of the active link until dismissed.

Media Primitives—Content Unbound from Applications

The Media Primitive is a base Applet that reifies a single piece of content. Primitives remain local, unless shared to another device or user. We currently support images, rich text, and colors; video and audio are planned additions. In particular, Media Primitives allow drag & drop with:

– *Unmodified Programs,* for insertion into Word, Illustrator, PowerPoint, Photoshop, File Explorer, the Desktop, etc.;
– *Portfolios and Containers*, for collection with other objects and sharing across devices and users; and
– *Any Other Applet,* to serve as the operand of the function appropriate to the drop-target.
– *Tools / Returned as Results.* Media Primitives also emerge as results from other Applets, such as when applying a Tool, or taking shared content out of a Portfolio.

Like other Applets, Media Primitives float on top of the window manager, where users can reposition, partially overlap, or otherwise arrange them freeform. Thus pieces of content in active use remain visible and readily at hand, in a "spatial holding pattern for current inputs and ideas"[44]—much like scraps of paper on a physical desk. Our intent is to allow the "intelligent use of space" [45] typical in knowledge work, such as to structure task steps, remind of important information, and afford juxtaposition of ideas [31, 68].

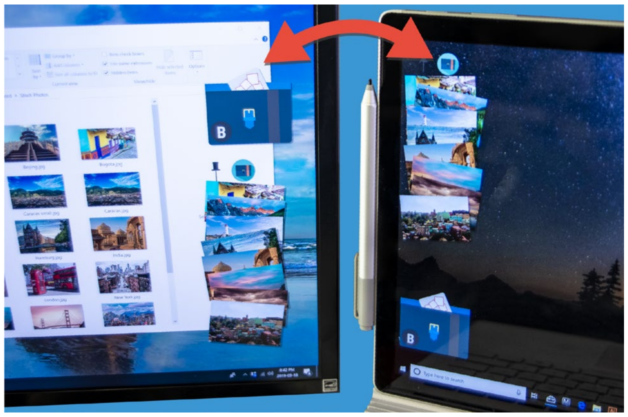
Figure 9. A Container synchronized across devices via a Portfolio. Any items added appear on the other device.

Containers: Collections of Media Primitives

A Container (Figure 9) is a floating Applet that aggregates and curates a set of media primitives, which appear by default as a fanned-out sheaf of items. Containers let users arrange, reposition, or pin up a set of objects (such as images) as a unit, much like one would place a stack of papers in a task-appropriate position on a physical desk [52]. Containers support vertically stacked, horizontally stacked, grid, and freeform arrangements of content (Figure 6b). They can even have a set number of items, and of a certain type—such as a Container with placeholders for three images that a user might pass to a collaborator, to populate with on-site photos of construction from three different camera angles.

Distributing the System Clipboard via Containers. In addition to inserting media via direct drag-and-drop from Applets and other programs, users can drag the Clipboard Tool onto a Container to associate them. Then, whenever the user copies media to the system clipboard, it also appears in the Container, with a salient "pop and bounce" animation. For individual use, this preserves a history of copied items, which the user can then drag out and re-use at any time. But by passing such a Container to select devices or collaborators (via the Portfolio, as discussed below), the user creates a shared distributed clipboard. Hence this combination of Applets and Tools shows how SurfaceFleet can unbind an abstraction like the system clipboard from the current device.

DISCUSSION

Here we reflect on the design probes as currently realized in SurfaceFleet, based on our own experiences with them, as well as some preliminary pilot user feedback we've received.

Power in Combination. SurfaceFleet frames mobility in knowledge work as transitions from one place to another, where *place* is generalized across multiple dimensions of unboundedness. Hence "sharing" is not just something for files & folders, but reconceived and de-coupled into Applets and Tools. Simple behaviors then support flexible "partial sharing" of pieces of documents, and other intermediate work-objects. The resulting expressive match [61] of Applets and Tools with instrumental interactions [7, 8] lends the system power in combination. And with the unbinding of *time* afforded by Interaction Promises, a unified interface with a small number of consistent concepts can support flexible workflows for *individual* deferred actions as well as *collaborative* delegation of tasks.

Appropriate Scale in Time and Number. The interface choices made in SurfaceFleet's current design probes suit some envisioned uses, but not others. It is intended for transient work-objects in active use, not the long tail of items in cold storage. A single active project, not a long-term archive of many. Small-group settings, not large meetings. Semi-private sharing among trusted peers, not open and (potentially untrusted) public participation. For example, the freeform, informal, and arguably more human way of organizing content afforded by a "messy desk" [1, 14, 44, 45] might not scale to hundreds of Applets left lying around on top of the window manager. However, these choices are not fundamental to our four cornerstones of unboundedness, our robust distributed-system foundation, or our toolkit, which could all be used to probe many other possibilities along this spectrum of choices in the future.

Feedback and Awareness. Perhaps the main weakness of our current visual interaction design is that certain aspects of state, such as which other devices or users (if any) a Portfolio is currently shared with, lack feedback. This might be as simple as showing pictures on-hover of who a Portfolio is shared with. It could also involve more animations such as the Portfolio's existing "wiggle" to signal it can accept drag-and-drop (Figure 7). And more generally, awareness of nearby devices or persons is lacking in SurfaceFleet. Such feedback could allay potential concerns of sharing something private by dragging it to the wrong Applet, for example.

Toolkit Availability and OS. The technical foundation of our system is complex and involves a number of layers that would need to be better packaged to make them usable and maintainable going forward. Nonetheless our medium-term intention is to release SurfaceFleet for open source. Further, developers curious about these directions can directly build on top of Ambrosia [24] and CRA [65]. For example, we are currently investigating the feasibility of building distributed interactions for JavaScript and Android via these layers.

CONCLUSION AND FUTURE WORK

Our work articulates a new way of thinking about mobility as transitions from one place to another. In particular, we generalize the notion of *place* to four cornerstones of unboundedness: device, application, user, and time. At present, SurfaceFleet just scratches the surface of these multi-dimensional gaps in cross-device interaction, and much work remains to be done to support them more fully. But collectively our existing toolkit and design probes of interaction techniques already show much potential.

We are also keen to develop more aspects of the system to a level where they could be deployed for real work—and for longitudinal studies. These could surface new issues and challenges in our multi-device, multi-user world (even if largely in the form of remote collaboration in this pandemic).

Many other issues remain to be explored by future work. For example, our present system does not attempt to sense or discover nearby devices and services for implicit or semi-automatic formation of device federations [26, 55]. We are especially interested to pursue techniques that exploit sensing on devices and semi-fixed features such as tables and displays, with flexible treatment of interpersonal space [27]. To explore these directions, we intend to add support for distributed sensing techniques to the SurfaceFleet toolkit.

More generally, present computing trends suggest that cross-device and distributed systems will have major impact on HCI going forward. With Moore's Law at an end, yet networking and storage exhibiting exponential gains, the future appears to favor systems that emphasize seamless *mobility of data*, rather than using any particular CPU. At the same time, the ubiquity of connected and inter-dependent devices, of many different form factors, hints at a Society of Technologies that establishes meaningful relationships amongst the members of this society. This favors the *mobility of user activity*, rather than using any particular device, to achieve a future where HCI can meet full human potential.

ACKNOWLEDGMENTS
We thank Nicolai Marquardt for feedback and help with some figures. We thank the reviewers for their feedback.

REFERENCES

[1] Anand Agarawala and Ravin Balakrishnan. *Keepin' it real: pushing the desktop metaphor with physics, piles and the pen*. in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 2006. Montréal, Québec, Canada: ACM. http://doi.acm.org/10.1145/1124772.1124965.

[2] Naser AlDuaij, Alexander Van't Hof and Jason Nieh. *Heterogeneous Multi-Mobile Computing*. in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. 2019. Seoul, Republic of Korea: ACM. 10.1145/3307334.3326096.

[3] Apple Inc. *Apple MacOS Continuity: All your devices. One seamless experience.* 2020 [cited 2020 April 30]; Available from: https://www.apple.com/macos/continuity/.

[4] Ron Baecker, Jonathan Grudin, William Buxton and Saul Greenberg, *Readings in Human-Computer Interaction: Towads the Year 2000*. 1995, San Mateo, CA: Morgan-Kaufmann.

[5] Jakob Bardram, Sofiane Gueddana, Steven Houben and Søren Nielsen. *ReticularSpaces: activity-based computing support for physically distributed and collaborative smart spaces*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2012. Austin, Texas, USA: ACM. 10.1145/2207676.2208689.

[6] Jakob E. Bardram, Steven Jeuris, Paolo Tell, Steven Houben and Stephen Voida, *Activity-centric computing systems.* Commun. ACM, 2019. **62**(8): p. 72-81. 10.1145/3325901.

[7] Michel Beaudouin-Lafon. *Instrumental interaction: an interaction model for designing post-WIMP user interfaces*. in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 2000. The Hague, The Netherlands: ACM. 10.1145/332040.332473.

[8] Michel Beaudouin-Lafon and Wendy E. Mackay. *Reification, polymorphism and reuse: three principles for designing visual interfaces*. in *Proceedings of the working conference on Advanced visual interfaces*. 2000. Palermo, Italy: ACM. 10.1145/345513.345267.

[9] Michel Beaudouin-Lafon. *Towards Unified Principles of Interaction*. in *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter*. 2017. Cagliari, Italy: Association for Computing Machinery. 10.1145/3125571.3125602.

[10] Benjamin B. Bederson, James D. Hollan, Allison Druin, Jason Stewart, David Rogers and David Proft. *Local tools: an alternative to tool palettes*. in *Proceedings of the 9th annual ACM symposium on User interface software and technology (UIST '96)*. 1996. ACM, New York, NY, USA. http://dx.doi.org/10.1145/237091.237116.

[11] Richard Bentley, Thilo Horstmann, Klaas Sikkel and Jonathan Trevor. *Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System*. in *Proceedings of the 4th International WWW Conference*. 1995. IEEE Computer Society Press.

[12] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton and Tony D. DeRose. *Toolglass and magic lenses: the see-through interface*. in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 1993. Anaheim, CA: ACM. http://doi.acm.org/10.1145/166117.166126.

[13] Richard Boardman and M. Angela Sasse. *"Stuff goes into the computer and doesn't come out": a cross-tool study of personal information management*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. 2004. ACM, New York, NY, USA. http://dx.doi.org/10.1145/985692.985766.

[14] Olha Bondarenko and Ruud Janssen. *Documents at Hand: Learning from Paper to Improve Digital Technologies*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2005. Portland, Oregon, USA: Association for Computing Machinery. 10.1145/1054972.1054990.

[15] Frederik Brudy, Christian Holz, Roman Radle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose

and Nicolai Marquardt. *Cross-Device Taxonomy: Survey, Opportunities and Challenges of Interactions Spanning Across Multiple Devices*. in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019. Glasgow, Scotland Uk: ACM. 10.1145/3290605.3300792.

[16] Colin G. Butler and Robert St. Amant. *HabilisDraw DT: a bimanual tool-based direct manipulation drawing environment*. in *CHI '04 Extended Abstracts on Human Factors in Computing Systems*. 2004. Vienna, Austria: ACM. 10.1145/985921.986049.

[17] W. Buxton. *Chunking and Phrasing and the Design of Human-Computer Dialogues*. in *Proceedings of the IFIP World Computer Congress*. 1986.

[18] W. Buxton. *Integrating the Periphery and Context: A New Taxonomy of Telematics*. in *Proceedings of Graphics Interface '95*. 1995. Quebec City, Quebec, Canada.

[19] Pei-Yu Chi and Yang Li. *Weave: Scripting Cross-Device Wearable Interaction*. in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015. Seoul, Republic of Korea: Association for Computing Machinery. 10.1145/2702123.2702451.

[20] Paul Dourish and Sara Bly. *Portholes: Supporting awareness in a distributed work group*. in *ACM CHI 1992 Conference on Human Factors in Computing Systems*. 1992. New York, NY: ACM.

[21] George Fitzmaurice, Azam Khan, Robert Pieké, Bill Buxton and Gordon Kurtenbach. *Tracking menus*. in *Proceedings of the 16th annual ACM symposium on User interface software and technology*. 2003. Vancouver, Canada: ACM. http://doi.acm.org/10.1145/964696.964704.

[22] George W. Fitzmaurice, Azam Khan, William Buxton, Gordon Kurtenbach and Ravin Balakrishnan, *Sentient Data Access via a Diverse Society of Devices.* ACM Queue, 2003. **1**(8 (Nov)).

[23] Hans Gellersen, Carl Fischer, Dominique Guinard, Roswitha Gostner, Gerd Kortuem, Christian Kray, Enrico Rukzio and Sara Streng, *Supporting device discovery and spontaneous interaction with spatial references.* Personal Ubiquitous Comput., 2009. **13**(4): p. 255–264. 10.1007/s00779-008-0206-3.

[24] Jonathan Goldstein, Ahmed Abdelhamid, Mike Barnett, Sebastian Burckhardt, Badrish Chandramouli, Darren Gehring, Niel Lebeck, Christopher Meiklejohn, Umar Farooq Minhas, Ryan Newton, Rahee Ghosh Peshawaria, Tal Zaccai and Irene Zhang, *A.M.B.R.O.S.I.A: providing performant virtual resiliency for distributed applications.* Proc. VLDB Endow., 2020. **13**(5): p. 588–601. 10.14778/3377369.3377370.

[25] Saul Greenberg and Michael Boyle. *Customizable physical interfaces for interacting with conventional applications*. in *Proceedings of the 15th annual ACM symposium on User interface software and technology*. 2002. Paris, France: ACM. 10.1145/571985.571991.

[26] Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob Diaz-Marino and Miaosen Wang, *Proxemic interactions: the new ubicomp?* interactions, 2011. **18**(1): p. 42–50. 10.1145/1897239.1897250.

[27] Jens Emil Grønbæk, Mille Skovhus Knudsen, Kenton O'Hara, Peter Gall Krogh, Jo Vermeulen and Marianne Graves Petersen. *Proxemics Beyond Proximity: Designing for Flexible Social Interaction Through Cross-Device Interaction*. in *Proceedings of the CHI 2020 Conference on Human Factors in Computing Systems*. 2020. ACM. 10.1145/3313831.3376379.

[28] Jonathan Grudin, *The case against user interface consistency.* Commun. ACM, 1989. **32**(10): p. 1164-1173. 10.1145/67933.67934.

[29] Jonathan Grudin. *Partitioning digital worlds: focal and peripheral awareness in multiple monitor use*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2001. Seattle, Washington, USA: Association for Computing Machinery. 10.1145/365024.365312.

[30] Carl Gutwin and Saul Greenberg, *A Descriptive Framework of Workspace Awareness for Real-Time Groupware.* Computer Supported Cooperative Work (CSCW), 2002. **11**: p. 411-446. https://doi.org/10.1023/A:1021271517844.

[31] Joshua Hailpern, Erik Hinterbichler, Caryn Leppert, Damon Cook and Brian P. Bailey. *TEAM STORM: demonstrating an interaction model for working with multiple ideas during creative group work.* in *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition (C&C '07)*. 2007. Washington, DC, USA. http://dx.doi.org/10.1145/1254960.1254987

[32] Peter Hamilton and Daniel J. Wigdor. *Conductor: enabling and understanding cross-device interaction*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2014. Toronto, Ontario, Canada: Association for Computing Machinery. 10.1145/2556288.2557170.

[33] Gary Hardock, Gordon Kurtenbach and William Buxton. *A marking based interface for collaborative writing*. in *Proceedings of the 6th annual ACM symposium on User interface software and technology (UIST '93)*. 1993. https://doi.org/10.1145/168642.168669.

[34] Steve Harrison and Paul Dourish. *Re-place-ing space: the roles of place and space in collaborative systems*. in *Proceedings of the 1996 ACM conference on Computer supported cooperative work*. 1996. Boston,

Massachusetts, USA: Association for Computing Machinery. 10.1145/240080.240193.

[35] D. Austin Henderson and Stuart Card, *Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface.* ACM Trans. Graph., 1986. **5**(3): p. 211–243. 10.1145/24054.24056.

[36] Ken Hinckley, Xiaojun Bi, Michel Pahud and Bill Buxton. *Informal Information Gathering Techniques for Active Reading*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. 2012. ACM, New York, NY, USA. http://dx.doi.org/10.1145/2207676.2208327.

[37] Steven Houben, Paolo Tell and Jakob E. Bardram. *ActivitySpace: Managing Device Ecologies in an Activity-Centric Configuration Space*. in *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*. 2014. Dresden, Germany: ACM. 10.1145/2669485.2669493.

[38] Steven Houben and Nicolai Marquardt. *WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications*. in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015. Seoul, Republic of Korea: ACM. 10.1145/2702123.2702215.

[39] Juan Pablo Hourcade, Benjamin B. Bederson, Allison Druin and Gustav Taxén. *KidPad: collaborative storytelling for children*. in *CHI '02 Extended Abstracts on Human Factors in Computing Systems*. 2002. Minneapolis, Minnesota, USA: ACM. 10.1145/506443.506449.

[40] Hilary Hutchinson, Wendy Mackay, Bo Westerlund, Benjamin B. Bederson, Allison Druin, Catherine Plaisant, Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, Heiko Hansen, Nicolas Roussel and Björn Eiderbäck. *Technology probes: inspiring design for and with families*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. 2003. ACM, New York, NY, USA. http://dx.doi.org/10.1145/642611.642616.

[41] Robert J.K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey and Jamie Zigelbaum. *Reality-based interaction: a framework for post-WIMP interfaces*. in *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*. 2008. Florence, Italy: ACM. http://doi.acm.org/10.1145/1357054.1357089.

[42] Robert Johansen, *GroupWare: Computer Support for Business Teams*. Vol. xviii. 1998, New York, NY, USA: The Free Press. 205.

[43] J. Johnson, T. Roberts, W. Verplank, D. Smith, C. Irby, M. Beard and K. Mackey, *The Xerox Star: A Retrospective*, in *Readings in Human-Computer Interaction: Towards the Year 2000*, R. Baecker, J. Grudin, W. Buxton and S. Greenberg, Editors. 1995, Morgan Kaufmann: San Francisco. p. 53-70.

[44] Alison Kidd. *The marks are on the knowledge worker*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1994. Boston, Massachusetts, USA: Association for Computing Machinery. 10.1145/191666.191740.

[45] David Kirsh, *The intelligent use of space.* Artificial Intelligence, 1995. **73**: p. p. 31-68.

[46] Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay and Michel Beaudouin-Lafon. *Webstrates: Shareable Dynamic Media*. in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 2015. Charlotte, NC, USA: ACM. 10.1145/2807442.2807446.

[47] Clemens Nylandsted Klokmose and Michel Beaudouin-Lafon. *VIGO: instrumental interaction in multi-surface environments*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2009. Boston, MA, USA: ACM. 10.1145/1518701.1518833.

[48] Henrik Korsgaard, *Toward Place-Centric Computing : Making Place With Technology Together*. 2017, Aarhus University.

[49] Axel Kramer. *Translucent patches-dissolving windows*. in *Proceedings of the 7th annual ACM symposium on User interface software and technology (UIST '94)*. 1994. ACM, New York, NY, USA. http://dx.doi.org/10.1145/192426.192474.

[50] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg and Saul Greenberg. *Evaluation Strategies for HCI Toolkit Research*. in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018. Montreal QC, Canada: ACM. 10.1145/3173574.3173610.

[51] Paul Luff and Christian Heath. *Mobility in collaboration*. in *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. 1998. Seattle, Washington, USA: Association for Computing Machinery. 10.1145/289444.289505.

[52] T. Malone, *How Do People Organize Their Desks? Implications for the Design of Office Information Systems.* ACM Transactions on Office Information Systems, 1983. **1**(1): p. 99-112.

[53] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring and Saul Greenberg. *The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies*. in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011. Santa Barbara, California, USA: Association for Computing Machinery. 10.1145/2047196.2047238.

[54] Nicolai Marquardt, Till Ballendat, Sebastian Boring, Saul Greenberg and Ken Hinckley. *Gradual engagement: facilitating information exchange between digital devices as a function of proximity*. in *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*. 2012. Cambridge, Massachusetts, USA: Association for Computing Machinery. 10.1145/2396636.2396642.

[55] Nicolai Marquardt, Ken Hinckley and Saul Greenberg. *Cross-device interaction via micro-mobility and f-formations*. in *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 2012. Cambridge, Massachusetts, USA: Association for Computing Machinery. 10.1145/2380116.2380121.

[56] Catherine C. Marshall and Sara Bly. *Saving and using encountered information: implications for electronic periodicals*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2005. Portland, Oregon, USA: Association for Computing Machinery. 10.1145/1054972.1054989.

[57] Michael Nebeling and Moira Norrie. *jQMultiTouch: lightweight toolkit and development framework for multi-touch/multi-device web interfaces*. in *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*. 2012. Copenhagen, Denmark: Association for Computing Machinery. 10.1145/2305484.2305497.

[58] Michael Nebeling, Elena Teunissen, Maria Husmann and Moira C. Norrie. *XDKinect: development framework for cross-device interaction using kinect*. in *Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems*. 2014. Rome, Italy: Association for Computing Machinery. 10.1145/2607023.2607024.

[59] Thomas Neumayr, Hans-Christian Jetter, Mirjam Augstein, Judith Friedl and Thomas Luger, *Domino: A Descriptive Framework for Hybrid Collaboration and Coupling Styles in Partially Distributed Teams.* Proc. ACM Hum.-Comput. Interact., 2018. **2**(CSCW): p. Article 128. 10.1145/3274397.

[60] Don Norman, *The design of everyday things: Revised and expanded edition*. 2013: Basic Books.

[61] Dan R. Olsen. *Evaluating user interface systems research*. in *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 2007. Newport, Rhode Island, USA: Association for Computing Machinery. 10.1145/1294211.1294256.

[62] Jun Rekimoto. *Time-machine computing: a time-centric approach for the information environment*. in *Proceedings of the 12th annual ACM symposium on User interface software and technology*. 1999. Asheville, North Carolina, USA: Association for Computing Machinery. 10.1145/320719.322582.

[63] George G. Robertson and Stuart K. Card. *Fix and float: object movement by egocentric navigation*. in *Proceedings of the 10th annual ACM symposium on User interface software and technology (UIST '97)*. 1997. http://dx.doi.org/10.1145/263407.263535.

[64] Mark Roseman and Saul Greenberg, *GroupKit: a groupware toolkit for building real-time conferencing applications*, in *Readings in Human-computer interaction: Toward the Year 2000*, Ronald M. Baecker, Jonathan Grudin, William A. S. Buxton and Saul Greenberg, Editors. 1995, Morgan Kaufmann Publishers Inc. p. 390-397.

[65] Ibrahim Sabek, Badrish Chandramouli and Umar Farooq Minhas. *CRA: Enabling Data-Intensive Applications in Containerized Environments*. in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019. https://doi.org/10.1109/ICDE.2019.00192.

[66] Stephanie Santosa and Daniel Wigdor. *A field study of multi-device workflows in distributed workspaces*. in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 2013. Zurich, Switzerland: ACM. 10.1145/2493432.2493476.

[67] Donald A. Schön, *Designing as reflective conversation with the materials of a design situation.* Research in Engineering Design, 1992. **3**(3): p. 131-147. http://dx.doi.org/10.1007/BF01580516.

[68] A. J. Sellen and H. R. Harper, *The myth of the paperless office*. 2002, Cambridge, MA: MIT Press.

[69] Teddy Seyed, Alaa Azazi, Edwin Chan, Yuxi Wang and Frank Maurer. *SoD-Toolkit: A Toolkit for Interactively Prototyping and Developing Multi-Sensor, Multi-Device Environments*. in *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. 2015. Madeira, Portugal: Association for Computing Machinery. 10.1145/2817721.2817750.

[70] Randall B. Smith, *Experiences with the Alternate Reality Kit: An Example of the Tension between Literalism and Magic.* IEEE Comput. Graph. Appl, 1987. **7**(9): p. 42-50. 10.1109/mcg.1987.277078.

[71] Robert St. Amant and Thomas E. Horton. *Characterizing tool use in an interactive drawing environment*. in *Proceedings of the 2nd international symposium on Smart graphics*. 2002. Hawthorne, New York, USA: ACM. 10.1145/569005.569018.

[72] M. Stefik, D. G. Bobrow, S. Lanning, D. Tatar and G. Foster. *WYSIWIS revised: early experiences with multi-user interfaces*. in *Proceedings of the 1986 ACM conference on Computer-supported cooperative work*. 1986. Austin, Texas: Association for Computing Machinery. 10.1145/637069.637107.

[73] Jason Stewart, Benjamin B. Bederson and Allison Druin. *Single display groupware: a model for co-present collaboration*. in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 1999. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery. 10.1145/302979.303064.

[74] Anthony Tang, Michael Boyle and Saul Greenberg. *Display and presence disparity in Mixed Presence Groupware*. in *Proceedings of the fifth conference on Australasian user interface - Volume 28*. 2004. Dunedin, New Zealand: Australian Computer Society, Inc.

[75] Craig S. Tashman and W. Keith Edwards. *LiquidText: a flexible, multitouch environment to support active reading*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2011. Vancouver, BC, Canada: Association for Computing Machinery. 10.1145/1978942.1979430.

[76] Stephen Voida and Elizabeth D. Mynatt. *It feels better than filing: everyday work experiences in an activity-based computing system*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2009. Boston, MA, USA: Association for Computing Machinery. 10.1145/1518701.1518744.

[77] Jagoda Walny, *Constructible Interaction. Chapter 11 of "Thinking with Sketches: Leveraging Everyday Use of Visuals for Information Visualization"*. 2016, University of Calgary, Calgary, AB.

[78] M. Weiser, *The Computer for the 21st Century.* Scientific American, 1991(September): p. 94-104.

[79] Haijun Xia, Ken Hinckley, Michel Pahud, Xiao Tu and Bill Buxton. *WritLarge: Ink Unleashed by Unified Scope, Action, & Zoom*. in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017. Denver, Colorado, USA: ACM. 10.1145/3025453.3025664.

[80] Jishuo Yang and Daniel Wigdor. *Panelrama: enabling easy specification of cross-device web applications*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. 2014. ACM, New York, NY, USA. http://doi.acm.org/10.1145/2556288.2557199.