Towards User-defined Cross-Device Interaction

Audrey Sanctorum and Beat Signer

Web & Information Systems Engineering Lab
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium
{asanctor,bsigner}@vub.ac.be

Abstract. Over the last decade we have seen various research on distributed user interfaces (DUIs). We provide an overview of existing DUI approaches and classify the different solutions based on the granularity of the distributed UI components, location constraints as well as their support for the distribution of state. We propose an approach for user-defined cross-device interaction where users can author their customised user interfaces based on a hypermedia metamodel and the concept of active components. Furthermore, we discuss the configuration and sharing of customised distributed user interfaces by end users where the focus is on an authoring rather than programming approach.

Keywords: Cross-device interaction; DUIs; end-user development

1 Classification of Distributed User Interfaces

Over the last few decades, distributed user interfaces (DUIs) have gained a lot of attention [23]. Various terms have been introduced in order to differentiate between different DUI systems, ranging from multi-device and multi-display interaction to interactive spaces and cross-device interaction. Multi-device applications started to emerge already in the late twentieth century when, for example, Robertson et al. [26] presented a system that allowed users to interact with a TV by using a personal digital assistant (PDA) and a stylus. A limitation of this early system was that the information flow was only possible in one direction from the PDA to the TV, which prevented a user from capturing information from the TV to their PDA. Only a year later, Rekimoto [25] introduced the pick-and-drop technique which allowed users to exchange information in any direction by picking up an object on one computer screen and dropping it on another screen by using a digitiser stylus. Since then, research in cross-device interaction has gone a long way and different techniques, interaction possibilities, frameworks and applications have been developed [10]. In order to pass information across devices, Frosini et al. [12] make use of QR codes. The Deep Shot system [7] supports information sharing between a smartphone and a computer screen via the smartphone's camera. While Deep Shot uses a feature matching algorithm, in the Conductor system Hamilton and Wigdor [13] proposed another solution which enables the distribution of information across different

tablets via broadcasting. An alternative way for cross-device communication has been presented by Rädle et al. [24] with the HuddleLamp system which uses a lamp with an integrated camera to track hand movements and the position of any mobile device that has been placed on the table the lamp is standing on. While HuddleLamp covers a limited area, other systems allow for interactions across a room [31,16,5,11,18], a network environment [21,15,14,32,13,7,6] or have no space limitations and can be used anywhere [3,4,20,8,27,2,12,17].

Apart from the 'space' dimension, other criteria can be used to highlight the differences between existing DUI solutions. For example, the granularity of the distributed components is another interesting criteria. Some systems, such as the ARIS system [5] where application windows can be moved across devices, only support the distribution of applications as a whole. Other solutions offer a specific set of components that can be distributed. For example, MultiMasher [15] supports the distribution of arbitrary elements from any website while Melchior et al. [20] support the distribution of application widgets and arbitrary pixels on a screen. This also offers the possibility to transfer the state of an application across devices and to have synchronous views of the shared data. The distribution of user interfaces and user interface components is often performed via a message passing mechanisms [13,18,1,27,3,4,16,21]. Moreover, certain systems developed their own software infrastructure for the sharing of information across devices as seen with BEACH in the i-LAND [31] project.

Distributed user interfaces are often built following a client-server architecture, where the server plays a central role in keeping each user interface on the different devices up to date [7,21,14,15,32,13,6]. However, this approach limits the interaction space of the connected devices since all devices need to be connected to the server. A solution to overcome this limitation is the use of peer-to-peer networks without a need for a central server [3,4,20].

While Demeure et al. [9] proposed a reference framework to differentiate existing DUI approaches based on the four dimensions of computation, communication, coordination and configuration, in Figure 1 we provide our classification of the discussed approaches based on their constraints in terms of location and the supported granularity for distributed UI components. On the x-axis we go from local solutions on the left to solutions without any space limitation on the right. On the y-axis we differentiate between systems where the entire UI and data can be shared to approaches that support the sharing of UI elements at a finer granularity. Note that we further highlight systems that support the distribution of state by labelling them in a bold blue font.

Some of the previously described systems focus on the portability, others centre around the collaborative aspect and a third group focusses on making it easier for designers and developers to create DUI applications. However, almost none of them deals with making the frameworks or applications available to end users without programming skills. Certain systems like Weave provide "easy-to-use" scripting languages to build DUIs or to ease the distribution across different devices. WebSplitter [14] provides users with an XML file which contains the distribution of the UI elements across devices. Going a step further,

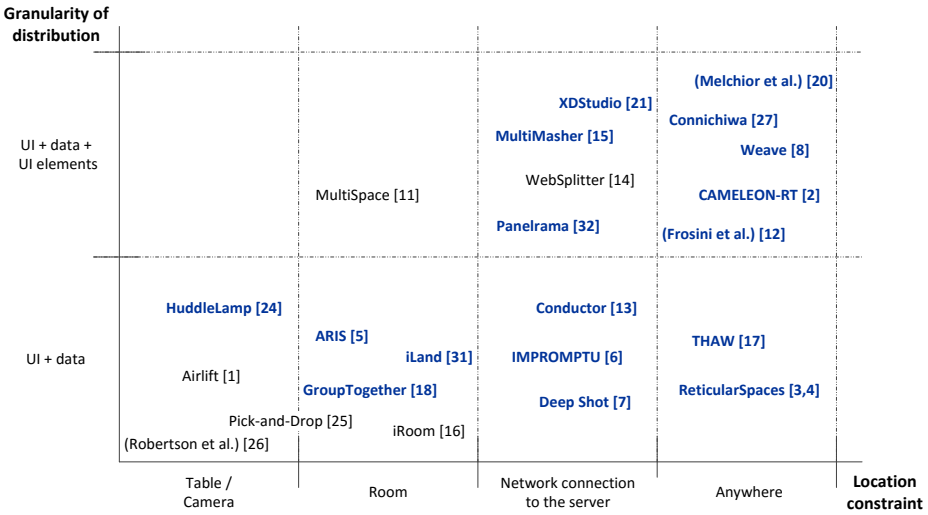

Fig. 1. Classification of DUIs based on location constraints (x-axis) and the supported granularity of distribution (y-axis)

XDStudio [21] provides a web-based authoring environment for designers who have only basic web development experience. Finally, Husmann et al. [15] presented MultiMasher, a tool for technical as well as non-technical users. MultiMasher is limited to the distribution of website components and users cannot distribute their applications and data. These systems make a step into the right direction but often still represent "closed solutions" where it is up to the developer or designer to define how exactly an end user can interact across devices.

2 Proposed Approach

In order to overcome some of the shortcomings of existing distributed user interface approaches described in the previous section, we aim at empowering end users to create, modify and reconfigure DUIs. This allows end users to combine multiple interfaces and build their own customised distributed user interfaces in order to better support their daily activities. A first question that arises in this context is how to concretely enable end users to define their customised interactions across electronic devices dealing with digital information and services. Further questions are: *"What will end users be able to modify?"*, *"How much control will end users have in terms of the granularity of the UI components to be distributed?"*, *"Will end users be limited by a specific location, space or office setting?"*, *"Will end users be able to share their configuration of customised UIs?"* and *"Can end users reuse parts of other configurations?"*.

In order to allow end users to customise existing user interfaces as well as to define their own new distributed interfaces, there is a need for end-user authoring tools that enable the specification of cross-device interactions. Note that

the authoring should not rely on a single method but offer different possibilities for unifying the different devices forming part of the interaction. We plan to develop a framework which enables the rapid prototyping of innovative DUIs by developers but also allow end users to customise existing interfaces or define their own new distributed user interfaces via a dedicated end-user authoring tool. However, we would also like to investigate new forms of authoring which go beyond the graphical definition and composition of DUI interactions based on programming-by-example. In order to develop such a rapid prototyping framework, we are currently investigating a model and the necessary abstractions for the end-user definition of cross-device interactions. Thereby, we aim for a solution where digital interface components, tangible UI elements as well as the triggered application services are treated as modular components. Any programming efforts for new cross-device user interfaces should further be minimised by turning the development into an *authoring rather than a programming activity.*

A number of authors presented models for cross-device interactions. For example, Nebeling et al. [22] introduced a model including the user, device, data, private session and session concepts that are used in their platform. Another platform model which is more centered around the concrete distribution of UIs has been introduced by Melchior [19]. In their case, a platform has the three main component categories of connection, hardware and audio/video. Existing models are often designed for a specific platform or system introduced by the authors, focus on the distribution across devices and lack concepts such as the re-usability of UI components, the different classes of users as well as the sharing of DUI configurations between users. We are currently developing a more user-centric cross-device interaction model addressing some of these issues.

A promising approach that we are currently investigating for modelling loosely coupled interaction between user interface components and various forms of actions is presented in the work of Signer and Norrie [28]. They proposed a resource-selector-link (RSL) hypermedia metamodel which enables the linking of arbitrary digital and physical entities via a resource plug-in mechanism. In our context, resources can be seen as different user interface components which can be linked together. Since often we do not want to link an entire resource but only specific parts of a resource, such as parts of a UI in order to control the granularity of the distributed user interface components as discussed in the previous section. The concept of a selector allows us to address parts of a specific resource. A detailed description of the RSL hypermedia metamodel can be found in [28]. An important RSL concept for realising our goal of DUI state transfer and the execution of third-party application logic is the concept of so-called active components [30,29]. An active component is a special type of resource representing a piece of program code that gets executed once a link to an active component is followed. This has the advantage that one can trigger some application logic by simply linking the UI, or parts of the UI represented by resources and selectors, to an active component. More importantly, an active component does not have to implement the application logic itself but can also act as a proxy for functionality offered by any third-party application. We fore-

see that the concept of active components can enable the rapid prototyping of cross-device applications by simply defining links between the necessary components. We further plan to address a number of other issues such as how to clearly separate the cross-device interactions from the underlying shared data and application state, different forms of lightweight data exchange between devices as well as the possibility for configuring interactions in an ad-hoc manner.

In addition to our model for cross-device interaction, we are currently designing an architecture and implementation of a framework which provides the necessary functionality to communicate between different user interface components and the corresponding application services. In order to facilitate replication and to enable the synchronisation of UIs and UI components on different devices, a distributed model-view-controller (dMVC) pattern, which has proven to be efficient by Bardam et al. [3,4], might be used. Another possibility is to follow the replication-based model of Biehl et al. [6] which captures the application window's pixel data and reproduces it on other devices. On the implementation level, we plan to use an event-based system and a publish/subscribe message passing mechanism as used by other systems [13,18,1,3,4,16,8,27,21,7]. Since we aim for a portable solution that can be used at any location without prior installation, such as some of the systems discussed earlier in the previous section [17,12,2,8,27,20], we consider using JavaScript to support the distribution across devices as seen in other DUI systems [24,8,27,21,7,15].

While we plan to base our cross-device interaction model on some of the concepts introduced in the RSL metamodel, we also intend to develop a framework providing the necessary functionality to communicate between different user interface components as well as a mechanism to discover and manage existing user interface components (e.g. resource/selector plug-ins and active components). The latter is encapsulated in the Developer Registry component shown in the general architecture overview of our envisioned framework in Figure 2. The Active Components sub-component stores all active components that have been implemented by developers while the Resource/Selector Plug-ins sub-component stores all the resource and selector plug-ins. In addition, it is essential to have a user interface registration and discovery service where end users can upload their newly composed interfaces to share them with other users. This functionality is encapsulated in the End User Registry component. The registry service is used to keep track of the different UI components in a given setting by means of user profiles. If a user created some new cross-device interactions between different UI components, they might be interested to make their new DUI configuration available to other users in a similar way as developers do this in the first place. For this purpose, users can post their own cross-device configurations to the Configuration Pool. This has the advantage that the interactions can be adapted and modified by different users over time which might be seen as an evolutionary development of the corresponding interactions. While in most cases users will adapt existing solutions based on their individual preferences, it can of course also be interesting to see whether some general interaction patterns evolve over time.

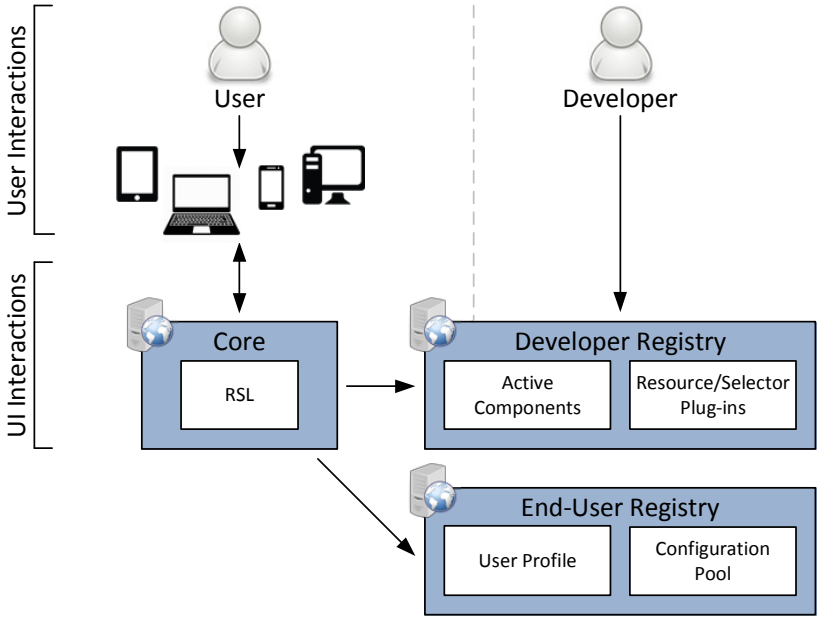

Fig. 2. Architecture for user-defined cross-device interactions

We foresee a synergy between interactions that have been predefined by a developer and are used as is, the ones that are slightly adapted by end users, as well as newly defined interactions by end users. Note that we do not plan to delegate all the interaction definitions to the end user. End users might still mainly rely on predefined interactions but will have the possibility to adapt them or add new cross-device interactions if necessary. By providing the end user the freedom to adapt the interactions, we address the issue that individual users might have slightly different requirements for certain tasks which makes it impossible to design interactions which perfectly suit everybody. Furthermore, the acceptance of specific user interfaces might be increased if end users have the chance to better integrate them with their existing work practices.

A last point that we want to address, which is related to the idea that users can share their interaction components, is how to control the granularity of the shared components. Based on the model that we plan to develop, a user could only share simple user interface components which trigger a single action via an active component. However, a user might often want to share more complex interactions involving multiple devices which can trigger different actions. We will therefore investigate how our model has to be extended in order to group multiple components together and share them as a package. Since the RSL meta-model offers the concept of structural links which can be used to group multiple entities, we plan to initially address this issue by analysing whether and how structural links could be used for defining more complex cross-device interactions by grouping multiple components.

3 Conclusion

We have proposed an approach for user-defined cross-device interaction based on a hypermedia metamodel where UI components can be linked to different application logic at any level of granularity based on the concept of active components. We have further introduced an architecture for the sharing of user-defined user interface components and discussed the authoring of these UIs.

Acknowledgements

The research of Audrey Sanctorum is funded by a PhD grant of the Research Foundation Flanders (FWO).

References

1. Bader, T., Heck, A., Beyerer, J.: Lift-and-Drop: Crossing Boundaries in a Multi-Display Environment by Airlift. In: Proc. of AVI 2010. Roma, Italy (May 2010)
2. Balme, L., Demeure, A., Barralon, N., Coutaz, J., Calvary, G.: CAMELEON-RT: A Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces. In: Proc. of EUSAI 2004. Eindhoven, The Netherlands (Nov 2004)
3. Bardram, J., Gueddana, S., Houben, S., Nielsen, S.: ReticularSpaces: Activity-Based Computing Support for Physically Distributed and Collaborative Smart Spaces. In: Proc. of CHI 2012. Austin, USA (May 2012)
4. Bardram, J., Houben, S., Nielsen, S., Gueddana, S.: The Design and Architecture of ReticularSpaces: an Activity-Based Computing Framework for Distributed and Collaborative Smartspaces. In: Proc. of EICS 2012. Copenhagen, Denmark (June 2012)
5. Biehl, J.T., Bailey, B.P.: ARIS: An Interface for Application Relocation in an Interactive Space. In: Proc. of GI 2004. London, Canada (May 2004)
6. Biehl, J.T., Baker, W.T., Bailey, B.P., Tan, D.S., Inkpen, K.M., Czerwinski, M.: IMPROMPTU: A New Interaction Framework for Supporting Collaboration in Multiple Display Environments and Its Field Evaluation for Co-located Software Development. In: Proc. of CHI 2008. Florence, Italy (Apr 2008)
7. Chang, T., Li, Y.: Deep Shot: A Framework for Migrating Tasks Across Devices Using Mobile Phone Cameras. In: Proc. of CHI 2011. Vancouver, Canada (May 2011)
8. Chi, P.P., Li, Y.: Weave: Scripting Cross-Device Wearable Interaction. In: Proc. of CHI 2015. Seoul, Republic of Korea (Apr 2015)
9. Demeure, A., Sottet, J., Calvary, G., Coutaz, J., Ganneau, V., Vanderdonckt, J.: The 4C Reference Model for Distributed User Interfaces. In: Proc. of ICAS 2008. Gosier, Guadeloupe (Mar 2008)
10. Elmqvist, N.: Distributed User Interfaces: State of the Art. In: Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem. Human-Computer Interaction Series (2011)
11. Everitt, K., Shen, C., Ryall, K., Forlines, C.: MultiSpace: Enabling Electronic Document Micro-Mobility in Table-Centric, Multi-Device Environments. In: Proc. of Tabletop 2006. Adelaide, Australia (Jan 2006)
12. Frosini, L., Manca, M., Paternò, F.: A Framework for the Development of Distributed Interactive Applications. In: Proc. of EICS 2013. London, United Kingdom (June 2013)

13. Hamilton, P., Wigdor, D.J.: Conductor: Enabling and Understanding Cross-Device Interaction. In: Proc. of CHI 2014. Toronto, Canada (Apr 2014)
14. Han, R., Perret, V., Naghshineh, M.: WebSplitter: a Unified XML Framework for Multi-Device Collaborative Web Browsing. In: Proc. of CSCW 2000. Philadelphia, USA (Dec 2000)
15. Husmann, M., Nebeling, M., Pongelli, S., Norrie, M.C.: MultiMasher: Providing Architectural Support and Visual Tools for Multi-device Mashups. In: Proc. of WISE 2014. Thessaloniki, Greece (Oct 2014)
16. Johanson, B., Fox, A., Winograd, T.: The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. IEEE Pervasive Computing 1(2) (2002)
17. Leigh, S., Schoessler, P., Heibeck, F., Maes, P., Ishii, H.: THAW: Tangible Interaction with See-Through Augmentation for Smartphones on Computer Screens. In: Proc. of TEI 2015. Stanford, California, USA (Jan 2015)
18. Marquardt, N., Hinckley, K., Greenberg, S.: Cross-Device Interaction via Micromobility and F-formations. In: Proc. of UIST 2012. Cambridge, USA (Oct 2012)
19. Melchior, J.: Distributed User Interfaces in Space and Time. In: Proc. of EICS 2011. Pisa, Italy (June 2011)
20. Melchior, J., Grolaux, D., Vanderdonckt, J., Roy, P.V.: A Toolkit for Peer-to-Peer Distributed User Interfaces: Concepts, Implementation, and Applications. In: Proc. of EICS 2009. Pittsburgh, USA (July 2009)
21. Nebeling, M., Mintsi, T., Husmann, M., Norrie, M.C.: Interactive Development of Cross-Device User Interfaces. In: Proc. of CHI 2014. Toronto, Canada (April 2014)
22. Nebeling, M., Zimmerli, C., Husmann, M., Simmen, D.E., Norrie, M.C.: Information Concepts for Cross-Device Applications. In: Proc. of DUI 2013. London, UK (June 2013)
23. Paternò, F., Santoro, C.: A Logical Framework for Multi-Device User Interfaces. In: Proc. of EICS 2012. Copenhagen, Denmark (June 2012)
24. Rädle, R., Jetter, H., Marquardt, N., Reiterer, H., Rogers, Y.: HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. In: Proc. of ITS 2014. Dresden, Germany (Nov 2014)
25. Rekimoto, J.: Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In: Proc. of UIST 1997. Banff, Canada (Oct 1997)
26. Robertson, S.P., Wharton, C., Ashworth, C., Franzke, M.: Dual Device User Interface Design: PDAs and Interactive Television. In: Proc. of CHI 1996. Vancouver, Canada (Apr 1996)
27. Schreiner, M., Rädle, R., Jetter, H., Reiterer, H.: Connichiwa: A Framework for Cross-Device Web Applications. In: Proc. of CHI 2015. Seoul, Republic of Korea (Apr 2015)
28. Signer, B., Norrie, M.C.: As We May Link: A General Metamodel for Hypermedia Systems. In: Proc. of ER 2007. Auckland, New Zealand (Nov 2007)
29. Signer, B., Norrie, M.C.: A Framework for Developing Pervasive Cross-Media Applications Based on Physical Hypermedia and Active Components. In: Proc. of ICPCA 2008. Alexandria, Egypt (Oct 2008)
30. Signer, B., Norrie, M.C.: Active Components as a Method for Coupling Data and Services - A Database-Driven Application Development Process. In: Proc. of ICOODB 2009. Zurich, Switzerland (July 2009)
31. Streitz, N.A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., Steinmetz, R.: i-LAND: An Interactive Landscape for Creativity and Innovation. In: Proc. of the CHI 1999. Pittsburgh, USA (May 1999)
32. Yang, J., Wigdor, D.: Panelrama: Enabling Easy Specification of Cross-Device Web Applications. In: Proc. of CHI 2014. Toronto, Canada (Apr 2014)