Human-oriented Infrastructures for Multi-surface Environments

Marc-Emmanuel Perrin, James R Eagan, Michel Beaudouin-Lafon

Human-oriented Infrastructures for Multi-surface Environments

Marc-Emmanuel Perrin
LTCI
Télécom ParisTech & CNRS
Université Paris-Saclay
75013 Paris, France
meperrin@telecom-paristech.fr

James R. Eagan
LTCI
Télécom ParisTech & CNRS
Université Paris-Saclay
75013 Paris, France
james.eagan@telecom-paristech.fr

Michel Beaudouin-Lafon
LRI
Univ. Paris-Sud, CNRS, Inria
Université Paris-Saclay
91405 Orsay Cedex, France
mbl@lri.fr

Presented at the Cross-Surface '16 workshop, in conjunction with ACM CHI'16. May 7, San Jose, USA.

Abstract
From offices to public spaces, dynamic multi-surface environments that can leverage the devices that users carry with them are becoming more common. However these environments are often implicit and therefore hard to discover, as are the multi-device interactions that they support. This position paper outlines the challenges that designers of multi-surface environments face to improve service discoverability, to support interactions that leverage users' devices, and to provide software tools to design and develop cross-devices applications.

Author Keywords
multi-surface interaction, wall-sized displays, infrastructures, distributed architectures

Introduction
Recent technological advances have rendered connected, personal devices much more ubiquitous. It is common for users to carry some combination of smartphones, tablets, laptops, and smart jewelry such as watches, bracelets, and rings. Similarly, physical infrastructures such as interactive wall-sized displays and tabletops, as well as systems that track the locations of users and devices, are becoming more prevalent. Advanced users, such as scientists and data analysts, increasingly incorporate such environments into their work. In the workplace, smart meeting rooms

Figure 1: Multi-surface interaction in the WILD Room [2].

are becoming more common. Even in everyday life, such interactive environments are finding their way into shopping malls and airports.

Each of these multi-surface environments, however, affords different interaction styles with different kinds of devices. Users might be able to extend the existing environment to include their own devices and data, or extend their own devices to appropriate the physical infrastructure. Discovering whether such capabilities are available and how to actually perform such operations remain unsolved problems: there are no well-established conceptual models for such distributed interfaces, and therefore users cannot integrate them into their own mental models.

Moreover, building such multi-surface applications, with interactions well-suited to users' needs, requires mastering not only the details of the application domain, but also the intricacies of low-level technologies. While it is possible to create cross-device applications with existing models, they are still too complex to build and often too brittle. To create multi-surface applications, developers need new abstractions for discoverability, management of shared data models, network communication, and adaptability to heterogeneous devices. Interaction designers, on the other hand, need more expressive models based on post-WIMP conceptual frameworks, such as instrumental interaction [1].

Multi-surface environments

The diversity of a users' devices and contexts of use results in a variety of multi-surface environments. We identify three broad categories of multi-surface environments: dedicated platforms, smart meeting rooms and public spaces.

Our work so far has focused on dedicated multi-surface environments in which interaction, processing and rendering may take place on different devices. Such distributed environments take the form of a fixed, dedicated infrastructure such as the WILD room [2] which combines wall-sized displays, motion capture systems, and data and computation clusters with heterogeneous portable devices that users may bring with them (Fig. 1). Since each platform may have specific capabilities, e.g. 3D display or multitouch wall-sized display, not available on other platforms, a design challenge is to create software that takes advantage of the specific capabilities of the platform yet can be ported to other environments.

Less extreme multi-surface environments, such as "smart" meeting rooms, may also create user-centered spaces that leverage cross-device interactions enabled by, e.g., Apple Handoff, Hamilton & Wigdor's Conductor [3], or Webstrates [4]. Unlike dedicated platforms, these environments are more standardized. Most work in this field has focused on interaction involving smartphones, tablets, tabletops and wall-sized displays. However, as wearable devices become more powerful and affordable, users will also want those devices to support new interaction capabilities in such environments.

Finally, multi-surface environments may be experienced in everyday life. In contexts such as shopping malls, airports and train stations, interactive ads and information displays are becoming more common. A user may search for a particular shop at a mall kiosk or consult an interactive subway map in a station. Since these environments are public, users may have a variety of kinds of devices. Interaction must therefore be reduced to the lowest common denominator to accomodate as many users as possible.

User perspective

We are interested primarily in multi-surface environments in which users can dynamically combine interaction between a fixed infrastructure and their own devices. For example, a user may extend the capabilities of her devices to take advantage of the local infrastructure or to enrich the local infrastructure with her own data or device capabilities. In either case, the user must first discover and pair the available devices and services before she can appropriate the new interaction space created by the combination of her own devices and the environment. Thus, the discovery and pairing processes must have low viscosity. Currently, if a user wishes to, e.g., interact with a subway map from his phone, he could easily spend more time connecting the devices together than actually interacting with the map.

Once the user's devices are connected, they create an implicit multi-surface environment that provides interaction capabilities and possibly access to data. To exploit these capabilities, the user must be aware of their existence and understand what interactions are possible and what their effects are. This requires proper feedforward and feedback to make interaction more discoverable.

For example, consider a simple task, such as editing a document on a shared display with other people in the room: how would the user discover that his device can be used to share the document with the display and notify other users that they can interact with it concurrently? What should the interaction look like to achieve that particular task? For now, the commonly used interaction models do not encompass such unified, seamless cross-device interactions. This results in ad-hoc solutions, mixing different interaction metaphors.

Technological perspective

Multi-surface interaction typically involves several devices in the interaction loop, requiring mechanisms to maintain and synchronize a consistent state across the devices as well as manage events coming from multiple devices. However, since current laptops, tablets, and smartphones were designed for standalone use, their operating systems and user interface toolkits do not provide adequate support for multi-surface environments.

The dynamicity of such environments, where devices can join and leave at any time, adds to the challenge. New software architectures and programming models are clearly needed to support these highly-distributed, dynamic and uncertain environments in order to let software developers build cross-surface applications that provide expressive and consistent interactions from the user's perspective. Conductor [3] is an example of a step in the right direction.

Our recent work on Webstrates [4] explores an alternative approach. Webstrates turn the web into shared, dynamic media: the pages served by a Webstrates server are automatically synchronized across the clients viewing them when any client makes a change. Pages can *transclude* the content of other pages [5], creating a host of possibilities to display and manipulate content. For example, an editor is a webstrate that transcludes a set of editing instruments (each is a webstrate) as well as a content page (a webstrate too). The instruments contain code that can edit the shared content. Users can configure and personalize their environment, as well as create and exchange content (including editors and instruments).

By using the web as infrastructure, any web-capable device can access Webstrates and users are immediately familiar with the basic interaction model. We have

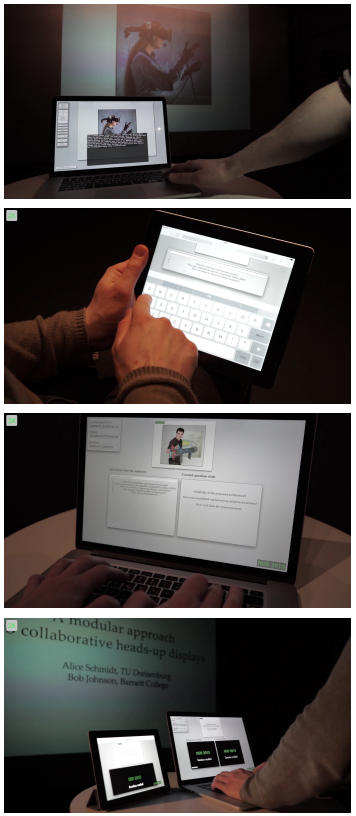

Figure 2: A slideshow controlled with Webstrates [4]. Top to bottom: presenter view, audience view, moderator view, session chair view.

created a number of scenarios that involve multiple interactive surfaces [4], such as a slide presentation with audience participation and session-chair control (Fig. 2). However, more work is needed to better support interactions that involve multiple devices. We also want to extend Webstrates to devices from the internet-of-things that do not support web protocols.

Three Challenges

We see three primary challenges faced by designers of cross-device applications:

Discoverability How can a user easily discover that pairing one of his devices with the environment might bring new interactions capabilities? Typical approaches include directing the user to a captive web portal, but this requires explicit user actions and several steps. For simple actions such as querying a display for a subway route and downloading it to their smartphone, the cost of discovery and pairing must be minimal or users will simply not use these features.

Interaction How can a user interact with multiple surfaces? Creating interactions that span several devices in a distributed environment is complex, even for simple ones, due to the dynamism of the infrastructure and the need to coordinate and synchronize multiple devices. From the users' perspective, it is critical to create a consistent conceptual model so that users can concentrate on the task at hand rather than struggle to understand what interactions are possible and how to perform them.

Software What architectures and tools should we provide to developers so they can build such applications more easily? WIMP and Post-WIMP toolkits and interface builders help developers create widgets and assemble them into functional applications with relative ease. Similar tools should be developed to design cross-device applications for multi-surface environments, as well as for managing the arrival and departure of devices in the environment.

Conclusion

This position paper has identified three categories of multi-surface environments with different levels of capabilities, and outlined the interaction and technological challenges of multi-surface interaction. We have briefly described our work on Webstrates, and highlighted three challenges for the creation of multi-surface environments.

Acknowledgements

This work was financed in part by the Digitéo project MultiVis and the French ÉQUIPEX DigiScope.

References

[1] M. Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-wimp user interfaces. In *Proc. CHI '00*, 446–453. ACM, 2000.

[2] M. Beaudouin-Lafon, S. Huot, M. Nancel, W. Mackay, E. Pietriga, R. Primet, J. Wagner, O. Chapuis, C. Pillias, J. R. Eagan, T. Gjerlufsen, and C. Klokmose. Multisurface interaction in the WILD room. *IEEE Computer*, 45(4):48 –56, April 2012.

[3] P. Hamilton and D. J. Wigdor. Conductor: Enabling and understanding cross-device interaction. In *Proc. CHI '14*, 2773–2782. ACM, 2014.

[4] C. N. Klokmose, J. R. Eagan, S. Baader, W. Mackay, and M. Beaudouin-Lafon. Webstrates: Shareable dynamic media. In *Proc. UIST '15*, 280–290. ACM, 2015.

[5] T. H. Nelson. The heart of connection: Hypermedia unified by transclusion. *CACM*, 38(8):31–33, 1995.