5th Workshop on Distributed User Interfaces: Distributing Interaction (DUI 2016)

5th Workshop on Distributed User Interfaces: Distributing Interactions (DUI 2016)

Organizers. María D. Lozano, José A. Gallud, Víctor M.R. Penichet, Ricardo Tesoriero, Computer Systems Department, University of Castilla-La Mancha, Albacete, Spain; Jean Vanderdonckt, Université catholique de Louvain, Belgium; Habib M. Fardoun, King Abdul Aziz University, Jeddah, Saudi Arabia; Juan Enrique Garrido, Computer Science Research Institute, University of Castilla-La Mancha, Albacete, Spain; Félix Albertos Marco, Computer Systems Department, University of Castilla-La Mancha, Albacete, Spain.

The 5th Workshop on Distributed User Interfaces was focused on distributing interactions. Current technology and ICT models generate configurations in which the same user interface can be offered through different interactions. These new technological ecosystems appear as a result of the existence of many heterogeneous devices and interaction mechanisms. Consequently, new conditions and possibilities arise, which not only affects the distribution of the user interfaces but also the distribution of the user's interactions. Thus, we shift the focus from addressing the distribution of user interfaces to the distribution of the user's interactions, which poses new challenges that need to be explored. In this context, Web engineering appears as a fundamental research field since it helps to develop device-independent Web applications with user interfaces that are capable of being distributed and accessed through different interaction modes. This fact makes Web environments especially interesting within the scope of this workshop. As in the previous workshops in this series, the main goal is to bring together people working on distributed interactions and enable them to share their knowledge in aspects related to new interaction paradigms such as movement-based interaction, speech recognition, gestures, touch and tangible interaction, etc., and the way we can manage them in a distributed setting.

The workshop started with Session 1, which was a somewhat mad session in which each participant introduced himself/herself. This session continued with two research presentations:

- Michael Krug and Martin Gaedke: "AttributeLinking: Exploiting Attributes for Inter-Component Communication." The authors propose exploiting attributes of client-side Web components to provide inter-component communication by external configuration. With the integration of a multi-device supporting Messaging Service, components can even be linked across multiple connected devices. This enables the development of distributed user interfaces.
- Juan Enrique Garrido Navarro, Victor M. R. Penichet, and Maria-Dolores Lozano: "Improving Context-Awareness in Healthcare Through Distributed Interactions." This paper describes a significant step forward in the concept of context-awareness with a comprehensive solution: Ubi4Health. The solution enhances context-awareness by adapting the user experience with the appropriate device, interface, and interaction mechanism on the basis of the given context.

Session 2 took place with six presentations:

- Amira Bouabid, Sophie Lepreux, and Christophe Kolski: "Distributed Tabletops: Study Involving Two RFID Tabletops with Generic Tangible Objects." This paper describes a study on an innovative system designed to support remote collaborative games running on

tabletops with tangible interaction. In addition, the authors model a set of collaborative styles that are possible between the tabletops users. The goal is to obtain objects that provide remote collaboration among users of interactive tabletops for tangible interaction.

- Félix Albertos Marco, Víctor M.R. Penichet, and Jose A. Gallud: "Distributing Interaction in Responsive Cross-Device Applications." In this work the authors introduce the foundations of a new approach called responsive cross-device applications (RCDA). RCDA applies the idea of responsive Web applications distributing user interactions across the new cross-device ecosystem, taking into account the interactive capacities of devices and users.
- Audrey Sanctorum and Beat Signer: "Towards User-Defined Cross-Device Interaction." The authors provide an overview of existing DUI approaches and classify the different solutions. In addition, they propose an approach for user-defined cross-device interaction where users can author their customized user interfaces based on a hypermedia metamodel and the concept of active components.
- Antonio Jesús Fernández-García, Luis Iribarne, Antonio Corral, Javier Criado, and James Z. Wang: "Optimally Storing the User Interaction in Mashup Interfaces Within a Relational Database." Storing the data generated from the interaction performed over the user interface can be challenging. To achieve this goal, in this paper a relational database for storing this interaction information generated on distributed user interfaces is proposed.
- Félix Albertos Marco,Víctor M.R. Penichet, andJose A. Gallud: "VirtualSpatially Aware Shared Displays." In this work, the authors present a technique for distributing content and devices in shared workspaces using cross-device displays. This technique, referred to as the virtual spatially aware technique, allows the creation of virtual shared displays and the coordination of cross-device interactions. By using this technique, they propose a method for arranging content and devices on virtual displays.
- Sergio Firmenich, Gabriela Bosetti, Gustavo Rossi, and Marco Winckler: "Flexible Distribution of Existing Web Interfaces: An Architecture Involving Developers and End-Users." This paper describes an architecture that allows end-users to collect UI objects into a distributed UI Component-oriented PIM, accessible from different users' devices. Once in the PIM, different DUI-based behaviors (that maybe triggered by the user) are added to the collected UI components as PIM object plug-ins.

The workshop finished with an interesting Session3, in which the participants collaborated by working together. The objective was to discuss the main ideas and results from the previous sessions, future research lines, and possible collaborations. The organization of the sessions involved all the participants. In particular, during Sessions 1 and 2, the participants listed concepts to be considered in the last session on post-it notes. These concepts were stuck on a board and categorized in Session 3. This activity allowed participants to discuss definitions, links, related and future concepts, etc. The results were an interesting exchange of ideas. Finally, this collaborative work involved the possibility of continuing to collaborate as an initial community related to distributed user interfaces and the topics included in the workshop.

July 2016

María D. Lozano
José A. Gallud

Víctor M.R. Penichet
Ricardo Tesoriero
Jean Vanderdonckt
Habib M. Fardoun
Juan Enrique Garrido
Félix Albertos Marco

Program Committee

María D. Lozano University of Castilla-La Mancha, Spain
José A. Gallud University of Castilla-La Mancha, Spain
Víctor M.R. Penichet University of Castilla-La Mancha, Spain
Ricardo Tesoriero, University of Castilla-La Mancha, Spain
Jean Vanderdonckt, Université catholique de Louvain, Belgium
Habib M. Fardoun, King Abdul Aziz University, Saudi Arabia
Juan Enrique Garrido, University of Castilla-La Mancha, Spain
Félix Albertos Marco, University of Castilla-La Mancha, Spain

AttributeLinking: Exploiting Attributes for Inter-component Communication

Michael Krug(✉) and Martin Gaedke

Technische Universität Chemnitz, Chemnitz, Germany
{michael.krug,martin.gaedke}@informatik.tu-chemnitz.de

Abstract. In this paper, we propose exploiting attributes of client-side web components to provide inter-component communication by external configuration. With the standardization of WebComponents, the Web is finally getting a uniform way to define and use client-side components. We determined that DOM elements already provide a standard configuration interface: attributes. Using the WebComponents technologies for state-of-the-art user-interface components, attributes can also act as output interfaces. By providing an Attribute-Link component, new web applications can be composed directly in the markup without knowledge of JavaScript. With the integration of a multi-device supporting MessagingService, components can be even linked across multiple connected devices. This enables the development of distributed user interfaces.

Keywords: Web components · Web application development · Composition · Distributed user interfaces · Reusable components

1 Introduction

Component-based application development is based on the composition of multiple components that are reused and combined to form new applications. Applying this to the Web creates new challenges, like the definition of independent components and loosely coupling. A lot of approaches were developed in the last years, but a standardized way of defining and using components for client-side Web application development was missing. Recently, the W3C is working on creating a new set of standards for components for the Web called *WebComponents*[1]. With those standards developers are now able to define new types of DOM elements with custom functionality. Similar to UI mashups, where user-interface components were mostly referred to as widgets, inter-component communication, which is required for application development by composition, is a central problem. This is unfortunately not covered by the W3C specifications. We analyzed different DOM elements and determined that most of them provide an accessible (input) interface, which is mainly used for configuration: *attributes*. Consequently, this interface is also available for WebComponents.

[1] WebComponents - W3C Wiki https://www.w3.org/wiki/WebComponents/.

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 157–161, 2016.
DOI: 10.1007/978-3-319-46963-8_13

```
<geo-coder address="Chemnitz"// input interface
           lat="50.83"       // output interface
           lng="12.92">       // output interface
</geo-coder>
```

Listing 1. Geo-Coder component with multiple attributes

With an accordingly configuration, attributes of WebComponents can also be used as output interfaces. To explain this idea, an example in Listing 1 is stated, where a *Geo-Coder* component is shown that converts an address into geographic coordinates (latitude, longitude), e.g., to show a place on a map. Thus, the attribute *address* is used as an input interface and *lat* and *lng* are the resulting output variables. If we now assume that there is also a *Map* component with latitude and longitude as input variables mapped to the according attributes, a composition of those elements can be achieved by connecting the output variables of the *Geo-Coder* to the input ones of the *Map*.

Therefore, we want to encourage developers to use attributes as communication interfaces when creating new WebComponents. By enabling external configuration of inter-component data exchange the components do not need to include any communication functionality by themselves. Thereby, components stay reusable and do not need knowledge of other components.

2 Related Work

Polymer, a WebComponent framework from Google, supports data bindings within markup through placeholders in attributes and by placing target and source within a binding element. There are also several JavaScript frameworks that support data binding, e.g., AngularJS, Knockout or Rivets.js. They need an attached data model to specifically bind data to the element's content or attributes. Furthermore, some publish/subscribe or event-based communication approaches for components or widgets, like presented in [1,2] and [3], exist. They have the disadvantage of configuring communication aspects within the components. There are only limited options to configure which components shall exchange data. Data transformation and message distribution for multi-device applications is not supported.

3 The AttributeLinking Approach

Based on the assumption that WebComponents are DOM-Elements that provide their in- and output interfaces through attributes, we enable inter-component communication by linking attributes through external configuration to facilitate web application development by composition. How to connect those interfaces?

3.1 The Attribute-Link Component

We propose to connect the attribute interfaces of components using a stand-alone *Attribute-Link* component, which is configured using three attributes: *source*, *target* and an optional *transformation* (see Listing 2). The *Attribute-Link* component is also implemented as a WebComponent. Thus, it can be natively used in web browsers supporting those standards and can be deployed directly in the markup without knowledge of JavaScript. To address an element we propose to use the established *CSS selector* syntax[2]. This syntax is applied to both the source as well as the target selector. Since CSS selectors can return multiple elements, we also support multiple elements as target and source. Enabling a cardinality of both ends from 1 to n. The attribute is addressed by its name separated by an `@` sign. Example: `geo-coder#id1@address`. The *Attribute-Link* can be included multiple times in the web application to define a complex inter-component communication setup. To watch for attribute changes without affecting the responsiveness of the application we use native DOM mutation observers[3] that set up microtasks and provide a callback for registered changes of the specific element's attribute. If a transformation function was specified, it will be applied and the resulting value is propagated to the defined attribute of the according target element.

```
<attribute-link source="selector@attribute"
                target="selector@attribute">
                transformation="JavaScript-Code"/*Optional*/>
</attribute-link>
```

Listing 2. Syntax of the proposed Attribute-Link component

3.2 Distributing Attribute Changes

In our approach, we also focus on the development distributed user interfaces. Thus, we propose to additionally address target components on connected devices. This is achieved using a *Messaging-Service* component and a *Messaging-Server* we provide. Our presented *Attribute-Link* component seamlessly integrates with the *Messaging-Service* by placing it in the DOM as a child element like display in Listing 3. It notifies the *Messaging-Service* of changed attributes that shall be distributed by dispatching a custom event. This event contains the target element selector, the attribute name and the new value to be set. Using the *WebSocket protocol*[4] the *Messaging-Server* is contacted to distribute this message to other connected devices (browsers running the application with the same endpoint configured that share the same context) (cf Fig. 1). The other devices also need to have instantiated an accordingly configured *Messaging-Service* component that will then propagate the received messages to the target elements in their DOM. We also provide a configuration option to only target remote components.

[2] Selectors Level 3 https://www.w3.org/TR/selectors/.
[3] DOM Standard https://dom.spec.whatwg.org/#mutation-observers.
[4] The WebSocket Protocol http://tools.ietf.org/html/rfc6455.

```
<messaging-service endpoint="protocol://address:port"
                   session="Session-Identifier">
        <attribute-link [...]></attribute-link>
</messaging-service>
```

Listing 3. Syntax of the Messaging-Service in combination with an Attribute-Link

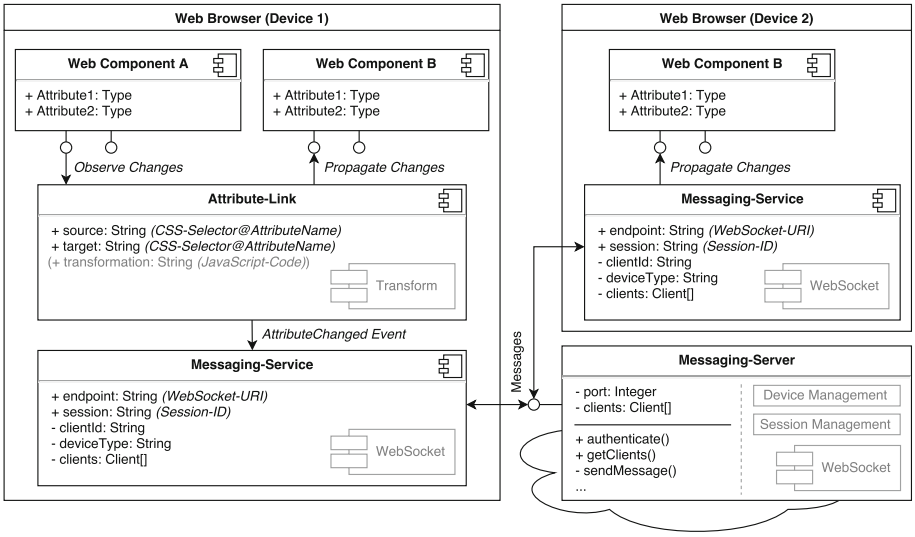

Fig. 1. Components of the AttributeLinking approach

4 Conclusion

In this paper, we proposed the external linking of attributes of DOM-based components to enable the composition of new web applications. Consequently, we want to start a discussion of the usage of attributes as in- and output interfaces for components in the Web. To demonstrate our idea, we presented an Attribute-Link component that is capable of observing and propagating attribute changes from and to multiple DOM elements. With an optional Messaging-Service those changes can also be distributed to connected devices. The concept of external configuration eliminates the need of modifying the components to enable communication. Further research will address how to enhance the support for distributed interfaces by providing more configuration options to e.g., select components on specific devices.

Online Demonstration: http://myvsr.eu/demo/dui/

References

1. Chudnovskyy, O., Müller, S., Gaedke, M.: Extending web standards-based widgets towards inter-widget communication. In: Grossniklaus, M., Wimmer, M. (eds.) ICWE Workshops 2012. LNCS, vol. 7703, pp. 93–96. Springer, Heidelberg (2012)
2. Krug, M., Gaedke, M.: SmartComposition: bringing component-based software engineering to the web. In: Proceedings of the 17th International Conference on Information Integration and Web-Based Applications and Services, pp. 474–477. ACM (2015)
3. Sire, S., Paquier, M., Vagner, A., Bogaerts, J.: A messaging API for inter-widgets communication. In: Proceedings of the 18th International Conference on World Wide Web, pp. 1115–1116. ACM (2009)

Improving Context-Awareness in Healthcare Through Distributed Interactions

Juan E. Garrido[1,2(✉)], Víctor M.R. Penichet[1,2], and María D. Lozano[1,2]

[1] Computer Science Research Institute, University of Castilla-La Mancha, Albacete, Spain
juanenrique.garrido@uclm.es
[2] Computer Systems Department, University of Castilla-La Mancha, Albacete, Spain
{victor.penichet,maria.lozano}@uclm.es

Abstract. Context-aware systems represent an appropriate tool for healthcare environments. The capability to offer necessary information and functionality on the basis of a user's context makes it possible to create better working environments for medical staff. Current technology supports greater customization, thus enabling the improvement of user interaction. This paper describes a significant step forward in the concept of context-awareness with a comprehensive solution: Ubi4Health. The solution enhances context-awareness by adapting the user experience in accordance with the device, interface and interaction mechanism used in a given context.

Keywords: Healthcare · Ubiquity · Context-awareness · DUI · HCI

1 Introduction

Medical staff have to manage complex situations [1] which involve: working with professionals from different fields; reaching agreement; performing multiple tasks; dealing with frequent contingencies that requires a constant adjustment of their actions because of the dynamic nature of their work [2]; and keeping in mind several pending tasks [3].

These conditions can be improved through context-aware [4] and ubiquitous [5] systems. Users can work without thinking about the system which is responsible for adapting itself to existing conditions at any time and under any circumstance. Generally speaking, context-aware systems adapt themselves to offer functionality and information that users need based on the context. However, current technological possibilities provide a chance of taking a step forward. The context offers valuable information, such as the physical capabilities of the user, which can be used to determine the appropriate interaction mechanism. In this sense, the distribution of multiple interaction methods across the working environment represents a good opportunity for context-awareness improvement.

The distribution of interaction mechanisms is directly related to multi-device environments. The possibility to interact with multiple devices offers more ways of interaction. But it also helps to continue improving context-aware capabilities. Users can interact with specific devices according to their needs, such as mobility conditions in

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 162–166, 2016.
DOI: 10.1007/978-3-319-46963-8_14

healthcare [6]. At the same time, a multi-device environment adds a new possibility of improvement for context-aware systems: the use of the distribution of user interfaces. As users are provided with a wide range of devices, the system can adapt the interface to the hardware and also to the users' context.

Taking all the above into account, the authors have developed a ubiquitous and context-aware comprehensive solution for deployment in healthcare centres, named Ubi4Health [7]. The solution supports many functions, from task management, an alert notification mechanism, and falls and fainting detection to a rehabilitation assistant. This paper describes how Ubi4Health generates a novel context-aware environment for users (employees, patients and residents). The main contribution of this work is in the use of distributed interaction mechanisms, a multi-device environment and appropriate user interfaces to adapt the system to a user's context. These elements provide users with an experience that is adapted to their needs at the information and functionality level, and also to the way they interact with the system. The related works studied [6, 8–10] offer solutions to specific fields, as Ubi4Health does within its modules. However, none of them considers context-awareness as more than adapting data and functionality capabilities.

The following section summarizes the way Ubi4Health improves healthcare conditions thanks to its support for a novel development of context-awareness. Finally, we present our conclusions.

2 Ubi4Health: Improving and Making Progress with Traditional Context-Aware Systems

Ubi4Health is a novel, context-aware and ubiquitous comprehensive solution to enhance working conditions in healthcare environments. The solution has been developed modularly, and each module is related to a specific domain (Fig. 1): (1) *tasks module* to manage tasks (alerts are tasks with a high priority); (2) *rehabilitation module* to improve specific rehabilitation processes; and (3) *KFF (Kinect for Falls and Fainting) system* to automatically detect anomalous situations in the environment.

The context-awareness in Ubi4Health represents a significant contribution. The system offers functionality and information based on a user's context. However, the objective has been to go a step further. To that end, three features and capabilities are used together: *distributed multimodality*, *multi-device* and the *distribution of user interfaces*.

Firstly, Ubi4Health has a complete set of interaction mechanisms distributed over the healthcare environment (Fig. 1). On the one hand, the solution allows users to interact with traditional mechanisms (keyboard, mouse and touchscreen). These offer the possibility to work with tasks and rehabilitation modules regardless of the circumstances. However, there are situations in which it is necessary to make use of other interaction methods to enhance the user experience. In the tasks module there is a role named operator. This role has to control the completion of tasks, adequate staff distribution and the assignment of alerts. As a result, the amount of information to be managed is huge. Therefore, Ubi4Health incorporates the use of users' gaze to interact with the operator.

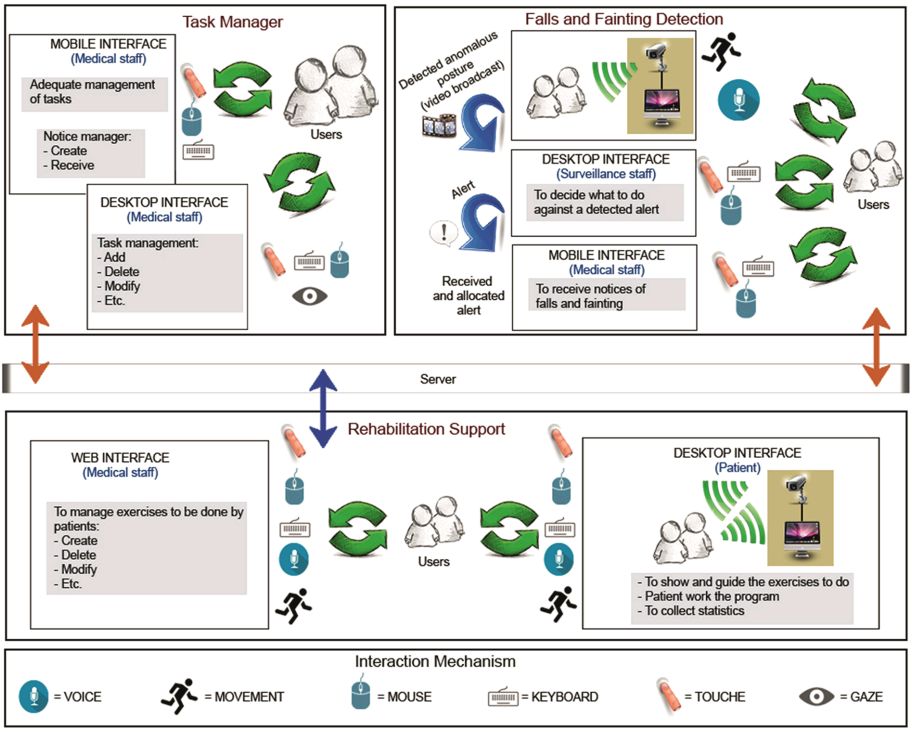

Fig. 1. Interfaces and interaction mechanisms of Ubi4Health

In this way, the system is able to know where the user is looking in order to identify loss of information on unattended displays. The system adapts its behaviour and aspect to a user's context as it generates notifications to assist the perception of information [11].

Movement-based interaction is an essential element. In the rehabilitation module, this interaction allows patients to complete their therapies at home. The system analyses a patient's movements and automatically guides and monitors. Also, this mechanism offers medical staff the possibility to define exercises to be performed by patients. In particular, medical staff must generate therapies, defined as posture repetitions. The postures can be created through a 3D designer, and also by the movement of the medical staff. Furthermore, in the KFF System, the solution uses movement interaction to continuously analyse a patient's/resident's postures. The objective is to detect falls and fainting, which is an important part of patient care.

The distributed multimodality offered by Ubi4Health is completed with the user's voice. The solution provides the possibility of control via voice for some interfaces. This feature helps in those situations in which the use of hands or the whole body is a complicated task. An example arises during the rehabilitation process. If a patient is performing an exercise, voice is a good way of controlling the system.

This multimodality is provided together with an important infrastructure which generates the second feature related to context-awareness improvement: the use of a wide device set. Ubi4Health offers pcs, laptops, tablets, PDAs, mobile phones, as well

as Kinect cameras, web cameras and touchscreens. The result is the capacity to adapt the use of the solution to the appropriate device based on the user's needs. For example, the task module considers two user types with the role of employee who present different mobility needs. Firstly, there are users who are constantly in movement, such as doctors. They have to work with convenient devices to be carried with them and which can be used anywhere and at any time. Ubi4Health offers them mobile phones, tablets or PDAs. The other type of user has no movement needs, and is named operator. They are offered a pc, but with one peculiarity, namely a multi-display setting. The main reason for this comes from the fact of working with a large amount of information from different domains.

The last feature of Ubi4Health that is related to context-awareness is the distribution of interfaces (Fig. 1). Modularity has implied the use of different interfaces for different components of the solution. This fact adds another level of customization to different needs and objectives. In addition, one of the interfaces is a distributed user interface, which is the one related to tasks and oriented towards operators. As these users have to manage large quantities of information, the related interface is deployed over three monitors to support three domains for analysis.

3 Conclusions and Future Work

Healthcare environments are an important research area in which to make efforts to improve working conditions. Medical staff have to be able to address complex situations in which mobility, multidisciplinary and dynamic features appear as constraints. Under these conditions, an appropriate way to help users is to offer a system for their daily duties with the capacity to adapt itself according to their needs. For this purpose, this paper has presented Ubi4Health, a comprehensive solution for healthcare environments, focusing on the contribution related to context-awareness. The solution supports users with appropriate and necessary information and functionality on the basis of their current context. In addition, the use of multiple devices, distributed interaction mechanisms and user interfaces has allowed us to go another step forward. A novel scenario appears as Ubi4Health is able to reach an improved level of user experience. Based on the user's context, the solution provides employees and patients with the appropriate device, user interface and way to interact with it at the time.

There are interesting lines of future work connected with the presented context-aware scenario. Currently, the authors are considering the possibility of extending the devices mentioned. For instance, wearable devices offer an interesting way in which Ubi4Health can increase its mobile capabilities and ways of interaction.

Acknowledgements. This work has been partially funded by project with reference 0215ITA017 from the Castilla-La Mancha Regional Government: Junta de Comunidades de Castilla-La Mancha, Consejería de Empleo y Economía.

References

1. Lymberis, A., Smart, A.: Wearables for remote health monitoring, from prevention to rehabilitation: current R&D, future challenges. In: Proceedings of the 4th International IEEE EMBS Conference on Information Technology Applications in Biomedicine, pp. 272–275 (2003)
2. Bardram, J.E., Bossen, C.: Moving to get aHead: local mobility and collaborative work. In: Proceedings of CSCW, pp. 355–374 (2003)
3. Bardram, J.E., Christensen, H.B.: Pervasive computing support for hospitals: an overview of the activity-based computing project. Pervasive Comput. **6**(1), 44–51 (2007)
4. Bricon-Souf, N., Newman, C.R.: Context-awareness in health care: a review. Int. J. Med. Inform. **76**, 2–12 (2007)
5. Weiser, M.: The computer for the twenty-first century. Sci. Am. **265**(3), 94–104 (1991)
6. Favela, J., Martínez, A.I., Rodríguez, M.D., González, V.M.: Ambient computing research for healthcare challenges, opportunities and experiences. Computación y Sistemas **12**(1), 109–127 (2008)
7. Garrido, J.E., Penichet, V.M., Lozano, M.D.: Ubi4Health: ubiquitous system to improve the management of healthcare activities. In: Proceedings of Pervasive 2012 Conference, ACM Digital Library, Newcastle, UK (2012)
8. Gonzales, A.L., Pollak, J.P., Retelny, D., Baumer, E.P., Gay, G.: A mobile application for improving health attitudes: being social matters. In: CHI 2011, Vancouver, BC, Canada, 7–12 May 2011
9. Bardram, J.E., Hansen, T.R., Mogensen, M., Soegaard, M.: Experiences from real-world deployment of context-aware technologies in a hospital environment. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 369–386. Springer, Heidelberg (2006)
10. Kjeldskov, J., Skov, M.B.: Supporting work activities in healthcare by mobile electronic patient records. In: Masoodian, M., Jones, S., Rogers, B. (eds.) APCHI 2004. LNCS, vol. 3101, pp. 191–200. Springer, Heidelberg (2004)
11. Garrido, J.E., Penichet, V.M., Lozano, M.D., Quigley, A., Kristensson, P.O.: AwToolkit: attention-aware user interface widgets. In: Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces 2014. ACM, Como (2014)

Distributed Tabletops: Study Involving Two RFID Tabletops with Generic Tangible Objects

Amira Bouabid[1,2], Sophie Lepreux[1], and Christophe Kolski[1(✉)]

[1] LAMIH-UMR CNRS 8201, University of Valenciennes, Valenciennes, France
{amira.bouabid,sophie.lepreux,
christophe.kolski}@univ-valenciennes.fr
[2] SETIT, University of Sfax, Sfax, Tunisia
amira.bouabid@gmail.com

Abstract. This paper describes a study on an innovative system designed to support remote collaborative. This system is a game running on tabletops with tangible interaction. Twelve test groups, each composed of three participants, tested a distributed application for learning and recognition of colors. We propose a set of generic tangible objects. They model a set of collaborative styles which are possible between users. Our goal is to obtain objects that provide remote collaboration among users of such interactive tabletops. This study is supported by observations, trace analysis and questionnaires. In this study, we analyze if the use of generic objects is easy and understandable by users in case of remote collaboration. The user satisfaction when using the distributed tangible tabletops is also studied.

Keywords: Tangible interaction · Tabletop · Distributed UI · Remote collaboration · Tangible object · Tangiget · RFID

1 Introduction

Collaborative work proves important within a team, or more generally within a group of users. In fact, team people often needs to exchange ideas [9], to work on common tasks [10] or to be informed about the progress of a task [11]. Much work has been done on the subject concerning the flow of information between users and platforms [2]. We seek in this work to facilitate the collaboration between different people working together. Our objective is to propose a system that provides remote collaboration throw interactive tabletops with tangible interaction [1]. Collaboration in this system is based on a set of generic tangible objects, called tangigets, initially defined by Lepreux *et al.* in [8] and Caelen and Perrot in [3]. These tangigets will materialize a set of collaboration styles listed by Isenberg et al. in [4].

In the paper we describe the design of the system and the interaction, as well as principles for remote collaboration on tangible tabletops. The study is then presented, and the first results are explained. Finally the paper concludes and proposes research perspectives.

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 167–173, 2016.
DOI: 10.1007/978-3-319-46963-8_15

2 System Design: DUI on Two RFID Tabletops

As Distributed User Interface (DUI), we use two *TangiSense* interactive tabletops allowing tangible interaction [6] (designed by the RFidees Company; see www.rfidees.fr). These tabletops use the RFID technology to recognize objects placed on the surface, as shown in Fig. 1.

Fig. 1. Two different user interfaces shown on the interactive tabletops with some of used (business and generic) tangible objects: on the left, user Interface displayed on the *child* tabletop during the correction of exercise; on the right, interface displayed on the *adult* tabletop during the supervision of the exercise realization

The application used in our study is a distributed version of an application on RFID tabletop allowing the learning and recognition of colors; the initial version was presented in [7]. A set of business objects is used; they present various colorless pictures. For this application the final users are very young children learning colors; they have to arrange the business objects in color areas according to dominant colors. These objects are divided into 4 categories; each category contains 8 objects of varying difficulty levels.

A set of generic tangible objects, called tangigets, is used to ensure remote collaboration between interconnected tabletops. These objects give users the ability to interact remotely using the features provided by the interactive tabletop.

3 Interaction Design: Use of Generic Tangible Objects

Remote collaboration between users on tabletops is carried out by the use of a set of generic objects (tangigets). The aim of these generic objects is to support an inter-user dialogue using only features offered by interactive tabletops:

- *Identification* tangiget: Used to identify users who are currently using the collaborative application and want to enter in collaboration with other users.
- *Task assignment* tangiget: Used to organize tasks between different users of the collaborative application.
- *Starting synchronization* tangiget: Used to synchronize the start of the activity distributed on connected tabletops.
- *Display Mode* tangiget: Used to change the display of the main interface according to the user needs.
- *Request help* tangiget: Used to ask for help or ask a question about a step or a detail of the collaborative activity.
- *Provide help* tangiget: Used to offer help about a step or detail following a request.
- *End task* tangiget: Used to mark the end of a task and/or to switch to another task.
- *Criticism* tangiget: Used to work on all of the activity (not on one task).

Figure 2 shows two examples of Tangigets used in this application: they correspond respectively to *Criticism* tangiget (*Magician* object) and *Provide help* tangiget (*Erase* object).

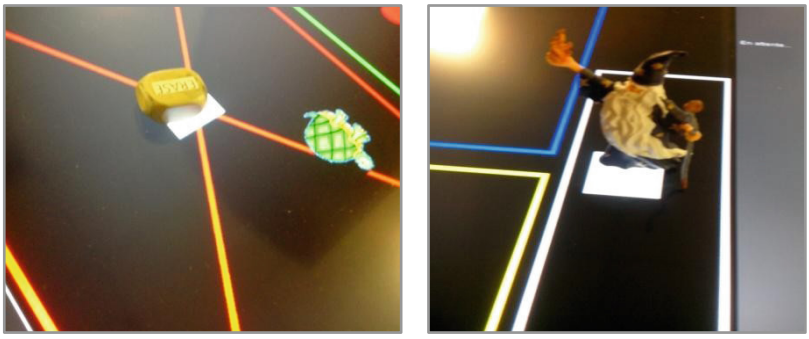

Fig. 2. Some of used tangigets in the application: on the left, *Provide help* tangiget (represented by *Erase* object); on the right, *Criticism* tangiget (represented by *Magician* object)

4 Remote Collaboration on Tangible Tabletops

Isenberg et al. [4] listed eight collaboration styles covering any type of collaboration (co-located or remote). The proposed tangigets have to comply with the standards of collaboration. So we instantiated each style by an action on the system by using: (1) one tangiget or (2) a coupling between two tangigets.

Table 1 shows the correspondence between the action of each tangiget and one of the collaboration styles.

Table 1. Tangigets contributing to collaboration styles.

Collaboration styles proposed in [4]	Representative action on the application by the use of a generic object
Active **disc**ussion	*Request help*
View **E**ngaged	*Task assignment*
Sharing of the **S**ame **V**iew	*Starting synchronization*
Sharing of the **S**ame **I**nformation but using **D**ifferent **V**iews	*Display mode*
	Identification
Working on the **S**ame **S**pecific **P**roblem	*Task assignment coupled with identification*
Working on the **S**ame **G**eneral **P**roblem	*Provide help*
Working on **D**ifferent **P**roblems	*Criticism*
Disengaged	*End task*

5 Case Study

In the application, we propose to use a set of tangigets useful for remote collaboration. Table 2 shows those objects and their main functionalities in the application.

Table 2. Instances of tangigets used for the application.

Type of tangiget	Instantiation	Definition of its role
Identification	Identification	Used to identify the person present remotely and ready to play
Starting Synchronization	Start	Start the game on the two connected tabletops
Task assignment	Category	Assign a category of object to the person identified
Request help	Collaboration area	Request remote assistance by placing in this area object(s) requiring help
Provide help	Erase	Offer help by crossing color areas
Display mode	Focus	Display the results of the exercise (textual representation)
End task	End exercise	Indicate the end of the exercise for each user
Criticism	Magician	Correct remotely the exercise

A presentation of the system and its functioning as well as the functioning of each tangiget was given for each group of three participants. It was followed by a familiarization phase with the interactive tabletops during which the participants were encouraged to try the application and all items offered and to freely ask questions. After this phase of familiarization with the system, tests were started with different scenarios provided. We designed three conditions that varied aspects of the use of the *Help request* and the correction of the exercise. In these different conditions, instructions were provided to users of the *child* tabletop. We aimed to get a definite number of mistakes

and requests for help. Figure 3 provides an illustration of the participants of a test group set in relation to their different roles.

Fig. 3. A participant testing the application (*adult* tabletop)

To simulate a remote collaboration, participants were in the same room; the two tabletops were separated by a folding screen to prevent users from each tabletop from seeing the contents of the other tabletop. Moreover, to prevent that the *Parent* participant from being disturbed by possible natural discussions coming from the *child* tabletop, he/she had a headset with music.

After the study, each participant had to complete a questionnaire. The questionnaire firstly concerned information on the usability aspects and participant satisfaction with the system. Secondly, he or she had to fill more specific information on generic objects, ease of use and their significance in relation to their role set by the designer.

Finally, the evaluator used a trace file in which all the games played by the group were recorded in order to understand how they had addressed the problem and to get their reactions on the technologies and principles used. The analysis of the trace file is based mainly on a set of reactions of the user following the action by the other user of the remote tabletop. According to the response received, we classified them into three categories: expected answer, acceptable answer and incoherent answer.

As an example, we illustrate our analyses with two tangigets (1) *Erase* object used by the user of the *adult* tabletop and (2) *Identification* object used by the user of the *child* tabletop. The results of the questionnaires for *Identification* and *Erase* objects are summarized in Fig. 4; the score of each answer is shown for each question. We can find from the figure that participants give high marks on the global situation. They think these tangigets are easy to use and have a meaningful form. Also they have understood the goal of collaboration. To study if the use of tangigets by the participants confirms or not their subjective answers, we analyzed all the trace files in which we recorded the events concerning all games played. We extracted all uses of tangigets; after that we classified them as expected use, acceptable use or incoherent use. Analyses relative to *Identification* and *erase* objects are summarized in Fig. 5.

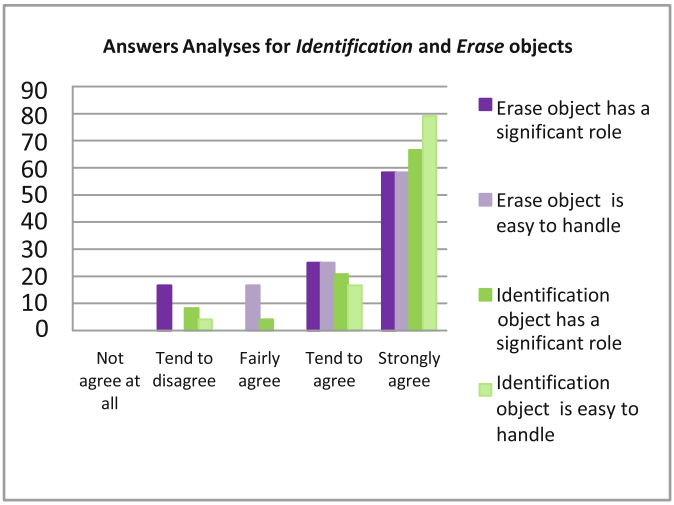

Fig. 4. Subjective answers of participants who used *Identification* and *Erase* tangigets

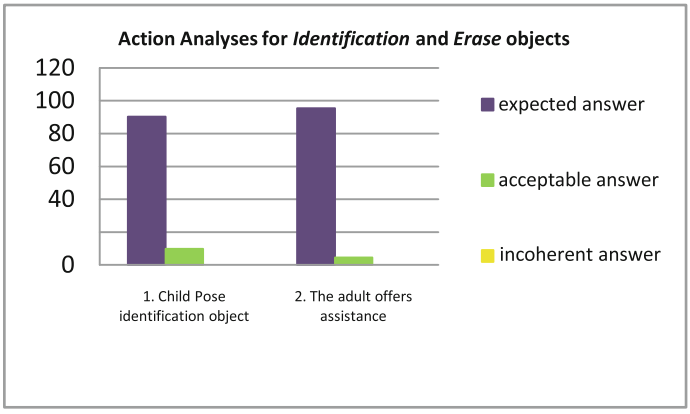

Fig. 5. Objective answers of participants who used *Identification* and *Erase* tangigets

6 Conclusion

In this paper, we introduce an innovative distributed application for the learning and recognition of colors. Generic tangible objects, called tangigets, are used to facilitate collaboration and exchange between distant participants about the exercise. This system takes advantage of large-scale tangible tabletops, (1) providing a simple user interface easy to manipulate; (2) enabling several users to collaborate remotely in each step of the exercise; (3) providing the possibility to cover a set of collaboration styles (in the sense of Isenberg et al. [4]) by the use of tangigets. A study was conducted with twelve

groups of three users. The results are promising and show the interest of the distributed approach. In future works we aim to test such tangigets with other more complex applications to verify if collaboration remains easy/possible.

Acknowledgements. The authors thank warmly Sebastien Kubicki for his work on the first centralized version of the application [5]. They thank also the 36 participants and Steve Gabet for his contribution to the distributed version.

References

1. Bouabid, A., Lepreux, S., Kolski, C., Havrez, C.: Context-sensitive and collaborative application for distributed user interfaces on tabletops. In: Proceedings of the 2014 Workshop on Distributed User Interfaces and Multimodal Interaction, pp. 23–26. ACM, New York (2014)
2. Brave, S., Ishii, H., Dahley, A.: Tangible interfaces for remote collaboration and communication. In: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, pp. 169–178. ACM (1998)
3. Caelen, J., Perrot, C.: Bibliothèque d'objets. Deliverable, IMAGIT ANR project. LIG, Grenoble (2011)
4. Isenberg, P., Fisher, D., Morris, M. r., Inkpen, K., Czerwinski, M., MacEachren, A., Miksch, S.: An exploratory study of co-located collaborative visual analytics around a tabletop display. Presented at the 2010 Proceedings of IEEE Symposium on Visual Analytics Science and Technology (VAST 2010), 1 January 2010
5. Kubicki, S., Lepreux, S., Kolski, C.: Evaluation of an interactive table with tangible objects: application with children in a classroom. In: Proceedings of the 2nd Workshop on Child Computer Interaction "UI Technologies and Educational Pedagogy", Conjunction with CHI 2011 Conference (2011)
6. Kubicki, S., Lepreux, S., Kolski, C.: RFID-driven situation awareness on TangiSense, a table interacting with tangible objects. Pers. Ubiquit. Comput. **16**(8), 1079–1094 (2011)
7. Kubicki, S., Wolff, M., Lepreux, S., Kolski, C.: RFID interactive tabletop application with tangible objects: exploratory study to observe young children' behaviors. Pers. Ubiquit. Comput. **19**(8), 1259–1274 (2015)
8. Lepreux, S., Kubicki, S., Kolski, C., Caelen, J.: From centralized interactive tabletops to distributed surfaces: the tangiget concept. Intl. J. Hum.-Comput. Interact. **28**(11), 709–721 (2012)
9. Qin, Y., Liu, J., Wu, C., Shi, Y.: uEmergency: a collaborative system for emergency management on very large tabletop. In: Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces, pp. 399–402. ACM, New York (2012)
10. Robinson, P., Tuddenham, P.: Distributed tabletops: supporting remote and mixed- presence tabletop collaboration. In: Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems, 2007. TABLETOP 2007, pp. 19–26 (2007)
11. Yamashita, N., Kaji, K., Kuzuoka, H., Hirata, K.: Improving visibility of remote gestures in distributed tabletop collaboration. In: Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, pp. 95–104. ACM, New York (2011)

Distributing Interaction in Responsive Cross-Device Applications

Felix Albertos-Marco(✉), Victor M.R. Penichet, and Jose A. Gallud

Computer System Department,
University of Castilla-La Mancha, Albacete, Spain
{felix.albertos,victor.penichet,jose.gallud}@uclm.es

Abstract. With the emergence of the Internet of Things there are many applications in which the interaction is distributed over multiple devices. Developing applications in these scenarios is challenging because there is not enough knowledge and even less consensus on how to distribute interaction. But following the ongoing trends such as responsive web design, why not enable applications to seamlessly adapt interaction to take advantage of available devices at any moment? In this work we introduce the foundations of a new approach called Responsive Cross-Device Applications (RCDA). RCDA applies the idea of responsive Web applications distributing user interactions over the new cross-device ecosystem, taking into account the interactive capacities of devices and users.

Keywords: Responsive interaction · Cross-device applications · Distributed interaction · Interactive capacities

1 Introduction

Distributed Interaction refers to the interaction process in which the user/s use/s dynamically separated input/output devices (e.g. a document viewer [5,6], a snake-like game [5], exploring a distributed map [2,6] or video streaming applications [2,5,6]). With the emergence of the Internet of Things, there is the prospect of a future in which devices can always be connected. Developing applications for these scenarios is challenging because there is not enough knowledge and even less consensus on how to distribute interaction.

In these scenarios, Santosa [4] identified specialization as the main characteristic to improve multi-device interaction and the coordination of devices. Hamilton et al. [2] found that little attention has been paid to interactions which enable the chaining of device functionality, and cross-device experience was found to be painfully absent in today's technologies. Yang presented Panelrama [6], a web-based framework for the construction of applications using distributed user interfaces. It allows users to interact with a single application from multiple devices that dynamically change user interface allocation to best-fit devices. Jokela [3] found that users want to use their devices to seamlessly work with each other, but in practice they continuously encountered problems in multi-device use. Schreiner presented Connichiwa [5], a framework for the development

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 174–178, 2016.
DOI: 10.1007/978-3-319-46963-8_16

of cross-device web applications focused on the integration of existent devices, independent from the network infrastructure, and offering versatility over application scenarios and API usability. Chi et al. presented Weave [1], a framework for developers to create cross-device wearable interaction by scripting.

Most of the developments are ad-hoc solutions based on particular case studies and/or developers' intuition. They do not fully take into account the whole interactive system: users, devices and tasks. There is a lack of consensus about how to distribute interaction according to the capabilities of the interactive system and their relation to the main goal of every interactive system: task accomplishment. However, following the ongoing trends such as responsive web design, why not enable applications to seamlessly adapt interaction to take advantage of available devices at any moment? This work provides the foundations for a new approach aimed at dealing with the design and development of Responsive Cross-Device Applications (RCDA) in current cross-device applications. The idea applies responsive Web applications to distributing user interactions over the new cross-device ecosystem.

The paper is structured as follows. First, the two pillars of RCDA are presented. Then the foundations of our proposal are described: user and device characterization, adaptation responsibility, runtime adaptations and user patterns. Finally, the conclusions and future work are presented.

2 Responsive Cross-Device Applications

The new approach of RCDA has two pillars: Responsive Web applications, and Cross-Device applications. The following paragraphs explain these two foundational ideas. Nowadays, and even more in the future, it seems to be clear that applications must support a user's tasks over multiple devices. This tendency is somewhat similar to the Web and how Web application development and use evolved to be what they are now: Responsive Web applications. They adapt to some characteristics of devices such as size, screen resolution, orientation or input modalities. But today's environments are far more complicated. In a similar way, the coordination of tasks in cross-device applications is complex [1]. Current approaches do not allow for simultaneous use of devices or cross-device interaction. Suitable applications are needed to deal with the wide range of existent cross-device scenarios [5].

However, responsive design in cross-device applications should also consider the fact that applications distribute functionality among different devices. Therefore, new features regarding such a distribution should be applied in the design. Thus, following the aforementioned two pillars, we propose a new approach for the design of cross-device applications: Responsive Cross-Device Applications (RCDA). RCDA adapts the interaction according to the ecosystem, which is composed of devices, users and tasks. The main goal of RCDA is to support users' tasks in cross-device environments, adapting interaction to facilitate user task completion.

From the analysis of previous works we set the foundations for the design and development of RCDA. The foundations are the following:

1. User and device characterization.
2. Adaptation responsibility.
3. Runtime adaptations.
4. User patterns.

These foundations are described in the following sections.

2.1 Users and Devices Characterization

User and device characterization has to take into account not only properties such as size and display resolution, but also the interactive capabilities of users and devices, the allocation of specific roles to each device and/or actor, the coordination of devices for simultaneous use and what the device is best suited for. In previous approaches, the description and categorization of interactive capabilities has been limited to physical properties such as size and display resolution. Only a few works [6] have used other device capabilities. But these descriptions are limited to specific parts of the user interface and some characteristics of a device's capabilities. In our approach the description of the interactive system is based on the interactive capabilities of the actors involved in the interaction process: users and devices. Therefore, in RCDA the actors have to be described according to their input and output interactive capabilities. These capabilities are mapped to support task completion on the interactive system. This mapping takes into account the description of the current task in terms of interactive capabilities. As a result, the interaction adapts to the task performed by the interactive system and the interactive capabilities of users and devices. The process of mapping interaction capabilities to tasks is still an open topic. But previous approaches can be used, such as defining roles [4] or defining fitness values [6], among others.

2.2 Adaptations Responsibility

The adaptation of user interaction is the responsibility of the system. It is the applications and not the users that have to adapt and move interaction across multiple devices to help users to fulfil their tasks. They are in charge of mapping the interactive capabilities of both users and devices. And, under specific circumstances, users will be able to decide from among multiple interaction paths. Depending on how users are able to decide from among multiple interaction paths, we have set three schemas:

1. Fixed. Users do not decide how the interaction is distributed.
2. Restricted. Users will be able to decide how the interaction is distributed in a limited way. They do not have full control over how to distribute interaction.
3. Free. Users are fully capable of distributing interaction over the interactive system.

Each one of these schemas has its strengths and weaknesses. While the first schema (fixed) is the most restrictive, not allowing users to decide how interaction is distributed, it is also the most simple from the point of view of the users. They only have to use the system. They are not aware of when, why and how adaptations are made. On the other hand, the free schema is the most complicated for users. They have to decide when and how the distribution has to be made. Therefore, it is important to support the schema that best fits the interactive system requirements.

2.3 Runtime Adaptations

RCDA has to dynamically categorize devices and users to change interaction to best-fit users' tasks completion. It is not enough to describe an application's scenario statically. Multi-device scenarios tend to change their configuration, adding or removing devices/users and consequently changing interaction capabilities. RCDA has to be aware of these circumstances. Therefore, they have to be up to date with the interactive input and output capabilities of the actors involved, adapting interaction in runtime. Therefore, RCDA has to support flexible mapping between these capabilities and the task the user/s is/are performing.

2.4 User Patterns

RCDA has to identify and support users' patterns for task execution in cross-device scenarios. Depending on how the task is transferred between devices, the information moves between them or their physical or logical arrangement in patterns that are classified as serial or parallel. In the serial or sequential pattern the task is transferred from one device to another [3,4]. One specialization of this pattern is the producer-consumer pattern, in which the content is produced on one device for it then to be moved to, and used by, another device [4]. In the parallel pattern many devices are coordinated for simultaneous use [4]. In the performer-informer pattern, one device has the resources to support the creation of content on other device [4]. In the performer-informer-helper pattern users use support devices (helpers) to perform traversal tasks [4]. In the controller-viewer/analyzer pattern users wish to combine device functionalities to perform different aspects of a single primary task [4]. Users also perform resource lending (borrowing resources from other devices), related parallel use (all devices are involved in a single task) and unrelated parallel use (devices are involved in different tasks) [3]. Devices can also be physically organized by mapping their physical position within the environment, following the spatial mapping pattern [2]. But there may be no spatial mapping when organizing devices [2]. The devices = windows pattern is used when devices are used as windows in the WIMP paradigm [2]. In the device = role pattern each devices is mapped with a single role for the current task [2]. In the device = storage pattern each device shows, for example, a specific document for the task [2].

3 Conclusions and Future Work

In this paper we have introduced the foundations of a new approach to handling the distribution of interactions in cross-device applications, and we have called it Responsive Cross-Device Applications (RCDA). The main characteristics of this approach are: it goes beyond screen size to design adaptations; applications are in charge of adapting interaction in cross-device environments, not users; and it makes this happen dynamically. Also, depending on how the task is transferred between devices, how the information moves between them or their physical or logical arrangement, the patterns presented have to be taken into account for the development of RCDA. As future work, and due to the complexity of RCDA, we will continue with the development of tools for the description and simulation of such complex scenarios.

Acknowledgments. This work has been partially supported by the fellowship 2014/10340 from the University of Castilla-La Mancha.

References

1. Chi, P., Li, Y.: Weave: scripting cross-device wearable interaction. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI 2015), pp. 3923–3932. ACM, New York (2015). doi:10.1145/2702123.2702451
2. Hamilton, P., Wigdor, D.J.: Conductor: enabling and understanding cross-device interaction. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2014), pp. 2773–2782. ACM, New York (2014). doi:10.1145/2556288.2557170
3. Jokela, T., Ojala, J., Olsson, T.: A diary study on combining multiple information devices in everyday activities and tasks. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, pp. 3903–3912. ACM, New York (2015). ISBN 978-1-4503-3145-6
4. Santosa, S., Wigdor, D.: A field study of multi-device workflows in distributed workspaces. In: Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013), pp. 63–72. ACM, New York (2013). doi:10.1145/2493432.2493476
5. Schreiner, M., Radle, M., Jetter, H., Reiterer, H.: Connichiwa: a framework for cross-device web applications. In: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA 2015), pp. 2163–2168. ACM, New York (2015). doi:10.1145/2702613.2732909
6. Yang, J., Wigdor, D.: Panelrama: enabling easy specification of cross-device web applications. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2014), pp. 2783–2792. ACM, New York (2014). doi:10.1145/2556288.2557199

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

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 179–187, 2016.
DOI: 10.1007/978-3-319-46963-8_17

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 [5,11,16,18,31], a network environment [6,7,13–15,21,32] or have no space limitations and can be used anywhere [2–4,8,12,17,20,27].

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 [1,3,4,13,16,18,21,27]. 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 [6,7,13–15,21,32]. 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 Fig. 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,

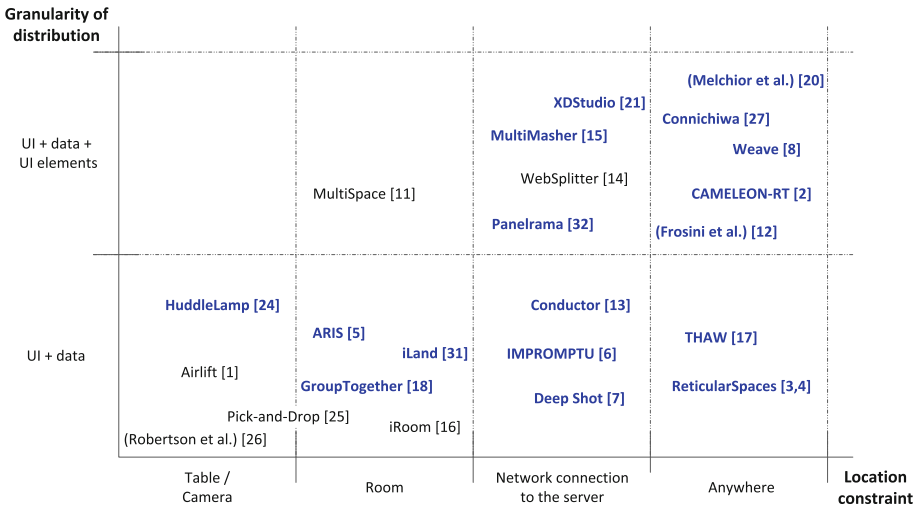

Fig. 1. Classification of DUIs based on location constraints (x-axis) and the supported granularity of distribution (y-axis) (Color figure online)

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 [29,30]. 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 foresee 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 [1,3,4,7,8,13,16,18,21,27]. 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 [2,8,12,17,20,27], we consider using JavaScript to support the distribution across devices as seen in other DUI systems [7,8,15,21,24,27].

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 Fig. 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.

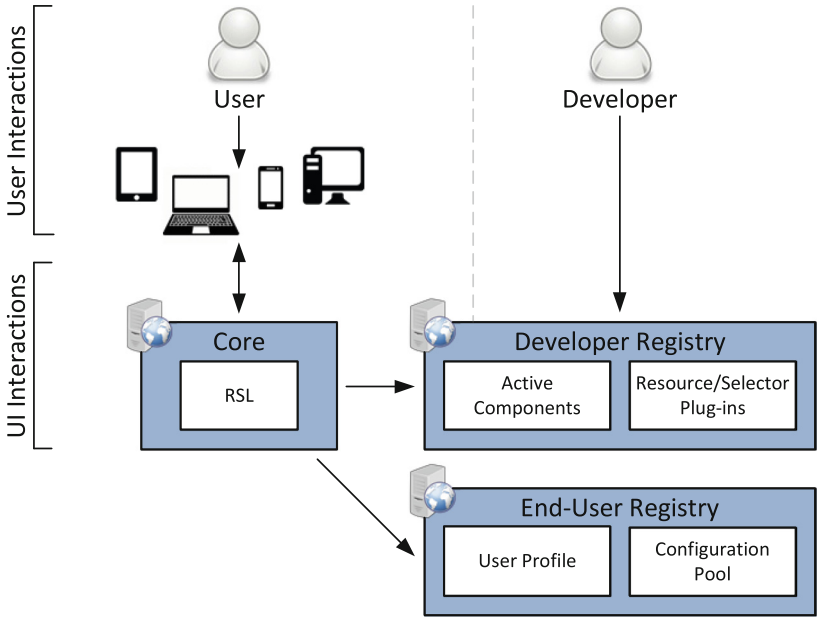

Fig. 2. Architecture for user-defined cross-device interactions

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: Proceedings of AVI 2010, Roma, Italy, May 2010
2. Balme, L., Demeure, A., Barralon, N., Calvary, G.: CAMELEON-RT: a software architecture reference model for distributed, migratable, and plastic user interfaces. In: Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L. (eds.) EUSAI 2004. LNCS, vol. 3295, pp. 291–302. Springer, Heidelberg (2004)
3. Bardram, J., Gueddana, S., Houben, S., Nielsen, S.: ReticularSpaces: activity-based computing support for physically distributed and collaborative smart spaces. In: Proceedings 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: Proceedings 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: Proceedings 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: Proceedings of CHI 2008, Florence, Italy, April 2008
7. Chang, T., Li, Y.: Deep shot: a framework for migrating tasks across devices using mobile phone cameras. In: Proceedings of CHI 2011, Vancouver, Canada, May 2011
8. Chi, P.P., Li, Y.: Weave: scripting cross-device wearable interaction. In: Proceedings of CHI 2015, Seoul, Republic of Korea, April 2015
9. Demeure, A., Sottet, J., Calvary, G., Coutaz, J., Ganneau, V., Vanderdonckt, J.: The 4C reference model for distributed user interfaces. In: Proceedings of ICAS 2008, Gosier, Guadeloupe, March 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: Proceedings of Tabletop 2006, Adelaide, Australia, January 2006
12. Frosini, L., Manca, M., Paternò, F.: A framework for the development of distributed interactive applications. In: Proceedings of EICS 2013, London, UK, June 2013

13. Hamilton, P., Wigdor, D.J.: Conductor: enabling and understanding cross-device interaction. In: Proceedings of CHI 2014, Toronto, Canada, April 2014
14. Han, R., Perret, V., Naghshineh, M.: WebSplitter: a unified XML framework for multi-device collaborative web browsing. In: Proceedings of CSCW 2000, Philadelphia, USA, December 2000
15. Husmann, M., Nebeling, M., Pongelli, S., Norrie, M.C.: MultiMasher: providing architectural support and visual tools for multi-device mashups. In: Benatallah, B., Bestavros, A., Manolopoulos, Y., Vakali, A., Zhang, Y. (eds.) WISE 2014, Part II. LNCS, vol. 8787, pp. 199–214. Springer, Heidelberg (2014)
16. Johanson, B., Fox, A., Winograd, T.: The interactive workspaces project: experiences with ubiquitous computing rooms. IEEE Pervasive Comput. **1**(2), 67–74 (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: Proceedings of TEI 2015, Stanford, CA, USA, January 2015
18. Marquardt, N., Hinckley, K., Greenberg, S.: Cross-device interaction via micromobility and F-formations. In: Proceedings of UIST 2012, Cambridge, USA, October 2012
19. Melchior, J.: Distributed user interfaces in space and time. In: Proceedings 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: Proceedings 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: Proceedings 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: Proceedings of DUI 2013, London, UK, June 2013
23. Paternò, F., Santoro, C.: A logical framework for multi-device user interfaces. In: Proceedings 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: Proceedings of ITS 2014, Dresden, Germany, November 2014
25. Rekimoto, J.: Pick-and-drop: a direct manipulation technique for multiple computer environments. In: Proceedings of UIST 1997, Banff, Canada, October 1997
26. Robertson, S.P., Wharton, C., Ashworth, C., Franzke, M.: Dual device user interface design: PDAs and interactive television. In: Proceedings of CHI 1996, Vancouver, Canada, April 1996
27. Schreiner, M., Rädle, R., Jetter, H., Reiterer, H.: Connichiwa: a framework for cross-device web applications. In: Proceedings of CHI 2015, Seoul, Republic of Korea, April 2015
28. Signer, B., Norrie, M.C.: As we may link: a general metamodel for hypermedia systems. In: Proceedings of ER 2007, Auckland, New Zealand, November 2007
29. Signer, B., Norrie, M.C.: A framework for developing pervasive cross-media applications based on physical hypermedia and active components. In: Proceedings of ICPCA 2008, Alexandria, Egypt, October 2008

30. Signer, B., Norrie, M.C.: Active components as a method for coupling data and services – a database-driven application development process. In: Norrie, M.C., Grossniklaus, M. (eds.) Object Databases. LNCS, vol. 5936, pp. 59–76. Springer, Heidelberg (2010)
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: Proceedings of the CHI 1999, Pittsburgh, USA, May 1999
32. Yang, J., Wigdor, D.: Panelrama: enabling easy specification of cross-device web applications. In: Proceedings of CHI 2014, Toronto, Canada, April 2014

Optimally Storing the User Interaction in Mashup Interfaces Within a Relational Database

Antonio Jesús Fernández-García[1(✉)], Luis Iribarne[1], Antonio Corral[1], Javier Criado[1], and James Z. Wang[2]

[1] Applied Computing Group, University of Almeria, Almería, Spain
{ajfernandez,luis.iribarne,acorral,javi.criado}@ual.es
[2] The Pennsylvania State University, State College, USA
jwang@ist.psu.edu

Abstract. Cross-device applications that have user interfaces managed in multiple forms of interaction are prevalent. In particular, component-based (or *mashup*) applications are growing in popularity due to their easiness to build customized user interfaces with pieces of information from different sources. Since the user interaction on mashup interfaces can generate a large quantity of data, which can be useful to improving the interaction and usefulness of the application, it may involve the creation of cloud infrastructures to manage the dynamic distributed user interfaces within this context. Storing the generated data from the interaction performed over the user interface can be challenging. To achieve these goals, in this paper, a relational database for storing this interaction information generated on distributed user interfaces is proposed. Thus, user interaction over heterogeneous interfaces and devices described in detail, will be easily accessible for further analysis using machine learning and data mining techniques to offer a better user experience.

Keywords: Mashup · User interaction · Multiforms of interaction · Cross-device applications · Relational database

1 Introduction

Today users consume information through heterogeneous devices such as computers, laptops, tablets or smartphones. Moreover, each device has a different way to interact with; some of them support classical forms of interaction by means of keyboard and mouse, others interact through touch interfaces, gestural interfaces or voice recognition (*Natural User Interaction*, NUI). Other interaction technologies are emerging, *e.g.*, virtual reality or wearables & IoT solutions.

Frequently, a same application needs to be available for multiple devices via different user interfaces (UIs). Users expect applications to be accessible via any device regardless of the screen size, the type of interaction, or the technologies involved in it [1]. It becomes even more complicated when it concerns to the user

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 188–195, 2016.
DOI: 10.1007/978-3-319-46963-8_18

configuration of the interface. Usually, UIs manage some configuration options and remember the behavior and the interactions performed by users and more features progressively.

Due to the increasing amount of services and APIs available, it is becoming a standard practice to use content from many sources in a Web application through a single UI. These UIs, commonly referred to as component-based UIs or *mashup*, allow users to easily customize their UI by employing different pieces of information or data creating their own tailored UI. Mashup interfaces [2] are typically used. Due to their granularity (coarse-grained) they facilitate the adaptation of their internal structure. A cloud infrastructure for the management of mashup UI can be a natural approach. In previous works a series of Web services, located in the platform-independent layer of a cloud infrastructure, have been created to support component-based architectures of mashup UI [3,4]. These services include features such as managing users, component or sessions; and the administration of modules, controllers and databases that underlies below. This infrastructure provides a solid base to create dynamic UIs [5].

This paper focuses on the interaction of users over *mashup* interfaces. There is an extraordinary potential in analyzing the interaction performed in mashup interfaces to improve the user experience by adapting the interface at run-time to the users' requirements and even stepping to the users' needs. Using *machine learning* and *data mining* techniques over the interaction data acquired from users makes it possible to discover behavioral patterns and create prediction models. For that, it is necessary not only to acquire the data but also to know exactly the morphology of component-based UIs and to create an *optimized* relational database that can storage all the data for further analysis. Currently, there is no database schema proposal to store user interaction. The problem is not straightforward because there are many mashup UI and each one of them has a different purpose and their users have different domain knowledge, skills and expectations. Also, Web technologies are diverse and for that reason the data acquisition process that has to be implemented to store the interaction in the database should be independent of the technology used to develop the mashup UI, as well as not intrusive and totally transparent for users.

The rest of the paper is organized as follows. Section 2 describes the morphology of a basic mashup graphic user interface (GUI). Section 3 proposes a relational database to store the interaction produced over this type of interfaces. Section 4 shows a query to the information gathered in the database deployed in a real mashup. We conclude and provide future directions in Sect. 5.

2 Essential Mashup GUI Morphology

Mashup User Interfaces (mashup UI) are Web applications that integrate one or more components from one or more sources to create a unique UI that combines different components that might or might not have relationship among them. This section explains in detail how mashup UIs are composed, with focus on mashup GUIs. A standard interface that covers all the common aspects of

mashups has been considered. There are many more features available in specific interfaces but all of them have some core elements and operations, which have been taken into consideration in this morphology definition.

There are many examples of commercial component-based interfaces. Nowadays, mashup interfaces (or component-based interfaces) are widespread in commercial software, particularly in Web applications [6]. Geckoboard is a KPI dashboard surface where users can visualize and work with their most important business data in real-time focusing on sales, marketing or operations among other features [7]. Cyfe allows users to build their own dashboard adding pieces of information through social media, analytics, sales, finance or project management components among others [8]. ENIA (Environmental Information Agent) is a mashup component-based GUI for environmental management used by the Andalusian Environmental Information Network (REDIAM) [9], a public organization that belongs to the Andalusian Regional Government (Spain) [10].

Figure 1 conceptually presents a component-based interface where the elements that form it are shown. Obviously, there may be many more elements and they could be positioned differently. All mashup UIs studied share some core elements and that is what is represented in Fig. 1.

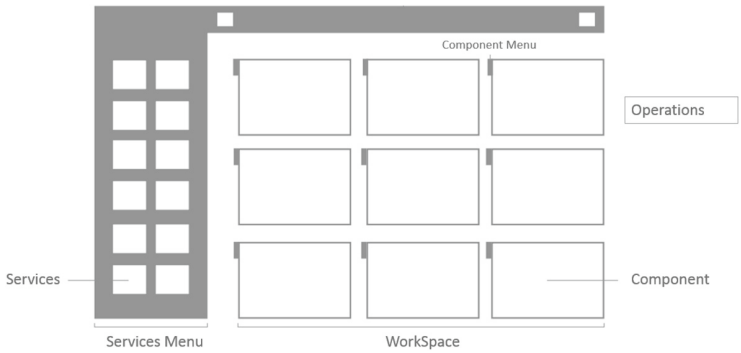

Fig. 1. Conceptual design of a component-based Web application.

Services. The capacities that the mashup application offers. They are available to users in order to operate with them. An instance of a *Service* is a *Component*.

Services menu. In this menu a list of all the *Services* available in the mashup application can be found. Users can navigate through this menu to find the services they may need. Usually, this menu is categorized and grouped by types of services and it has some search tools to locate them directly.

Component. When a user adds a Service to the *workspace* it is automatically transformed into a *Component*. A *Component* is a *Service* that is being used by a user at a certain moment in time. When a *Component* is instantiated, a set of attributes like width, height or position are assigned to it.

Workspace. The *Workspace* is the work area where users have all the *Components* (*Services* instantiated) they are working with.

Operations. All possible actions that are able to be applied over the *Components* such as resize, move or delete, among others.

Therefore, a mashup GUI ($\mathcal{M}$) is defined in the following manner: $\mathcal{M} = \{\mathcal{S}, \overline{\mathcal{S}}, \mathcal{C}, \mathcal{W}, \mathcal{O}\}$. Thus, M is comprised of a set of services $\mathcal{S}$, a service menu $\overline{\mathcal{S}}$, a set of components $\mathcal{C}$, a workspace $\mathcal{W}$ and a set of operations $\mathcal{O}$. The set of *services* $\mathcal{S}$ is defined as $\mathcal{S}$=$\{S_1, S_2, .., S_N\}$ where N is the number of *services* registered in the information system. The set of *components* $\mathcal{C}$ is defined as $\mathcal{C}$=$\{C_1, C_2, .., C_L\}$ where L is the number of *components* instantiated in the workspace $\mathcal{W}$. A concrete component C_i has some properties so it could be defined as $C_i = \{PosX, PosY, Width, Height\}$. Finally, the set of operations is defined as $\mathcal{O}$=$\{Add, Delete, Move, Resize\}$ and they are described below:

Add. Consists in adding a service to the workspace from the services menu, so it is instantiated into a component. When instantiating, some properties such as position in the x-axis, position in the y-axis, width and height are assigned to the component.

Delete. Consists in removing a component from the workspace. That happens mostly because it is of no use and users decide to dispense of it.

Resize. Consists in changing the size assigned to a component. It modifies the 'width' (w) and 'height' (h) properties. Sometimes the Resize operation can be decomposed in several operations such as:

$$Resize(x) = \begin{cases} x = ResizeBigger \mid (w_i * h_i) < (w_{i+1} * h_{i+1}) \\ x = ResizeSmaller \mid (w_i * h_i) > (w_{i+1} * h_{i+1}) \\ x = ResizeShape \mid (w_i * h_i) = (w_{i+1} * h_{i+1}) \\ \quad \wedge ((w_i \neq w_{i+1}) \vee (h_i \neq h_{i+1})) \end{cases},$$

where ResizeBigger operation is considered when the area covered after the operation is bigger; ResizeSmaller, when the area covered after the operation is smaller; and ResizeShape when the area covered is the same but the values of the properties are differents.

Move. Consists in changing the position of a component. It modifies the $PosX$ and $PosY$ properties. When $(PosX_i \neq PosX_{i+1}) \wedge (PosY_i = PosY_{i+1})$ the component has been displaced horizontally, when $(PosX_i = PosX_{i+1}) \wedge (PosY_i \neq PosY_{i+1})$ the component has been displaced vertically and finally, when $(PosX_i \neq PosX_{i+1}) \wedge (PosY_i \neq PosY_{i+1})$ the component has been displaced both horizontally and vertically.

3 Database Design for Storing Interactions

When an interaction occurs in the mashup application, a data acquisition process will start and save all the information regarding the operation by the interaction. Together with the operation it is convenient to save the information about the user that generates the interaction as well as the component that is affected. It is

also advisable to save the state that remains in the workspace after the operation. Although storing all the workspace might seem rather costly, it would make it possible to rebuild all the users' behavior step by step throughout the interfaces in case further analysis, not considered at design time, is required. That is why this option is viewed in this proposal.

In order to store data of the interactions that have been performed by users in the mashup UI, it is necessary to define a relational database model that would be able to store all the relevant information of the interaction. This relational database should be as complete as possible to have a good understanding of the interaction itself and the circumstances that surround that interaction. Figure 2 shows a proposal relational database schema that represents the interaction performed as well as the situation of the UI after it.

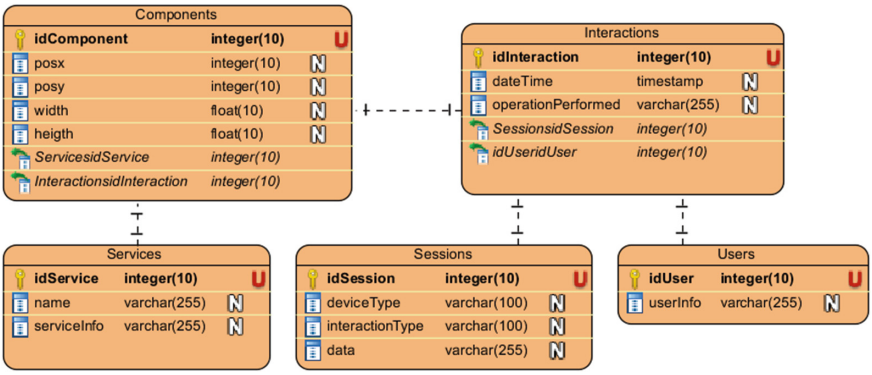

Fig. 2. Database schema to storage interaction in a standard mashup UI

Each row of the *Interactions* table corresponds to an interaction taken by the user and the field *operationPerformed* saves the kind of operation performed. The *Interactions* table is related to the *Sessions* table, thus all the operations performed in the same session are grouped. The *Sessions* table has two important fields *deviceType* and *interactionType*. The first one storages the kind of device used in the session where the operations are performed; it can be a Tablet, a Laptop, a SmartPhone, Home Automation Systems or Smart Watches, among others. The second one storages the type of interaction used when performing the operation: mouse, keyboard, gesture, voice or presence, among other. Note that there can be many sessions working currently because one application can be use at the same time through more than one UI from the same or different devices or systems.

The *Users* table, which is related to the *Interactions* one, has information of all users registered in the application. Usually, there are a lot of users registered in most of applications, therefore, it is important to distinguish the operation performed by each one of them. Some information systems allow guest users to access to the system and, for those kind of users, there is normally a specific row in the *Users* table. Note that it would be necessary to obtain extra information

about users that do not come with the interaction, hence it would be useful a request to Web services provided by the service, if any. In case the mashup UI has no users registered the *Users* and *Sessions* tables can be thrown out.

The *Components* table includes all components that populate the workspace after and interaction has been performed. With the information gathered in this table, it is possible to rebuild the workspace exactly as it was when the interaction was performed. The *posx, posy, width* and *height* attributes are enough to set each component in the workspace. Finally, the *Services* table, related to the *Components* table, has information about all the services that are registered in the Information Systems. As in the *users* table, it could be necessary, but not mandatory, to access to external Web services to obtain more relevant information about services that do not come with the interaction.

This database schema could seem rather costly due to the extensive resources consumption that may involve to store the workspace with all the components contained within. However, it is optimized in the sense that the database is expressive enough not only to generate datasets with rich data for further analysis, but for recreate user interaction step by step in case that more information about any aspect of the interaction could be detected as needed to infer a knowledge not contemplated when designing the database schema.

4 Database Behavior in a Real Mashup

Once the relational database schema proposed has been deployed in a database over a real environment, it is possible to access to all the interactions that have occurred with a great detail. In this case, we deployed the relational database in ENIA, the mashup interface previously described, which focuses on the management of environmental information. The real implementation of the database has more tables and the mashup interface has more operations compared to the set of fields and operations we have discussed previously in this paper. But, as a matter of fact, the operations and tables described are present. The next piece of SQL code queries a MySQL database deployed in a platform as a service cloud infrastructure provided by Azure. ClearDB provides the MySQL databases in Azure as database as a service. This query extracts all the operations that have been performed by users and sessions specifying in each case the kind of UI upon which the interaction was performed (browser, mobile browser, tablet app, smartphone app...) as well as the type of interaction used to perform the operation (mouse, keyboard, touch, gesture, voice...).

```
SELECT interactions.idInteraction, interactions.dateTime,
   interactions.operationPerformed, interactions.Sessions_idSession,
   interactions.Users_idUser, sessions.deviceType,
   sessions.interactionType
FROM interactions, sessions, users
WHERE interactions.Users_idUser=users.idUserClient AND
   interactions.Sessions_idSession=sessions.idSession
```

Figure 3 presents the data obtained from the SQL code shown before. We can be distinguish between add, move and delete operations. All of them have been performed in a desktop or laptop browser and the form of interaction has been made by touching the laptop or computer screen.

idInteraction	dateTime	operationPerformed	Sessions_idSession	Users_idUser	deviceType	interactionType
9631	2016-03-08 11:23:57	Add	691	1	Browser	Touch
10271	2016-03-09 09:15:12	Add	711	1	Browser	Touch
9611	2016-03-07 13:39:55	Add	671	31	Browser	Touch
12961	2016-03-11 18:06:21	Add	681	31	Browser	Touch
12971	2016-03-11 18:06:21	Add	681	31	Browser	Touch
12981	2016-03-11 18:06:21	Add	681	31	Browser	Touch
13021	2016-03-11 18:08:49	Move	741	31	Browser	Touch
13031	2016-03-11 18:08:57	Move	741	31	Browser	Touch
13041	2016-03-11 18:10:09	Delete	751	31	Browser	Touch
13051	2016-03-11 18:21:42	Add	681	31	Browser	Touch
13061	2016-03-11 18:21:42	Add	681	31	Browser	Touch
13071	2016-03-11 18:21:43	Add	681	31	Browser	Touch
13081	2016-03-11 18:21:43	Add	681	31	Browser	Touch
13091	2016-03-11 18:21:43	Add	681	31	Browser	Touch

Fig. 3. Results from the query to the interaction db

This database allows to access to every operation performed in the UI from heterogeneous devices; it also enables to recreate the user behavior step by step by analyzing the *workspace*, as it has been later each operation performed, for a better understanding of the user's behavior.

5 Conclusions and Future Work

This paper proposes a relational database to allow mashup UIs to store the interaction performed by users over them. The database is valid even when the interface runs in distributed heterogeneous devices that support different interactions modes. A clear definition of a mashup GUI morphology has been made in order to study it and suggest a relational database that can save the interaction with accuracy.

The creation of a data acquisition process is proposed as future work. This data acquisition process could be a microservice than runs in the cloud and is continuously listening to the request from different mashup UIs distributed in multiple devices. It can just receive all the data directly from the client and store it or even make some requests to the mashup UI services, if any, to obtain extra information about users or components. Moreover, it can make a request to third party services that can provide valuable context information.

The information stored in the database proposed contains valuable information about the users' behavior. Machine learning experiments can be performed for the creation of an automatic learning system that can be used to offer a better

user experience. The discovery of behavioral patterns gives the opportunity to create prediction models that assist users, providing them with the components they are most likely to need, including the shape, size and layout configuration properties they expects.

A future deployment of the data acquisition process that storage the interaction in the relational database proposed and the machine learning experiments can be used over the work shown at Criado *et al.* [11], where component-based interfaces are adapted at run time using model transformation according to a set of rules. The new rules generated can update the rules repository making the application autonomously evolve over time [12].

Acknowledgments. This work was funded by the EU ERDF and the Spanish Ministry of Economy and Competitiveness (MINECO) under Project TIN2013-41579-R and under a FPI Grant BES-2014-067974 and by the Andalusian Regional Government (Spain) under Project P10-TIC-6114. Wang has been funded by the US National Science Foundation.

References

1. Elmqvist, N.: Distributed user interfaces: state of the art. Distributed User Interfaces DUI 2011. Human–Computer Interaction Series, pp. 1–12. Springer, London (2011)
2. Daniel, F., Matera, M.: Mashups - Concepts, Models and Architectures. Springer, Heidelberg (2014)
3. Vallecillos, J., Criado, J., Padilla, N., Iribarne, L.: A cloud service for COTS component-based architectures. Comput. Stand. Interfaces **48**, 198–216 (2016). Elsevier
4. Fernández-García, A.J., Iribarne, L.: TDTrader: a methodology for the interoperability of DT-Web Services based on MHPCOTS software components, repositories and trading models. In: 2nd International Workshop of Ambient Assisted Living (IWAAL2010), pp. 83–88 (2010)
5. Roscher, D., Lehmann, G., Schwartze, V., Blumendorf, M., Albayrak, S.: Dynamic distribution and layouting of model-based user interfaces in smart environments. In: Hussmann, H., Meixner, G., Zuehlke, D. (eds.) Model-Driven Development of Advanced User Interfaces. SCI, vol. 340, pp. 171–197. Springer, Heidelberg (2011)
6. Hoyer, V., Fischer, M.: Market overview of enterprise mashup tools. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 708–721. Springer, Heidelberg (2008)
7. Geckoboard. Commercial mashup KPI dashboard. https://www.geckoboard.com/
8. Cyfe. Commercial mashup business dashboard. https://www.cyfe.com/
9. The Andalusian Environmental Information Network (REDIAM). http://www.juntadeandalucia.es/medioambiente/site/rediam
10. ENIA. Environmental Inf. Agent. http://acg.ual.es/projects/enia/ui/
11. Criado, J., Rodríguez-Gracia, D., Iribarne, L., Padilla, N.: Toward the adaptation of component-based architectures by model transformation: behind smart user interfaces. Softw.: Pract. Experience J. **45**(12), 1677–1718 (2015)
12. Fernandez-Garcia, A.J., Iribarne, L., Corral, A., Wang, J.Z.: Evolving mashup interfaces using a distributed machine learning and model transformation methodology. In: Ciuciu, I., et al. (eds.) OTM 2015 Workshops. LNCS, vol. 9416, pp. 401–410. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26138-6_43

Virtual Spatially Aware Shared Displays

Felix Albertos-Marco(✉), Victor M.R. Penichet, and Jose A. Gallud

Computer System Department Albacete,
University of Castilla-La Mancha, Albacete, Spain
{felix.albertos,victor.penichet,jose.gallud}@uclm.es

Abstract. Nowadays, sharing resources over multiple devices is a common task. Some approaches consist in sharing a common workspace among users, or moving user interface elements between displays. But distributing interaction between displays is critical in cross-device environments. In this work, we present a technique for distributing content and devices in shared workspaces using cross-device displays. This technique, referred to as the virtual spatially aware technique, allows the creation of virtual shared displays and the coordination of cross-device interactions. By using this technique, we propose a method for arranging content and devices on virtual displays. We also present a prototype that supports the virtual spatially aware technique. This prototype has been built using web technologies, and it is able to run in any modern web browser.

Keywords: Distributed interaction · Virtual spatially aware · Shared workspaces · Content representation

1 Introduction

There are many approaches for the distribution of interaction among multiple devices. Albertos et al. presented synchronous interaction over shared resources in Drag&Share [1]. They provided a shared workspace which is the same for all users. Other approaches are aimed at moving user interface elements between displays according to a device's characterization [4]. But these approaches do not take into account the arrangement of cross-device displays within a shared display.

Radle et al. [3] initiated the debate about the use of spatially-aware (use of real-world spatial configurations as the referential domain) or spatially-agnostic cross-device interaction techniques. Their results showed that spatially-aware techniques are preferred by users. There is also no consensus on how to make the mapping between input and output in applications using multi-display and/or cross-device interactions.

In this work we present a technique for the design of virtual spatially-aware cross-device interaction. Using a planar technique, which is easy to match to the space [2], we propose the use of virtual spatially aware shared displays for supporting the creation of shared interaction spaces using cross-device displays.

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 196–199, 2016.
DOI: 10.1007/978-3-319-46963-8_19

This new technique is based on the arrangement of devices and resources using a virtual display and distributing interaction over multiple devices. It is intended to be used to share resource interaction on multiple displays or to facilitate the set-up of multi-display screens.

2 Arranging Content on Virtual Spatially Aware Shared Displays

The use of virtual spatially aware shared displays allows the management of devices and resources in a common interaction area. This is achieved through the use of virtual elements that represent devices and resources that are virtually arranged on the virtual display. This arrangement might depend on physical or logical conditions.

Figure 1 shows a virtual display on which multiple resources are shared among multiple devices, which are represented by a semi-transparent gray square with a number representing their ID. The device will display the resources according to the representation of the virtual display. For example, the device with ID 122 will show the picture of a graph. These representations can be moved freely within the virtual display to show other resources or to make compositions with other displays. For example, real displays which are represented by IDs 102, 107, 112 and 117 are arranged as one big display. Therefore, they will show the area of the virtual display as a real and larger display, with the corresponding resources on it.

A support tool has been developed to support the virtual spatially aware shared displays. This tool allows the management of virtual displays, devices and content. In the following figures the virtual display is shown within a green square, device displays are in a pink square, and resources are shown within a blue square. Mapping between virtual and real elements is represented using arrows.

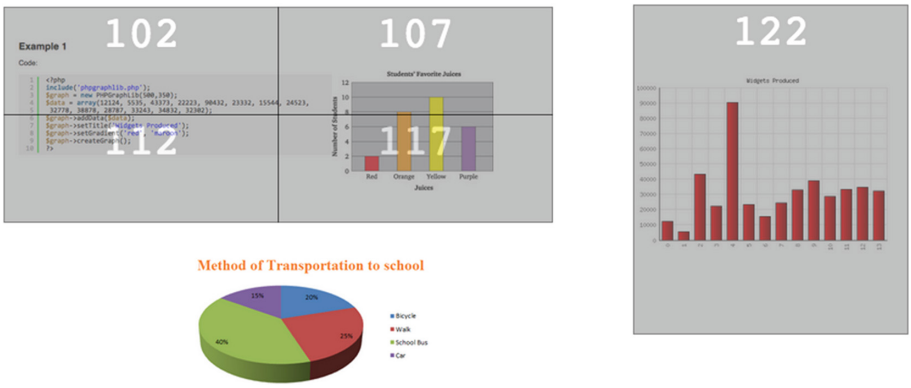

Fig. 1. Virtual display sharing resources among multiple devices

Figure 2 shows a scenario with a computer managing the virtual display and other devices (laptop, desktop pc and mobile phone) connected to the virtual display. It is worth pointing out that on the computer that manages the virtual display there are two browsers representing devices connected to the virtual display. There are no limitations on the location of virtual or device displays. Both systems work over web technologies and only require a modern browser to be run. Devices are connected through a browser to the virtual display.

Figure 3 shows how resources are managed using a virtual display. The set-up consists of two displays connected to two computers and another computer that hosts the virtual display. The resource in the virtual display is an image, but it can be any web resource. The devices on the virtual display are arranged so that the image is between them. As a result, the image generated by the two displays corresponds to the arrangement on the virtual display. The image is within a blue square, both on the real multi-display and in the virtual set-up. In addition, any interaction with the image on the virtual display or on the connected devices will be reflected across the entire environment (e.g. the movement of the image on any device).

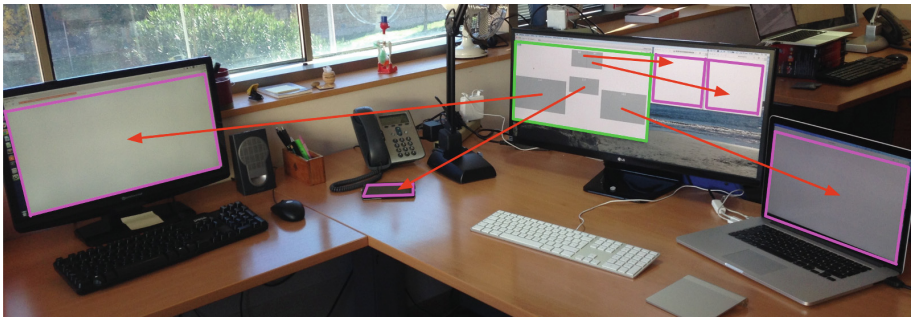

Fig. 2. Mapping between devices and the virtual display

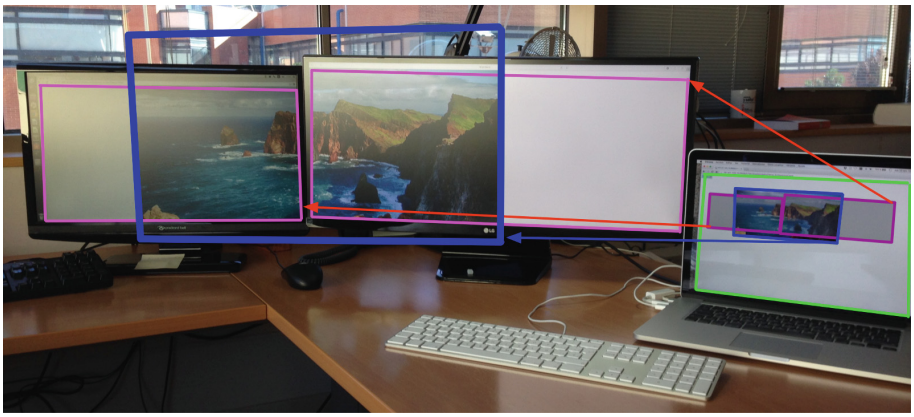

Fig. 3. Two displays arranged on the virtual display showing a resource

3 Conclusions and Future Work

In this work we have presented a tool for supporting the virtual spatially aware shared displays interaction technique. This technique allows the arrangement of content and devices on a virtual display. It supports cross-device interactions. As future work, we will continue with the development of the support tool. This development will allow us to perform an analysis of more sophisticated interaction techniques within the system and also, among others, the study of the impact of positioning devices on the virtual display according to their real position in the physical world.

Acknowledgments. This work has been partially supported by grant 2014/10340 from the University of Castilla-La Mancha.

References

1. Marco, A.F., Penichet, V., Gallud, J.A.: Collaborative e-learning through drag & share in synchronous shared workspaces. J. Univ. Comput. Sci. **19**(7), 894–911 (2013). http://dx.doi.org/10.3217/jucs-019-07-0894
2. Nacenta, M.A., Gutwin, C., Aliakseyeu, D., Subramanian, S.: There and back again: cross-display object movement in multi-display environments. J. Hum.-Comput. Interact. **24**(1), 170–229 (2009). http://pdfserve.informaworld.com/764323_770885140_910602221.pdf
3. Radle, R., Jetter, H., Schreiner, M., Lu, Z., Reiterer H., Rogers, Y.: Spatially-aware or Spatially-agnostic? Elicitation and evaluation of user-defined cross-device interactions. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI 2015), pp. 3913–3922 (2015). ACM, New York. http://doi.acm.org/10.1145/2702123.2702287
4. Yang, J., Widgor, D.: Panelrama: enabling easy specification of cross-device web applications. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2014), pp. 2783-2792. ACM, New York. http://doi.acm.org/10.1145/2556288.2557199

Flexible Distribution of Existing Web Interfaces: An Architecture Involving Developers and End-Users

Sergio Firmenich[1(✉)], Gabriela Bosetti[1], Gustavo Rossi[1], and Marco Winckler[2]

[1] LIFIA, Facultad de Informática, Universidad Nacional de La Plata and CONICET, La Plata, Argentina
{sergio.firmenich,gabriela.bosetti,
gustavo}@lifia.info.unlp.edu.ar
[2] ICS-IRIT, University of Toulouse 3, Toulouse, France
winckler@irit.fr

Abstract. This paper presents a novel approach towards the opportunistic and lightweight distribution of existent Web User Interfaces. We describe an architecture that allows end-users to collect UI objects into a distributed UI-Component-oriented PIM, accessible from different user's devices. Once in the PIM, the collected UI components are wrapped with different DUI-based behaviours that may be triggered by the user, as PIM's objects plug-ins. We present an overview of the architecture; some default DUI-based behaviours are introduced and illustrated through examples. Besides, we show how the architecture supports the development of new DUI-based behaviours.

Keywords: Client-side adaptation · DUI · End-user development

1 Introduction

The distribution of user interfaces (DUI) has been a growing trend in the last ten years. Beyond the understanding about how DUI applications can improve user experience [12], several works for engineering DUI Web applications have emerged [10, 13]. Even more, approaches for more specific cross-device interaction support has been defined, such as the use of Kinect [11]. However, it is still hard to find popular Web sites or applications supporting DUI.

When applications do not offer features that users may need, experience has shown that the crowd react trying to satisfy these needs by itself. This is a very common practice in Web Browsing Augmentation, i.e. using tools (deployed such as Web Browser Extensions) to augment Web application capabilities. To cite one example, simple solutions for cross-device interaction such as "Slides – Presentation Remote"[1] has more than sixty thousand users, just offering a remote control for presentations in some well-known Web applications (Google Drive, SlideShare, Prezi, etc.). Examples like this clearly show that while ad-hoc developers may create this kind of experience, there are also users expecting them.

[1] https://chrome.google.com/webstore/detail/slides-presentation-remot/mhfdnafbhfglkcjgk-goopjoadaopcomi.

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 200–207, 2016.
DOI: 10.1007/978-3-319-46963-8_20

We started this work by analyzing how to apply the lessons learned in Web Augmentation [3] into the field of DUI. Web Augmentation comprises those approaches that aim to adapt content and functionality of existing (usually third-party) Web applications, once these are loaded on the client. This technique has been used successfully in different domains, such as mash-ups [17] or end-user driven Web tuning [4]. Web Browser Augmentation is a perfect target for some DUI applications, such as supporting opportunistic remote control, layout distribution or UI migration.

In this paper, we propose to involve not only developers but also end users in the process of user interface distribution. The main idea is to provide developers with an easy way to implement DUI layers over existing Web pages, while end users may decide how to apply such layers on their preferred Web sites. This is achieved by managing a UI-Component-oriented PIM, where users may collect the target UI components that then can be manipulated from the applications defined by developers. The paper presents the overall architecture and the supporting tools through some case studies.

The paper is structured as follows. First, we present the related works in Sect. 2. Section 3 introduces our approach and the main components of the underlying architecture. The supporting tools are illustrated via case studies in Sect. 4. Finally, we discuss our contribution in Sect. 5, in conjunction with our future work.

2 Background and Related Works

Since the early years of Web Augmentation [2] (WA) and Mash-Ups applications [5], several approaches have emerged for adapting or integrating existing Web contents. Diverse communities of users, both developers and amateur end users with technical know-how, set up the basis and contributed in the creation of new repositories of augmentation artefacts, which improved the Web with extra features that original Web applications did not contemplate. Such is the case of Greasyfork, or Mozilla addons. Some End User Development [9] (EUD) works arose later in those fields, to empower users to solve their particular needs by themselves. Concerning DUI in conjunction with the WA and EUD fields, and as pointed out in [14], we can appreciate that it is not easy to provide DUI for existing and third party Web content, although some approaches have tackled isolated specific dimensions, specifically involving end-users. User-Driven DUI was previously defined in [15]; however, although this approach does not consider third-party existing Web sites as potential targets, other approaches do. For instance, [7] lets users to annotate some parts of exiting Web UI in order to migrate components under user demand. The main idea is to allow users to access a Web application from a desktop environment, and then to continue the interaction with it from a mobile device, migrating those annotated portions of the UI. Other similar approaches using proxies are Proxywork [16] and WebSplitter [8]. And there are also approaches addressing flexible interface migration [1].

In this paper, we present an architecture to apply DUI layers over existing Web content. But in contrast with the existing works, we do not provide an end-user tool for performing specific DUI applications, but an architecture to enable developers from Web Augmentation communities to easily define new kind of DUI applications, by

taking advantage of the underlying distributed UI-Component-oriented PIM. However, since we share the philosophy behind empowering end-users with specific tools, our approach lets them to instantiate a DUI application in their preferred Web sites.

Our approach may support several kinds of DUI dimensions, such as light interface migration, remote control, distributed layout, etc. We have designed a client-site visual tool that lets users to specify how to distribute the UI of any existing Web site.

3 Distributing Web UI Objects

The main idea behind our approach is to provide end users with a specialized PIM with extra WA and DUI capabilities. Instead of collecting and structuring personal data such as traditional PIMs, we propose a UI-component-oriented PIM. This allows users to collect arbitrary DOM elements from existing Web sites. A DOM element represents a particular component of the user interface. These UI components are collected into the PIM and materialized as UIObjects. Our approach rests on the idea that a UIObject may be used by the end-user for triggering different behaviours supporting DUI. We call this kind of actions DUI behaviour. For instance, the user may activate the DUI behaviour "Close on other devices…" for a UIObject. This action would hide that UIObject in any other device registered in the platform by the user, and it would be functional only in the current device. Furthermore, although users may use some predefined "DUI behaviours", developers may create new ones. We call them behaviours because these are performed individually for a UIObjects; however, if several UIObjects are collected for a specific Web site, and different behaviours are executed with each of them, then a more complex DUI experience (involving combinations of objects) could be defined. In this way, at the end, the addition of several DUI behaviours may abstract a specific kind of distributed use of a Web site. For instance, a simple distributed layout could be supported if, after collecting several UIObjects, the user executes the "Close on other views…" behaviour for different UIObjects running in different devices/monitors. Meanwhile, other combined use of these behaviours may be oriented to control the UI displayed on a desktop computer from a mobile device. Since it is not our purpose to foreseen every possible DUI behaviour we propose instead a flexible architecture based on this UI-object PIM. The architecture was designed with two premises in mind: (1) users should be able to collect UIObjects into the PIM and use a specific DUI behaviour with them, and (2) developers should be able to create new kinds of DUI behaviours that may be added to the collected UIObjects. If the user does not select a behaviour, this is set by default according to the type of DOM element that was collected.

The overall idea is that by triggering some behaviour, the user is enriching an existing Web site with DUI features. In order to do that, users are allowed to annotate some existing DOM elements as UIObjects, that will be collected into the PIM and then, presented in another view. In this section, we introduce the components and concepts of the approach and some aspects about the architecture. Then we show how end-users may create and use UIObjects.

3.1 Underlying Architecture

As shown in Fig. 1, our approach mostly works at client-side. The UIObject-PIM is a Web Browser extension that, once installed in different devices, it allows users to share a unique space of information among them. In order to maintain the UIObjects synchronized, we provide a synchronization server (accessible from our Web Browser extension) that allows an instance (UIObject) of a particular UI component to be synchronized, by acting as a mediator among all the contexts where such instance is running. The approach is no constrained to a centralized server, the browser extension may be configured to work with any specific deployment of that server.

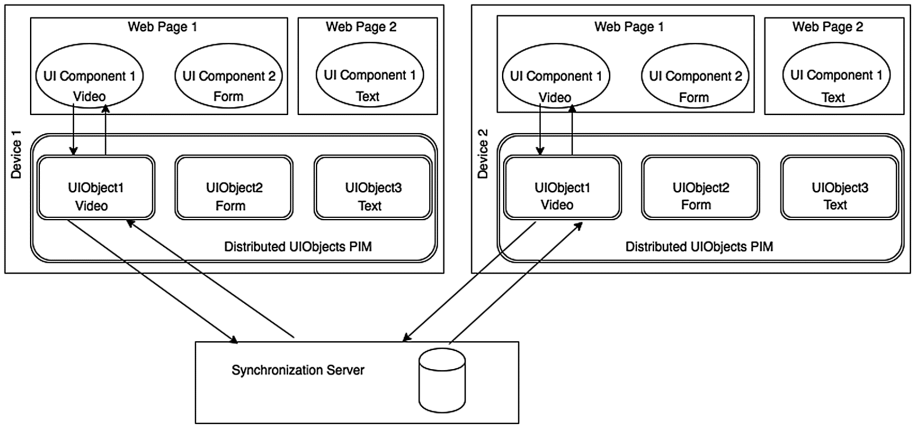

Fig. 1. Architecture schema of the UIObject-PIM platform

In this paper, we focus on the client-side features of this architecture, in which we can find three main artefacts:

- UI Component: in this way, we refer to the native UI components or widgets that compose a particular Web site. For instance, a UI Component could be a form, a video, a part of the DOM tree, etc. The main idea is that end-users may select under demand those UI Components of their interest. Through this selection, a UIObject is generated from a UI Component.
- UI Objects: these are representations of UI Components enriched with some behaviour that they do not have associated by default (in the original Web page). These UI Objects live in the UIObject-PIM.
- UI-Component-oriented PIM: this is a PIM oriented to maintain references to those UIObjects created by end users. This PIM requires authentication, so the user may collect and retrieve his preferred UIObjects from any of his devices. The UIObjects collected into the PIM are not just façades of UI Components, but managers of UIComponents with specific DUI-based behaviour added to its corresponding UIObject. This behaviour is materialized as operations that are executed for a particular UIObject, and it is executable directly by the user. For instance, if the user wants to render a UI Component only in one of his devices, then he must run the "Show only

in…" operation, that will ask the user which is the target device and then it will carry on the desired UI effect.

In this way, the PIM maintain the current state of the DUI model based on the operations (DUI-based behaviour) that were executed for the collected objects. In this way, the UI Component is like a view (in the same way as in the MVC pattern).

3.2 UIObjets and DUI-Based Behaviours in Detail

UIObjects are JavaScript objects abstracting UIComponents. During the abstraction process, the user may choose a target UIComponent. Figure 2 shows in (1), a user enabling the DOM selection and, in (2), a DOM element being highlighted for selection, with a proper context menu enabling the harvesting. Once selected, he can configure some of its aspects. For instance, give it a name and associate it some properties, as shown in (3). One of the most relevant properties is the UIComponentStereotype. This property allows users to choose among different kinds of UI widgets, such as images, text, forms, videos, etc. Once defined, all the UIObjects are available in the PIM, as the one in (4), so the user can interact with the offered behaviour.

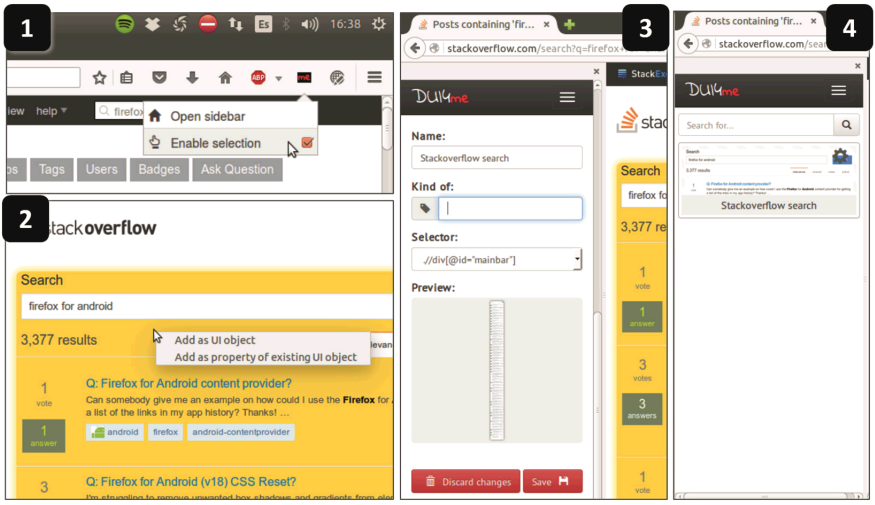

Fig. 2. Defining a UIObject

When a UIObject is collected into the PIM, the matching DUI-based behaviours (those preferred by the user) are attached to it under the basis of the decorator pattern [6]. There are two kinds of DUI-based behaviours. First, we can have stereotype-agnostic behaviours that may be attached to every UIObject since their goals are compatible with all kind of UIComponents. For instance, showing a particular UIObject only in a particular device. On the other hand, stereotype-specific behaviours are attached only to those UIObject that were collected as a specific UI stereotype, such as a YouTube Video. In this case, more specific operations such as *"Play video on…"* can be defined.

It is worth mentioning that the set of decorators applied to the UIObject can be configured by the end-user, by clicking the gear icon shown in Fig. 1; after doing this a context menu opens and the user has to select «manage» and finally «applied behaviour». The list of decorators is presented, and a series of parameters can be set.

Although we provide some basic behaviours, new ones may be defined by developers. To develop a new behaviour requires programming with client-side Web technologies (HTML, JavaScript, CSS), specially using a JavaScript API that allow developers to access, manage and add behaviour to UIObjects collected into PIM, considering that the communication mechanism among devices is solved transparently.

New behaviours may address new kind of DUI applications (for instance a particular kind of distributed layout), but may also perform specific operations wrapped in messages that the objects into the PIM may respond to. For instance, a developer may define a new behaviour for controlling a YouTube player by defining messages such as «*stop*» and «*play*». Then, if a user collects this object into the PIM and apply the decorator, then these messages could be sent from any device transparently. In these cases, developers are benefited with the underlying synchronization mechanism among PIMs state from different devices.

4 Case Study and Prototype

Although the approach lets developers to create new DUI behaviours, we have defined some predefined ones. So far we have identified: (1) opportunistic and volatile interface migration, (2) multi-monitor layout, (3) distributed layout, (4) messaged-based remote control, (5) navigation control and (6) remote UI Effect. For the sake of space, we are covering just one of them, in order to depict the feasibility of the approach.

At the time of this publication, we have partially implemented two prototype client-side tools supporting the approach. The first one is a Firefox 44 extension for desktop edition, the second one is a Firefox for Android extension, for version 42.0a1. At server-side, we implemented a minimal functionality to synchronize changes of the UIObjects both tools manage.

Consider a scenario where Máximo is living abroad for some months and he wants to write about every detail of his experiences in his blog. There are moments when he can write an entire entry from his computer, but he must migrate the task to his phone often, because his job requires constant mobility. His mother tongue is Spanish, but he is writing the blog in English. Sometimes, he have some doubts about language expressions, so he checks it out in linguee.com. It is desirable for him to have a mean for distributing some elements of the Linguee and Blogger Web sites.

Figure 3 shows our tool sidebar, with some UIObjects created from both sites; a search box and a results box from Linguee, and then a Blogger's entry main options, and a text area. So, when Máximo has to leave his house, he may open such elements in his mobile, the«XT1021» listed device, as shown in the left-bottom area of Fig. 3. Then, from his device, the browser extension will be notified to open the same URL and just open the defined UIObjects the user has defined for such URL.

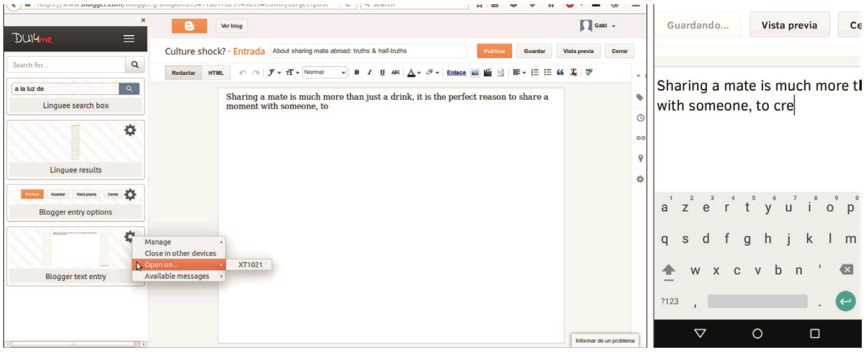

Fig. 3. Distributing UI components from a web application to mobile

While the user changes the document, such modifications are notified to our server module, who asks every registered listener to update the view of the proper element. As we can see, the site functionality is not altered by our adaptation: the blog is still functional (you can see the automatic saving being executed in the right screen of Fig. 3). The advantages of accessing the blog through our platform is that: (1) Blogger does not provide a mobile version, so the same interface elements are presented in small devices; (2) While Blogger offers the ability to save changes in the entry, it does not offer the ability to keep the content up-to-date from two or more devices, and without our platform, it is required for every device to refresh the Web page every time they want to see the changes from another device.

5 Conclusions and Future Works

In this paper, we have presented an architecture for supporting the development of Web Browser Augmentation artefacts for the field of distributed user-interfaces. Browser augmentation oriented to DUI was also propose in previous works [7, 16]. We share the philosophy behind them, but we think that providing developers with an architecture that abstracts the more complicated technical issues underlying to DUI applications may help to create several kinds of new experiences by developing new DUI-based behaviours. Besides that, the same architecture contemplates a mechanism to allow end-users to instantiate those behaviours opportunistically in any Web site.

Currently, we are completing the implementation of the support system, which is partially operative. Once it is finished, we plan to carry on an evaluation with end-users in order to measure how useful and easy is for them to use our tools. We also plan to analyse new possible behaviours and study how this approach may raise new possibilities in the context of Web Augmentation, mash-ups and collaborative environments.

Acknowledgments. This work was supported by STIC AMSUD project WAMAW-OUR: Web Augmentation Methods for Adapting Web Sites for Supporting Opportunistic User Requirements

References

1. Bandelloni, R., Paternò, F.: Flexible interface migration. In: Proceedings of the 9th International Conference on Intelligent User Interfaces, pp. 148–155. ACM, January 2004
2. Bouvin, N.O.: Unifying strategies for web augmentation. In: Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia: Returning to Our Diverse Roots: Returning to Our Diverse Roots, pp. 91–100. ACM, February 1999
3. Díaz, O., Arellano, C.: The augmented web: rationales, opportunities, and challenges on browser-side transcoding. ACM Trans. Web (TWEB) **9**(2), 8 (2015)
4. Díaz, O., Arellano, C., Aldalur, I., Medina, H., Firmenich, S.: End-user browser-side modification of web pages. In: Benatallah, B., Bestavros, A., Manolopoulos, Y., Vakali, A., Zhang, Y. (eds.) WISE 2014, Part I. LNCS, vol. 8786, pp. 293–307. Springer, Heidelberg (2014)
5. Ennals, R.J., Garofalakis, M.N.: MashMaker: mashups for the masses. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 1116–1118. ACM, June 2007
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education, Upper Saddle River (1994)
7. Ghiani, G., Paternò, F., Santoro, C.: On-demand cross-device interface components migration. In: Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services, pp. 299–308. ACM, September 2010
8. Han, R., Perret, V., Naghshineh, M.: WebSplitter: a unified XML framework for multi-device collaborative web browsing. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, pp. 221–230. ACM, December 2000
9. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm, pp. 1–8. Springer, Dordrecht (2006)
10. Melchior, J., Vanderdonckt, J., Van Roy, P.: A model-based approach for distributed user interfaces. In: Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 11–20. ACM, June 2011
11. Nebeling, M., Teunissen, E., Husmann, M., Norrie, M.C.: XDKinect: development framework for cross-device interaction using kinect. In: Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 65–74. ACM, June 2014
12. Santosa, S., Wigdor, D.: A field study of multi-device workflows in distributed workspaces. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 63–72. ACM, September 2013
13. Schreiner, M., Rädle, R., Jetter, H.C., Reiterer, H.: Connichiwa: a framework for cross-device web applications. In: Proceedings of the 33rd ACM Conference Extended Abstracts on Human Factors in Computing Systems, pp. 2163–2168. ACM, April 2015
14. Vanderdonckt, J.: Distributed user interfaces: how to distribute user interface elements across users, platforms, and environments. In: Proceedings of XI Interacción, vol. 20 (2010)
15. Vandervelpen, C., Vanderhulst, G., Luyten, K., Coninx, K.: Light-weight distributed web interfaces: preparing the web for heterogeneous environments. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 197–202. Springer, Heidelberg (2005)
16. Villanueva, P.G., Tesoriero, R., Gallud, J.A.: Proxywork: distributing user interface components of web applications. In: DUI@ EICS, pp. 58–61, June 2013
17. Wong, J., Hong, J.I.: Making mashups with marmite: towards end-user programming for the web. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1435–1444. ACM, April 2007