Towards End-User Development of Distributed User Interfaces

Sanctorum, Audrey; Signer, Beat

Published in:
Universal Access in the Information Society

DOI:
10.1007/s10209-017-0601-5

Publication date:
2019

License:
Unspecified

Document Version:
Accepted author manuscript

Universal Access in the Information Society (UAIS), 2019
The final publication is available at Springer via
https://doi.org/10.1007/s10209-017-0601-5

Noname manuscript No.
(will be inserted by the editor)

Towards End-User Development of Distributed User Interfaces

Audrey Sanctorum · Beat Signer

Received: date / Accepted: date

Abstract Over the last decade we have seen an increasing number of solutions for distributed user interfaces (DUIs). This paper provides a detailed overview of existing DUI approaches and classify the different solutions based on the granularity of the distributed UI components, the supported interaction space as well as their support for the distribution of state. After the analysis of existing solutions, we discuss a DUI scenario and derive a number of requirements for end-user DUI development. We propose an approach where users can author their customised user interfaces based on a hypermedia metamodel and the concept of active components. We further discuss possibilities for the configuration and sharing of customised distributed user interfaces by end users where the focus is on an authoring rather than a programming approach.

Keywords distributed user interfaces · cross-device interaction · end-user development

1 State of the Art in Distributed User Interfaces (DUIs)

Over the last two decades, distributed user interfaces (DUIs) have gained a lot of attention [33]. Various terms have been introduced in order to differentiate between different DUI systems, ranging from multi-device and multi-display environments 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. [36] presented a system that allowed users to interact with a television by using a personal digital assistant (PDA) and a stylus. A limitation of this early system was the one-directional information flow from the PDA to the TV, which prevented users from capturing and moving information from the TV to their PDA. Just about a year later, Rekimoto [35] introduced the *pick-and-drop* technique allowing users to exchange information in both directions by picking up an object on one computer screen and dropping it on another display via a stylus. Since these early systems, research in cross-device interaction has gone a long way and various interaction techniques, frameworks and applications have been developed [13].

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

1.1 Interaction Techniques and Devices

In order to distribute information across devices, many techniques and input devices with cameras or sensors have been introduced. As described before, both Rekimoto [35] and Robertson et al. [36] used a stylus for allowing users to transfer information between devices. A few years later, Han et al. [19] presented WebSplitter which allows a web page to be distributed among multiple users and displays. Furthermore, WebSplitter can also be used to split views of a web slideshow presentation across devices. The presenter can, for example, navigate through the slides via a wireless PDA, while students can follow the presentation on their own devices (PDA or laptop). An XML metadata policy file containing all predefined access privileges is used to provide users a different view of a web page or presentation. By using this policy file, WebSplitter somewhat limits the user interaction. The file defines which XML tags can be shared between which user groups and builds

partial views on the users' devices. These views cannot be changed by the user once the system is set up. Moreover, for each web page or presentation a separate policy file needs to be created. Most recent DUI systems offer users more freedom for the distribution of data and applications across devices. However, as discussed in the remainder of this section, there are still some limitations in terms of what can be distributed, where it can be distributed and how it is distributed. An overview of the discussed interaction techniques together with the corresponding used systems and devices is provided in Table 1.

A simple method to distribute data that has also been widely used to forward people to a website are QR codes. For example, Frosini et al. [15] make use of QR codes to enrich a museum visit by allowing visitors to distribute images, videos and texts between their smartphones and larger displays. Users run an application on their smartphone which is used to scan the QR code on large public displays in the museum in order to establish a connection and distribute selected content to these displays. Di Geronimo et al. [16] used QR codes for establishing a connection between devices in aCrossETH, where users can upload, favourite and like images. The most popular images are shown on a slideshow screen, while all recently uploaded images are displayed on one or multiple voting diplays. By scanning a QR code on the slideshow and voting displays, users can connect their mobile device in order to interact with the voting displays. The first connected user will control the browsing of images by swiping left and right, while other connected users can see the selected image in the role of a viewer. Users can vote for an image by a *hold tap* gesture at the bottom of their smartphone's screen and a quick *tilt down* of the device. The aCrossETH application has been developed based on the Cross-Tilt-and-Tap (CTAT) framework which allows users to utilise the tilting of devices for other interactions as well.

Another system making use of QR codes for connecting devices is Panelrama by Yang and Wigdor [50]. Panelrama is a web-based framework that divides a user interface into panels and distributes these panels across different devices depending on their characteristics. A YouTube video could, for example, be automatically distributed such that the video itself is shown on a large screen, the comments section on a laptop and the video controllers on a smartphone. Since for some users the automatic distribution of the panels might be disruptive, a toggle can be used to lock the panel to a certain device. Furthermore, in their YouTube video application the authors also demonstrate other functionality such as the *push* and *pull* technique which can be used to push a video to the global panel or copy the video to the device by tapping the corresponding icons.

Instead of scanning a QR code, some systems use a smartphone camera to take a picture of another screen for initiating an information exchange. Based on this technique, Chang et al. developed the Deep Shot framework [7] which supports information sharing between a smartphone and a computer screen. The authors introduce the two new interaction techniques of *deep shooting* and *deep posting*. *Deep shooting* is used to pull documents and application states from one device to another while *deep posting* pushes them from one device to another. After taking a picture of a screen with the smartphone, Deep Shot runs an image matching algorithm in order to convey the data to the right device. A similar setup has been introduced by Leigh et al. [25], where a smartphone is used as magic lens to offer near-surface interaction between the smartphone and a computer screen. The authors present a system called THAW and explain various interaction techniques. THAW allows the smartphone to be used as a physical token for interacting with digital items depending on their position on the screen. The smartphone can, for example, be used as a *container* to drag and drop items on the screen or as a *pass-through filter* to translate data on the screen. It can also act as a lens to control or augment objects on the screen via a *see-through filter* or *click/touch-trough*. Furthermore, the smartphone can be used as a secondary screen offering additional space for a *clipboard* or *palette*. In order to accomplish these interactions, traditional touch gestures such as *drag and drop*, *tap* to copy or *pinch in* and *pinch out* to scale an item are used. Further, motion gestures are used to manipulate image properties, including *rotation* to blur or sharpen the image and *shaking* to add grain to the image.

By using a camera, spatial characteristics such as the relative position of two devices can be taken into account for specific interactions. THAW is not the only system taking advantage of these characteristics and many existing camera-based information sharing solutions also consider the spatial location of devices. Rädle et al. [34] introduced the HuddleLamp system which uses a lamp with an integrated camera to track hand movements as well as the position of any mobile device that has been placed on the table the lamp is standing on. With this setup, five interaction techniques to explore and share data across devices have been investigated. The first technique consists of moving a tablet on the surface of a table to explore a virtual map while in the second interaction multiple tablets can be combined to explore a larger map area at once. The role of a device can be adapted by changing its orien-

Interaction Techniques	System	Devices
deep shooting, *deep posting*	Deep Shot [7]	Smartphone and computer
QR code scanning, *tap*, *long press*	Frosini et al. [15]	Large screens and smartphone
QR code scanning, *push*, *pull*	Panelrama [50]	Microsoft Surface (Samsung SUR40) multi-touch table, PC, TV, smartphone and tablets
QR code scanning, *tap*, *double tap*, *hold tap*, jerk or continuous tilting: *tilt left*, *tilt right*, *tilt down*,...	CTAT [16]	Laptop screens, tablets, smartphones and smartwatches
tilt-to-preview, *face-to-mirror*, *portals*, *pinch-to-zoom*	GroupTogether [26]	2 overhead Kinect cameras, wall display SmartBoard and 8GHz band QSRCT radios
portals	UbiTable [40]	DiamondTouch multi-user interactive table [12], PC and laptops
portals	MultiSpace [14]	2 projectors (to display content), 2 fixed surfaces: DiamondTouch multi-user interactive table [12] and interactive wall
flicking, *touch-and-flick*, *pick, drag and drop*	Huddlelamp [34]	Desk lamp with an integrated low-cost RGB-D camera, smartphones and tablets
pick-and-drop	Pick-and-Drop [35]	WACOM stylus, wall-sized displays (computers), desktop displays, physical hand-held tablets and PDAs
lift-and-drop	Airlift [1]	Tablet PC, tabletop display, stereo camera and LED clusters
Drag and drop of application windows via a GUI	ARIS [5]	Wall-mounted plasma screens with touch-sensitive overlays, video wall, audio system, IR beacons, badge detectors, PDA and 2 graphics tablets
Room-control GUI to move info, *superpointer*	PointRight [24], iRoom [23]	3 touch white-board displays, interactive mural, table with display, various cameras, microphones and other interactive devices (e.g. PDAs, laptops)
On the DynaWall: *take and put*, *shuffle passage-mechanism*	i-LAND [46]	Interactive electronic wall, interactive table, 2 computer-enhanced chairs and 2 bridges for the passage-mechanism
Speech, hand gestures: *grip*, *grip release*, *press*	XDKinect [31]	2 displays and 1 Kinect
block, *container*, *pass-through filter*, *parent*, *force-field*, *see-through filter*, *click/touch-through*, *pop-up*, *palette*, *clipboard*	THAW [25]	Smartphone and computer
Tap or touch corresponding option in context menu, bonding of devices via *duets*	Conductor [18]	Tablets and smartphones
device stitching	Connichiwa [38]	A Microsoft Surface Pro 2, tablets and smartphones

Table 1 Different interaction techniques with the corresponding systems and used devices grouped by similarities

tation. For example, in portrait mode a device might be used to show some application menu. Finally, the remaining interaction techniques in HuddleLamp focus on distributing data via specific gestures such as *flicking*, *touch-and-flick* and the well-known *pick, drag and drop*. Similarly, Bader et al. [1] proposed the tracking of hands and fingertip gestures via the Airlift device consisting of a stereo camera and LED clusters. Users can move objects from a tablet to the table surface using the *lift-and-drop* interaction technique, which resembles the *pick-and-drop* technique [35] but offers continuous feedback while performing the movement. Cameras do not always have to be mounted on tables in order to enable cross-device interaction as shown in XDKinect [31], where a Kinect camera is placed next to two displays in order to browse and copy pictures between the displays. Speech commands and different hand gestures such as the *grip*, *grip release* and *press* gestures are used for interaction. In order to widen the interaction space, multiple cameras can be placed in a room as realised in the GroupTogether setup [26] consisting of two overhead Kinect cameras, a wall display SmartBoard and 8GHz band QSRCT radios to detect a small group of people. GroupTogether allows users to freely move in the room with their tablets and share information between tablets or between a tablet and the wall display by using four main interaction techniques. Content can be temporary copied from one tablet to another by using the the *tilt-to-preview* and *face-to-mirror* interaction techniques. A copy can be made permanent by touching it on the receiving device. An alternative way to transfer content is provided via the *portals* technique which enables the move of content between devices by dragging an item towards the tinted edge of the tablet and releasing it to move it to the tablet next to it. Note that portals have already been introduced

earlier in the UbiTable [40] and MultiSpace [14] prototypes for transfering content from a tabletop to other devices and vice versa.

A similar technique is used for the PointRight [24] system forming part of the iRoom project [23]. It allows any device to become a so-called *superpointer*. By using the spatial visibility of the room, the edges of the screen of the device running PointRight are associated with the edges of other surrounding displays. This allows a user to move the cursor through the edges of one screen to another and hence move control among the displays in an interactive workspace. In order to move an information object represented via a URL or a file icon, the room control system—a GUI showing a map of the room highlighting the displays, projectors and lights—can be used. Users can drag information into a region on the map in order to move the information to the corresponding display. Moreover, all the lights and projectors can also be controlled via the map. A similar technique has later been introduced in ARIS [5], an interactive space window manager showing an iconic map of the workspace in order to relocate application windows across devices in the room. Users can *drag* and *drop* the window representation on the map to the destination screen. Note that the use of a map for moving content or applications across a room of devices makes it difficult to add or remove devices to the interactive space.

Systems that do not use a fixed map of the interaction space often allow users to add devices by using NFC technology, bumping the devices, connecting to a URL, Bluetooth, installing an application or by using QR codes as mentioned earlier. For example, Conductor [18] enables users to add a new device by using either a QR code, NFC, bumping or proxies, where a physical QR code or NFC tag is placed next to the device to connect to. Once the connection has been established, users can share information between devices by broadcasting cues. When a user taps on a cue containing geographical information, it can be broadcast to all devices by choosing *broadcast* from a context menu or the user can target a specific device from the menu. Tapping the cue on the receiving device and dragging it to a specific application such as Google Maps, will directly show the address within the application. Conductor further supports the *bonding of devices* via *duets* which allow users to synchronise multiple devices. For instance, after a contact has been tapped on one device, a map showing the address of that contact can be shown on another device. The Conductor task manager provides an overview of the distributed space in the form of up-to-date screenshots of each device and can be used to clone the state of a device.

The synchronisation of devices is also provided in Connichiwa [38] through the use of *device stitching*, a synchronous gesture on two devices which resembles the well-known pinch gesture. When performing such a synchronous gesture, the relative position of the devices is determined in order to adjust the displayed content accordingly. For example, a large image can be displayed across multiple devices and panning on one device will then be reflected on all stitched devices. In order to unstitch a device, it can just be picked up or moved from its location.

1.2 Interaction Space

Distributed user interfaces can range from very restricted interaction spaces such as around the table interaction or interaction within a room to solutions without any space restrictions. In our classification of existing DUI solutions, the location where a solution can be used is therefore used as one of the dimensions as shown on the x-axis in Figure 1. We define four categories of space restrictions as introduced in [37]. The first category consists of systems having a setup that requires a camera and whose interactive space is limited to a table. In the second category we gather all the systems which still use cameras and other input devices but have a larger interactive space, normally covering an entire room. In the third category we group systems that work without the need for static cameras and sensors, but still rely on a connection with a central server. Finally, the last category contains all the solutions which neither rely on static cameras nor on sensors or a central server to enable cross-device interaction. Systems belonging to this last category can work almost anywhere. In the following, we analyse the interaction space of the DUI systems that have been introduced in the previous section.

Robertson et al. [36] enabled the interaction between a PDA and a television via infrared technology, limiting the interaction space to a few square meters. In a subsequent system by Rekimoto [35], an early form of WiFi was used to enable an information transfer across PDAs and allowed any item to be picked up with the stylus and dropped on any LAN-connected PDA in the room. An item could also be dropped on a wall-sized display (computer) or desktop display, which have to be directly connected to the Ethernet. In comparison to the solution by Robertson et al., Rekimoto's setup significantly broadened the interaction space. However, the wired connection of the displays limits the system to be used within the room where the displays are located. The physical palm-sized tablets in Rekimoto's

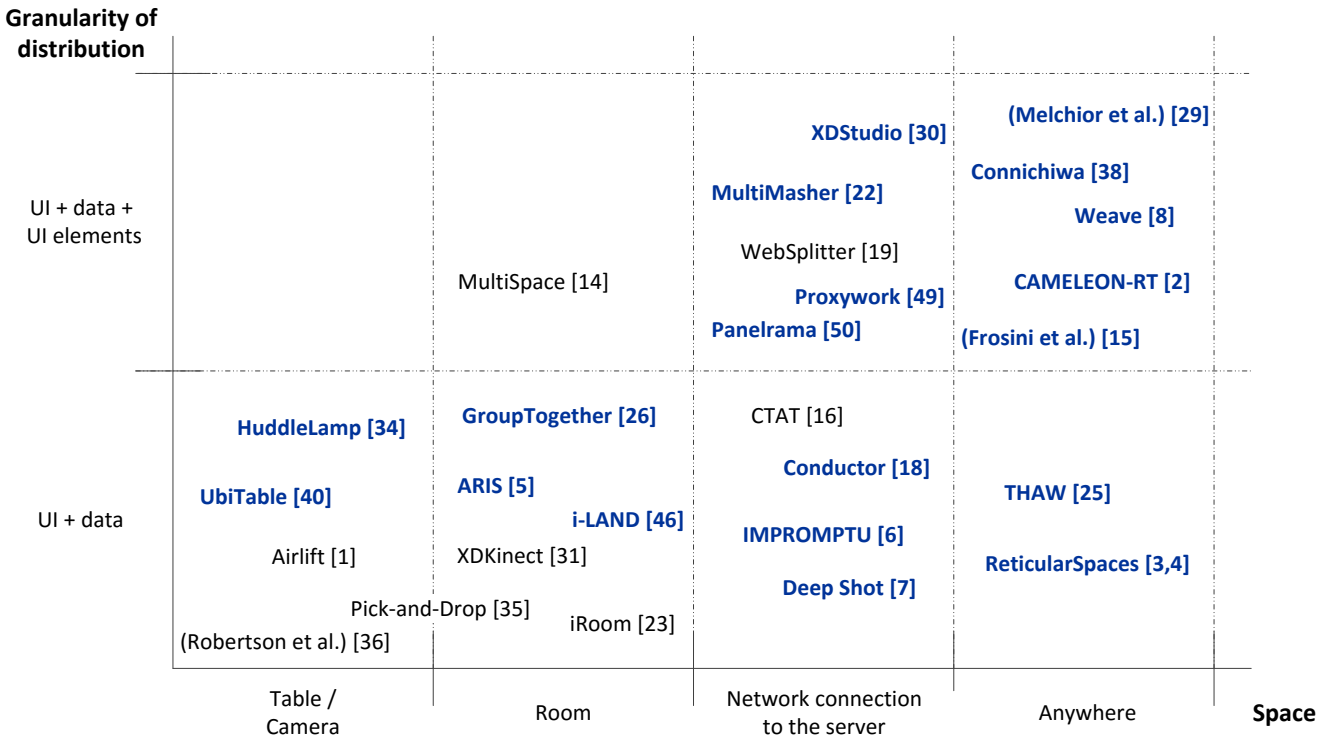

Fig. 1 Classification of DUIs based on location constraints (x-axis) and the supported granularity of distribution (y-axis). Solutions supporting the distribution of state are further highlighted in bold and blue. An updated online version is available at `http://dui.wise.vub.ac.be/classification/`.

approach have also been used in the PaperIcons prototype where users can pick a paper object and drop it on a computer display. The paper-digital interaction has been realised by placing a paper with an ID mark on a pen-sensitive tablet with a camera mounted above it. Hence, the overall system provided by Rekimoto still offers a limited interaction space since for some applications a fixed camera or wired connection is necessary.

Similar to Rekimoto's PaperIcons solution, other systems use a camera to monitor a table surface, including HuddleLamp [18], Airlift [1] and UbiTable [40], which limits their interaction space to the table surface. The use of multiple cameras might broaden the interaction space to room level as seen in XDKinect [31] and GroupTogether [26]. Solutions such as MultiSpace [14] and i-Land [46] use multiple projectors to increase the size of the interaction space while other systems such as iRoom [23] use a combination of both.

Often, cameras and projectors are not sufficient in order to cover the interaction space needed by a system and therefore a number of approaches use additional input devices to make a room more interactive. For instance, iRoom [23] uses three touch whiteboard displays, an interactive mural, a table with a display, various cameras, microphones as well as other interactive devices such as PDAs and laptops to support cross-device interaction across the workspace. By using an event heap, the interactive room allows users to move data around, control any device or application from their location and supports the coordination of applications. Based on the same infrastructure, Streitz et al. [46] developed i-LAND, an interactive landscape consisting of multiple computer-augmented objects such as two computer-enhanced chairs (CommChairs), an interactive electronic wall (DynaWall) and an interactive table (InteracTable). Later, ConnecTables [47]—small tables with a pen-sensitive display—have been added to the interaction space.

The previously described systems rely on customised setups with multi-touch tables, projectors and other input or output devices which limits their portability since they cannot just run in any lab or home environment. Furthermore, some of these solutions rely on the accuracy of the input devices as well as on pattern matching algorithms as seen in Deep Shot [7]. In contrast, Conductor [18] enables the distribution of information across different tablets via broadcasting. However, while these systems easily transmit and recognise transferred data, the distribution based on the spatial location of the devices is not possible anymore. On the other hand, the interactive space is no longer limited

to an interactive table [1,34] or room [5,14,23,26,46], since in Conductor devices do not rely on any input device and only need to be connected to a central server for exchanging data. This kind of setup and client-server architecture has also been used in many other cross-device systems [6,7,19,18,22,30,50,49]. Although these systems do not rely on static locations, they still need a server to pass data around the different devices. This implies that once the connection with the server is lost, information can no longer be spread across devices. This issue has been addressed in Weave [8], where the server runs on one of the interacting devices. Whenever this device should be disconnected, another device takes over its place. Other solutions overcome this limitation by using a peer-to-peer architecture [2–4,15,29,38]. Most of these systems enable the ad-hoc construction of an interaction environment by dynamically setting up a network between all peers. In order to set up a peer-to-peer network, an application must first be deployed on all interactive devices. For example, Bardam et al. [3,4], deployed a copy of their Activity Manager on each connected device to be able to create their ReticularSpaces system.

1.3 Distribution Granularity and State

Apart from the *space* dimension, other criteria can be used to highlight the differences between existing DUI solutions. In a second dimension, we classify existing DUIs into two categories depending on their level of granularity. The first category groups all the systems that allow images, documents or interfaces to be distributed as a whole, while the second category includes DUI solutions that also support the distribution of parts of images, documents or interfaces. An overview of the classification of existing systems based on their level of granularity is illustrated on the y-axis in Figure 1. Solutions supporting the distribution of state are further highlighted in bold and blue. In the following, we discuss the differences in the distribution granularity of existing DUI solutions as well as their support for state transfer during the distribution.

Some systems where application windows can be moved across devices, such as in ARIS [5], only support the distribution of applications as a whole. Similarly, in Rekimoto's solution [35] only entire items can be transferred across devices. Users cannot transfer only parts of an image or document. The same applies for XDKinect [31] and CTAT [16], where only entire pictures can be distributed, as well as for Deep Shot [7] where users can only capture the whole screen and not a part of the information that is shown on the screen (e.g. information about a restaurant shown on a map).

A number of solutions allow parts of an image to be shown on multiple devices next to each other as illustrated in Figure 2. However, in these systems the image is still distributed as a whole. Users can choose to show a part of the image to another user when they are in proximity, but they cannot distribute parts of an image and get out of range again. This kind of interaction is, for example, provided in HuddleLamp [34] and GroupTogether [26] when two tablets are placed next to each other.

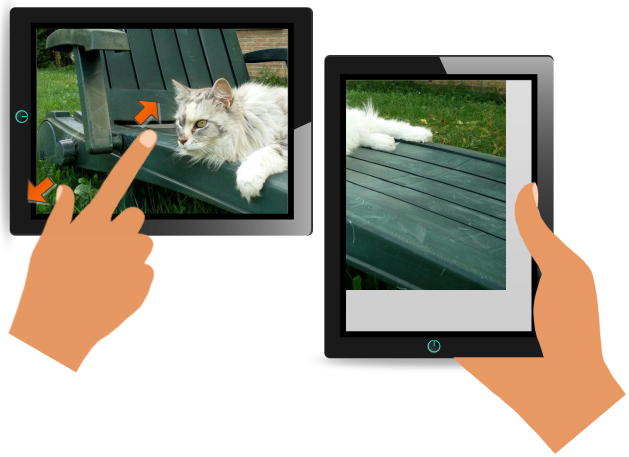

Fig. 2 Distributed image shown on multiple tablets

Other solutions offer a higher level of granularity, allowing parts of the data, such as paragraphs of a document, some pixels of an image or parts of an application, to be distributed. For instance, MultiMasher [22] and Proxywork [49] support the distribution of parts of a website, where arbitrary DOM-level elements of a website (e.g. the search bar, navigation panel or an image) can be distributed to different devices. Similarly, the web-based Panelrama [50] framework divides a user interface into different panels which are groupings of HTML user interface elements and enables their distribution across multiple devices. Documents can also be split into different parts to obtain a higher level of distribution granularity. This has been done in MultiSpace [14], which enables users to distribute entire documents as well as parts of the documents, such as sections of documents, images or text. Another example can be found in the framework proposed by Frosini et al. [15] for dynamic distribution of interactive user interface components. In their museum application text and images can be transferred to a separate device. Enabling an even finer granularity, the framework of Melchior et al. [29] supports the distribution of application widgets and arbitrary pixels on a screen.

The capturing of pixels on a screen can also be used to support easy state transfer. For example, if

an image is rotated on one device after distributing it, it will also be rotated on the destination device. By not only copying the raw pixels but also some meta-data, one can not only transfer the content but also the state of the distributed item or application as realised in Deep Shot [7]. State transfer is often achieved by other means. For example, when an application is relocated in ARIS [5], the application is closed on the source machine, invoked on the destination machine and the runtime state is passed from one machine to the other. State transfer can also be used to support synchronised views of the shared data on multiple devices. Panelrama [50], for example, keeps state information inside panels in order to keep them synchronised. A change in state information in one panel is processed by an event listener and might change the view of one or more panels on other distributed devices. Synchronised content can also be found in some example applications of Connichiwa [38], including a document viewer where highlighted paragraphs are synchronised between different devices. Frequently, such UI distribution is accomplished via a message passing mechanisms [1,3,4,18,23,26,30,38]. Moreover, certain systems developed their own software infrastructure for cross-device information sharing as seen with BEACH and iROS in the i-LAND [46] and iRoom [23] projects.

1.4 Existing DUI Classifications

While we propose a classification of DUIs in terms of interaction space, the supported granularity for distributed UI components and the distribution of state, Demeure et al. [11] proposed a reference framework to differentiate existing DUI approaches based on the four dimensions of computation, communication, coordination and configuration. Some of these dimensions are quite similar to ours whereas others are completely different. The *computation* of a DUI represents the elements that can be distributed. This might sound similar to our second granularity dimension, however Demeure et al. separate the elements in terms of UI layers rather than on granularity, including the presentation layer, logical presentation, control, functional code and its adapter. The second dimension, the *communication*, takes the distribution time into account. Since a distribution can be static or dynamic, systems can be divided into two categories: the ones allowing distribution at design time and the ones affording run-time distribution. The third dimension is the *coordination* which compares systems based on who is in charge of performing the distribution. The distribution can be user initiated, system initiated or mixed initiated. Finally, the last dimension, the *configuration*, takes into account where and how a UI is distributed. Is the same representation kept or adapted depending on the new location? Apart from the dimension addressed by Demeure et al., Elmqvist [13] briefly introduces some other dimensions (input, output, platform, space and time) based on his definition of the term *distributed user interface*. The space dimension is similar to our first dimension with a difference made between interfaces which are restricted to the same physical/geographic space and the ones that can be distributed to interactive remote spaces.

However, in the classification that we present, the space dimension subdivides systems into more categories, going from very local solutions to solutions that can be used anywhere. Similarly to the *computation* dimension of Demeure et al., we also focus on the elements a system can distribute, but we split the systems based on the granularity of their distribution. Our last dimension, the support for state transfer, is not used in any of the previously described classifications. Nevertheless, it is interesting to see that many of the existing DUI systems seem to support state transfer.

In order to make our classification more interactive, we created an online version[1] which shows the classification grid together with a time slider. The slider can be used to highlight the systems that have been developed during a certain timespan. A search box can further be used to find a particular research paper based on its title and information about the paper is then shown in an information panel next to the classification grid. By selecting a paper in the grid, information about the selected paper is shown in the same panel. While in this paper we present an extensive overview and classification of existing DUI approaches, we plan to add new as well as potentially missing solutions to our interactive online DUI classification over time. We further foresee to add the functionality that authors can individually add their papers to our online classification grid and thereby hope to establish a valuable resource for the DUI research community.

Based on the presented three dimensions, we defined the goal for the system that is currently under development. The aim is to be in the upper-right corner of the grid, hence having a system with no restriction in terms of interaction space and a fine granularity of distribution. We further aim to support state transfer during the distribution. Note that there are already some systems in that corner of our classification. However, these solutions do normally not support end-user DUI development but focus on designers or developers. We aim to enable the end-user development of DUIs as further discussed in the remainder of this paper.

[1] `http://dui.wise.vub.ac.be/classification/`

1.5 Development of DUIs

Some of the previously described systems focus on interaction techniques [1,7,25,34,35] or the collaborative aspect [3–6,14,26], others on the portability [38] and a third category focusses on making it easier for designers and developers to create DUI applications [2,8,15,29–31,38,50]. However, almost none of the existing solutions deal with making the frameworks available to end users without any programming skills. Some systems such as Weave [8] provide "easy-to-use" scripting languages in order to build DUIs or to ease the distribution across different devices. WebSplitter [19] provides users with an XML file specifying the distribution of UI elements across devices. Going a step further, XDStudio [30] offers a web-based authoring environment for designers with only basic web development experience. Finally, Husmann et al. [22] presented MultiMasher, a tool for technical as well as non-technical users. Nevertheless, MultiMasher is limited to the distribution of website components and users cannot distribute their own applications and data (e.g documents, pictures and files). While these systems make a step into the right direction they 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.

1.6 Discussion

We have presented a rich body of work in the domain of distributed user interfaces. While some projects focus on the interaction techniques used in performing cross-device interactions, others focus on the elements that can be distributed among devices. A third group of solutions pay special attention to the environment in which the distribution takes place.

Authors focussing on the cross-device *interactions* all propose new interaction techniques to manipulate and share data across devices. However, each application provides their own set of interactions, often without taking into account existing interactions introduced by other systems. This results in inconsistent interactions for similar actions across different cross-device applications. For example, in one system the sharing of a UI component is achieved by tilting the device, while in another solution the user has to swipe the UI component to the target device. In addition, different modalities such as speech or gestures are sometimes used to transfer content. Further, in order to be able to share data the devices need to be connected to each other, entailing different kinds of interactions such as bumping or stitching as well as other technologies including QR codes, NFC tags, Bluetooth or WiFi.

A second group of research investigates the *elements* that can be distributed across devices. In some cases, different UI components can be distributed, while in other approaches the UI can only be distributed as a whole, restricting the granularity of the distribution. The distribution of UI components is often limited to specific devices. For example, in some of the presented solutions, data can only be shared between Android devices.

Finally, a number of the presented solutions only work in a specific *environment*. These systems are restricted in terms of space due to their fixed setup using static cameras, projectors, infrared as well as other sensors. Various approaches rely on a central server in order to achieve cross-device distribution where users are no longer bound to a fixed location. However, these solutions still depend on a connection to a central server. A last group of systems does neither need a fixed setup nor a central server. These solutions often rely on peer-to-peer connectivity and therefore are not limited in terms of interaction space.

To conclude, in an ideal DUI solution users should be able to distribute any UI element as well as an entire user interface to any other UI-compatible device by using well-defined interactions and without the need for a fixed setup. By all means such a system should also be user friendly. The following section describes our approach towards a flexible solution for distributed user interfaces addressing some of the discussed issues.

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 allows end users to customise existing interfaces as well as define their own distributed user interfaces via a dedicated end-user authoring tool. In order to develop such a rapid prototyping framework, we are currently investigating a conceptual model with 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.*

In the remainder of this section we discuss a DUI scenario to highlight some of the issues to be addressed in a DUI framework for end users. We then present our vision and some initial ideas for an end-user DUI framework. Finally, we discuss some requirements for end-user authoring tools.

2.1 Scrapbook Scenario

Consider Chloe, a young PhD student who just got back from her holidays in Japan. During her trip, Chloe took a lot of pictures with her digital camera as well as with her smartphone. She even recorded some sound clips with her smartphone in order to capture the Asian atmosphere. Now that she is back home, she wants to transfer the pictures to her laptop. Since Chloe often transfers information from her devices to her computer, she defined a swipe gesture specifically for that purpose. In order to be able to regroup the pictures on her laptop, she swipes the folder containing the pictures on her camera into the direction of her computer screen and she does the same for the pictures on her smartphone. After the pictures have been transferred, Chloe starts sorting them on her laptop. A few minutes later, she gets interrupted by a call from her parents, asking her to come over for dinner. It is a long way from Chloe's flat to her parents' house and she wants to continue with the sorting of her pictures. Hence, she quickly takes her tablet and performs a swipe gesture from the corresponding folder on her computer screen to move the holiday pictures to her tablet and continues with the deletion and renaming of pictures on the bus ride to her parents. After dinner at Chloe's parents, her parents would already like to see some of her holiday pictures. Therefore, Chloe swipes her pictures from her tablet to her parents' TV screen and opens the remote controller application she created. As the parents would also like to control the stream of pictures, they ask Chloe to share her application on their smartphones. Chloe quickly explains how to use the remote controller and during the instructions, her father repositions some interface components in order to make it easier for him to use the remote controller application.

The next morning, Chloe goes early to her workplace. Having some free time, she wants to make a scrapbook with the pictures that she sorted the day before. Chloe often creates a scrapbook from her holidays and always looses some time transferring the pictures and importing them in her scrapbooking software. Now she decides to create an interface recognising a specific gesture to copy the pictures to her desktop computer and automatically import and open them in her scrapbooking program. She defines that a *double swipe* gesture should be used for this functionality. Using her newly defined gesture, the transfer of the pictures to her desktop computer, the opening of her scrapbooking software as well as the import of the pictures is done automatically. Finally, Chloe decides to add some audio fragments to her scrapbook. Therefore, she takes her smartphone, opens the audio file to select a small fragment and uses a hold and tilt gesture to transfer the fragment to her scrapbooking program.

In this scenario, we presented an example of how our approach could be used by end users in their daily lives. Since different end users do not always expect the same behaviour from their smart devices and want similar actions to be performed in slightly different ways [20], we aim for enabling users to define their own interface, gestures and the actions triggered by these gestures. Chloe defined four interfaces, each of them recognising a different gesture in the described scenario. The *swipe* gesture is used to transfer folders, the *double swipe* to transfer pictures and open them in the scrapbooking software, the *hold and tilt* gesture transfers a specific audio fragment to the currently opened software and the *tap* gesture is used in the remote controller interface to browse pictures on a secondary screen. In our scenario, the latter has also been shared with other users. These users might want to adapt the user interface or change some of the interactions, as we illustrated with Chloe's father. Furthermore, interfaces can run on multiple devices. Note that Chloe's swipe interface to transfer pictures and folders is shared across all

her devices. In the described scenario Chloe used only touch gestures to interact with her devices. However, our approach is not limited to touch gesture interaction but allows for alternative interaction styles such as speech commands, gaze interaction or mid-air gestures.

Based on the described scenario and the envisioned interactions, we derived the requirements R1–R5 for an end-user DUI framework:

R1: A user must be able to add new functionality *Users should be able to add new user interface components, gestures and actions through a simple plug-in mechanism.*

R2: The creation of a user interface must be simple *Users should be able to create their own user interface based on a set of pre-defined user interface components, gestures and actions.*

R3: A user must be able to re-use and combine different user interfaces *Users should be able to re-use and combine user interfaces, user interface components, gestures and actions in order to create their customised interfaces, gestures or actions.*

R4: A user must be able to reconfigure existing user interfaces *Users should be able to use external user interfaces and reconfigure them accordingly.*

R5: A user must be able to share their user interfaces *Users should be able to share their user interfaces with other users.*

2.2 DUI Model

In order to facilitate the development of an end-user authoring tool and the creation of DUI applications, we are currently investigating a conceptual model by basing ourselves on existing related work. A number of authors presented models for cross-device interactions. For example, Nebeling et al. [32] introduced a model including the concepts of user, device, data, private session and session. These concepts are tracked by a multi-user/multi-device platform and the relationships between concepts are timestamped while the actions that can be performed on the data are hold in an action attribute. An action can be the adding, editing, copying, deleting, moving and logging of data. Another platform model which is more centred around the concrete distribution of UIs has been introduced by Melchior [28]. In his model, a platform can consist of other platforms. For example, a laptop has three platforms, the laptop itself, the screen and the keyboard. Furthermore, a platform has the three main component categories of connection, hardware and audio/video. The connection represents the input and output connections (e.g. Bluetooth, WiFi) available on the platform. The hardware defines the hardware components of the platforms, such as the battery, CPU and memory. Finally, the audio/video category holds the components linked to the media (e.g. camera, speaker or microphone). Most existing models are designed for a specific platform or system introduced by the authors. The models are neither made nor used to describe other systems. Therefore, Tesoriero [48] presented a metamodel to describe the capabilities and states of DUI systems. Two model editors are presented which can be used to build and modify UI distribution models. Tesoriero analysed five different user interfaces using the editors, which showcased the distribution characteristics of each UI and its components. All these models focus on the distribution across devices. However, there is a lack of concepts such as the re-usability of UI components, different classes of users as well as the sharing of DUI configurations between users. We are currently developing a user-centric cross-device interaction model addressing some of these issues.

A promising approach that we are currently investigating for modelling loosely coupled interactions between user interface components and various actions is based on the work of Signer and Norrie [42]. They introduced the resource-selector-link (RSL) hypermedia metamodel and the corresponding iServer [41, 45] implementation for linking arbitrary digital and physical entities via a resource plug-in mechanism. In our context, resources can be seen as different user interfaces and user interface components which can be linked together based on the `Link` concept illustrated in Figure 3. We often do not want to link an entire resource but only specific parts of a resource, such as a part of an audio file as shown in Chloe's case in our scenario. The concept of an RSL selector allows us to address parts of a specific resource and enables a finer level of distribution granularity. Note that it is out of the scope of this paper to describe all the RSL components but a detailed description of the RSL hypermedia metamodel can be found in [42]. An important RSL concept for realising our goal of DUI state transfer and the execution of third-party application logic (requirement R1) is the concept of so-called active components [43, 44]. 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 triggered. 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 as discussed in [43]. 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 (requirement R2) from a growing set of active components.

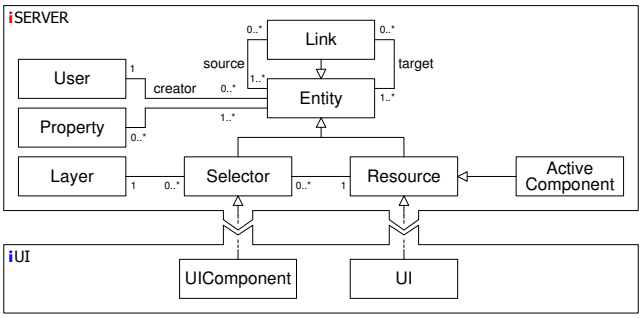

Fig. 3 Approach based on the RSL hypermedia metamodel

Let us illustrate the idea of RSL-based DUI development via a simple example from the previously introduced scrapbook scenario. In the last part of the scenario, Chloe wants her pictures to be copied to her computer and automatically imported into her scrapbooking software with a single gesture. She therefore defined this functionality by linking multiple active components (ACs) as illustrated in Figure 4. The idea is to have a swipe area which can be linked to different active components. In our example, Chloe has linked the `Gesture` active component to the swipe area. The `Gesture` active component can recognise different gestures such as a swipe, double swipe or tap. Since Chloe wants to use the *double swipe* gesture to copy her pictures, she links the `Gesture` active component to the `Data Transfer` component. She further specifies the type of gesture for which the `Data Transfer` active component should be triggered via the "double swipe" value of the `ac:gesture` link property. In the `Data Transfer` active component she defines her desktop computer as the receiving device. Consequently, all interaction in the swipe area will be forwarded to the `Gesture` active component which will trigger the `Data Transfer` active component once a double swipe gesture is recognised. Note that the corresponding parameter—in this case the type of the gesture (double swipe)—is stored as a property of the RSL link instance. In order to trigger the opening of the scrapbooking software with the transferred data, Chloe linked the `Data Transfer` active component to the `Application Launcher` active component which has been configured to open the corresponding application. This simple example illustrates the flexibility of the proposed active component-based approach. The `Gesture` active component might be linked to multiple active components in order to enable new functionality (requirement R1). For example, by reusing the `Gesture` active component, Chloe could define a new action that saves and closes the scrapbooking software on her desktop computer when a *tap* gesture is performed in the swipe area (requirement R3). In addition, multiple active components might be defined as the target of an RSL link in order to trigger multiple actions.

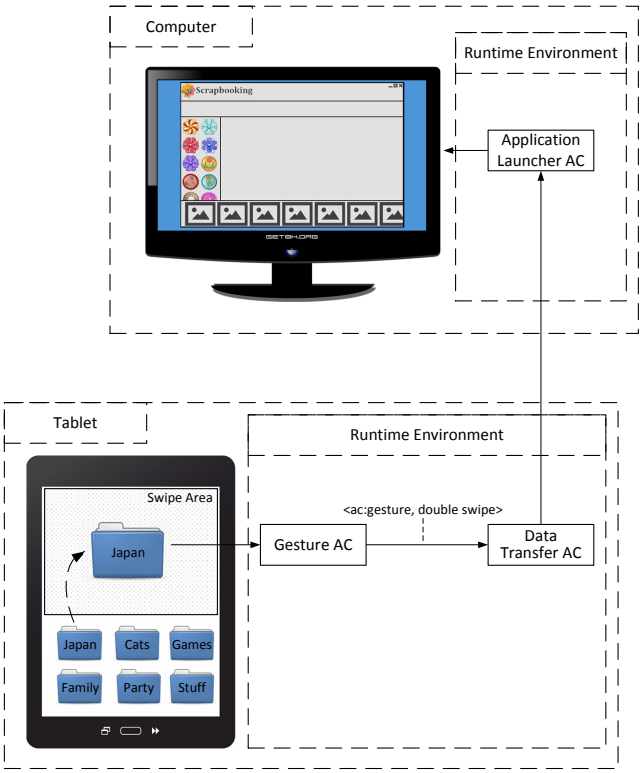

Fig. 4 Graphical representation of the proposed approach for the Scrapbook scenario

We further plan to address a number of other issues such as how to clearly separate 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.

2.3 End-User-Oriented DUI Framework

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 [1,3,4,7,8,18,23,26,30,38]. Since we aim for a portable solution that can be used at any location without prior installation, we consider using JavaScript and other web technologies to support the distribution across devices as seen in other DUI systems [7,8,22,30,34,38]. Furthermore, the Web has great potential for the development and use of future DUI systems in daily life [39].

While the proposed cross-device interaction model is based on some of the concepts introduced by the RSL hypermedia 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) (requirement R4). The latter is encapsulated in the `Developer Registry` component shown in the general architecture overview in Figure 5. The `Active Components` subcomponent stores all active components that have been implemented by developers while the `Resource/Selector Plug-ins` subcomponent 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 (requirement R5). 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 cross-device configurations to the `Configuration Pool`. Note that 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 (requirement R4). While in most cases users will adapt existing solutions based on their individual preferences, it might also be interesting to see whether some general interaction patterns will evolve over time.

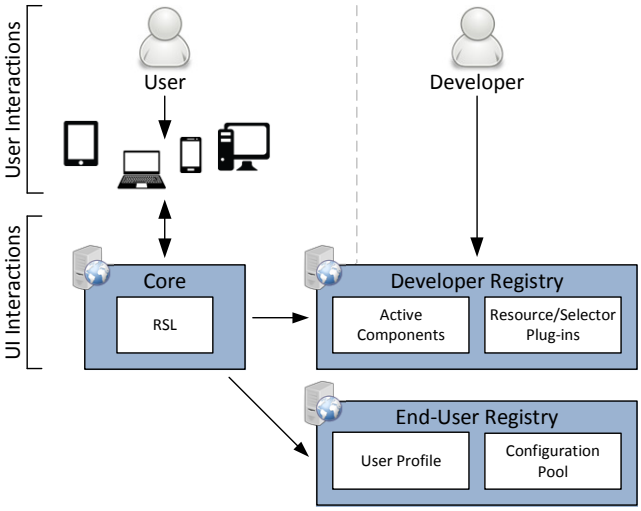

Fig. 5 Architecture for user-defined DUI interactions

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 have the possibility to adapt them or add new cross-device interactions if necessary. By providing end users 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 that perfectly suit everyone. 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 should be addressed and 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 proposed model, 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 are therefore investigating how the proposed model has to be extended in order to group multiple components together and share them as a whole. Since the RSL hypermedia metamodel offers the concept of structural links for grouping multiple entities, we plan to further investigate whether and how

structural links can be used for defining more complex cross-device interactions (requirement R3).

2.4 End-User Authoring Tool

In order to enable end users to define their own customisable interactions, there is a need for an end user authoring tool. We focus on two user groups, end users as well as non-expert developers. Therefore, the tool needs to be straightforward, error-prone and easy to use without any necessary programming skills (requirement R2). Furthermore, it must be extensible so that new components and functionality can be added via a plug-in mechanism over time (requirement R1). In order to increase the learnability and usability of the tool, the best metaphor for conveying information to the user has to be examined. A number of different metaphors for end-user authoring tools have been investigated. Recently, most of them have been applied in the domain of smart home and Internet of Things. Danado and Paternò's Puzzle framework [10] uses jigsaw pieces to provide end users the possibility to create, modify and execute mobile applications as well as the support for interaction with smart things, phone functionality and web services. Jigsaw pieces have also been used by Humble et al. [21] to allow users to reconfigure their ubiquitous domestic environments via a tablet editor. Some systems focus on a specific domain and provide a graphical editor tool with widgets depending on this domain. For example, the end user development environment of Ghiani et al. [17] enables users to customise the functionality and user interface of a multi-device museum guide. They provide a direct manipulation visual environment for editing the main features of the museum guide and for the creation of associated educational games. Likewise, the EUPHORIA system [27] enables end users to construct direct manipulation GUIs via a graphical tool palette. However in EUPHORIA, users can draw their own widgets with alternative representations. Nowadays, a popular visual tool is the *If This Then That* (IFTTT)[2] web application that allows users to create conditional statements to connect applications and Web Services to automate certain actions. For example, if it is midnight turn all the lights off. Such a rule-based authoring mode can also be found in Atooma[3], a free Android application, which allows end users to define more complex rules than in IFTTT. Zhang and Brügge [51] provide a rule-based interface to build smart home applications and introduce a second version of their tool based on the jigsaw metaphor. Another kind of end-user authoring tools are based on pseudo natural language, such as the SPOK system presented by Coutaz et al. [9] which is used for smart home environments. End-user authoring tools can also be found in mashup tools allowing Web Services to be linked to each other. Some of these tools including Microsoft PopFly and Yahoo Pipes use the workflow metaphor.

Further research will be necessary based on these existing authoring tools in order to find a suitable metaphor supporting the concepts introduced in our model and framework to allow end users as well as non-expert developers to create and share their customisable DUI applications. Moreover, we plan to investigate new forms of authoring which go beyond the graphical definition and composition of DUI interactions based on programming-by-example.

[2] https://ifttt.com

[3] http://www.atooma.com

3 Conclusions

A detailed analysis and classification of existing distributed user interface solutions has been presented. The proposed classification helps comparing existing DUI approaches in terms of the supported granularity for distributed UI components, differences regarding the supported interaction space as well as their support for the distribution of state. Given that a major contribution of this paper is the discussion and analysis of existing DUI research, we designed an interactive online version of the proposed DUI classification scheme which will be continuously extended and updated as new DUI solutions appear. We further proposed some ideas for the end-user development of distributed user interfaces based on a hypermedia approach. Thereby, UI components can be linked to arbitrary application logic at any level of granularity based on the concept of active components. Finally, we have introduced an architecture for the sharing of user-defined DUI components, enabling an evolutionary crowd-based DUI development and discussed some innovative ideas for the authoring of distributed user interfaces.

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, International Conference on Advanced Visual Interfaces, pp. 139–146. Rome, Italy (2010). DOI 10.1145/1842993.1843019

2. Balme, L., Demeure, A., Barralon, N., Coutaz, J., Calvary, G.: CAMELEON-RT: A Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces. In: Proceedings of EUSAI 2004, Symposium on Ambient Intelligence, pp. 291–302. Eindhoven, The Netherlands (2004). DOI 10.1007/978-3-540-30473-9_28
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, Conference on Human Factors in Computing Systems, pp. 2845–2854. Austin, USA (2012). DOI 10.1145/2207676.2208689
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, Symposium on Engineering Interactive Computing Systems, pp. 269–274. Copenhagen, Denmark (2012). DOI 10.1145/2305484.2305529
5. Biehl, J.T., Bailey, B.P.: ARIS: An Interface for Application Relocation in an Interactive Space. In: Proceedings of GI 2004, Conference on Graphics Interface, pp. 107–116. London, Canada (2004). DOI 10.20380/GI2004.14
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, Conference on Human Factors in Computing Systems, pp. 939–948. Florence, Italy (2008). DOI 10.1145/1357054.1357200
7. Chang, T., Li, Y.: Deep Shot: A Framework for Migrating Tasks Across Devices Using Mobile Phone Cameras. In: Proceedings of CHI 2011, Conference on Human Factors in Computing Systems, pp. 2163–2172. Vancouver, Canada (2011). DOI 10.1145/1978942.1979257
8. Chi, P.P., Li, Y.: Weave: Scripting Cross-Device Wearable Interaction. In: Proceedings of CHI 2015, Conference on Human Factors in Computing Systems, pp. 3923–3932. Seoul, Republic of Korea (2015). DOI 10.1145/2702123.2702451
9. Coutaz, J., Caffiau, S., Demeure, A., Crowley, J.L.: Early Lessons From The Development of SPOK, an End-User Development Environment for Smart Homes. In: Proceedings of UbiComp 2014, International Joint Conference on Pervasive and Ubiquitous Computing, pp. 895–902. Seattle, USA (2014). DOI 10.1145/2638728.2641559
10. Danado, J., Paternò, F.: Puzzle: A Mobile Application Development Environment Using a Jigsaw Metaphor. Journal of Visual Languages and Computing **25**(4), 297–315 (2014). DOI 10.1016/j.jvlc.2014.03.005
11. 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, Conference on Autonomic and Autonomous Systems, pp. 61–69. Gosier, Guadeloupe (2008). DOI 10.1109/ICAS.2008.34
12. Dietz, P.H., Leigh, D.: DiamondTouch: A Multi-User Touch Technology. In: Proceedings of UIST 2001, Symposium on User Interface Software and Technology, pp. 219–226. Orlando, USA (2001). DOI 10.1145/502348.502389
13. Elmqvist, N.: Distributed User Interfaces: State of the Art. In: Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem, pp. 1–12 (2011). DOI 10.1007/978-1-4471-2271-5_1
14. 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, Workshop on Horizontal Interactive Human-Computer Systems, pp. 27–34. Adelaide, Australia (2006). DOI 10.1109/TABLETOP.2006.23
15. Frosini, L., Manca, M., Paternò, F.: A Framework for the Development of Distributed Interactive Applications. In: Proceedings of EICS 2013, Symposium on Engineering Interactive Computing Systems, pp. 249–254. London, UK (2013). DOI 10.1145/2480296.2480328
16. Geronimo, L.D., Husmann, M., Patel, A., Tuerk, C., Norrie, M.C.: CTAT: Tilt-and-Tap Across Devices. In: Proceedings of ICWE 2016, International Conference on Web Engineering, pp. 96–113. Lugano, Switzerland (2016). DOI 10.1007/978-3-319-38791-8_6
17. Ghiani, G., Paternò, F., Spano, L.D.: Cicero Designer: An Environment for End-User Development of Multi-Device Museum Guides. In: Proceedings of IS-EUD, Symposium on End-User Development, pp. 265–274. Siegen, Germany (2009). DOI 10.1007/978-3-642-00427-8_15
18. Hamilton, P., Wigdor, D.J.: Conductor: Enabling and Understanding Cross-Device Interaction. In: Proceedings of CHI 2014, Conference on Human Factors in Computing Systems, pp. 2773–2782. Toronto, Canada (2014). DOI 10.1145/2556288.2557170
19. Han, R., Perret, V., Naghshineh, M.: WebSplitter: A Unified XML Framework for Multi-Device Collaborative Web Browsing. In: Proceedings of CSCW 2000, Conference on Computer Supported Cooperative Work, pp. 221–230. Philadelphia, USA (2000). DOI 10.1145/358916.358993
20. Holloway, S., Julien, C.: The Case for End-User Programming of Ubiquitous Computing Environments. In: Proceedings of FoSER 2010, Workshop on Future of Software Engineering Research, pp. 167–172. Santa Fe, USA (2010). DOI 10.1145/1882362.1882398
21. Humble, J., Crabtree, A., Hemmings, T., Åkesson, K., Koleva, B., Rodden, T., Hansson, P.: "Playing with the Bits": User-Configuration of Ubiquitous Domestic Environments. In: Proceedings of UbiComp 2003, Conference on Ubiquitous Computing, pp. 256–263. Seattle, USA (2003). DOI 10.1007/978-3-540-39653-6_20
22. Husmann, M., Nebeling, M., Pongelli, S., Norrie, M.C.: MultiMasher: Providing Architectural Support and Visual Tools for Multi-device Mashups. In: Proceedings of WISE 2014, International Conference on Web Information Systems Engineering, pp. 199–214. Thessaloniki, Greece (2014). DOI 10.1007/978-3-319-11746-1_15
23. Johanson, B., Fox, A., Winograd, T.: The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. IEEE Pervasive Computing **1**(2), 67–74 (2002). DOI 10.1109/MPRV.2002.1012339
24. Johanson, B., Hutchins, G., Winograd, T., Stone, M.C.: PointRight: Experience with Flexible Input Redirection in Interactive Workspaces. In: Proceedings of UIST 2002, Symposium on User Interface Software and Technology, pp. 227–234. Paris, France (2002). DOI 10.1145/571985.572019
25. 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, International Conference on Tangible, Embedded, and Embodied Interaction, pp. 89–96. Stanford, USA (2015). DOI 10.1145/2677199.2680584
26. Marquardt, N., Hinckley, K., Greenberg, S.: Cross-Device Interaction via Micro-mobility and F-formations. In: Pro-

ceedings of UIST 2012, Symposium on User Interface Software and Technology, pp. 13–22. Cambridge, USA (2012). DOI 10.1145/2380116.2380121
27. McCartney, P., Goldman, K.J., Saff, D.E.: EUPHORIA: End-User Construction of Direct Manipulation User Interfaces for Distributed Applications. Software-Concepts and Tools **16**(4), 147–159 (1995). DOI 10.7936/K7PZ572S
28. Melchior, J.: Distributed User Interfaces in Space and Time. In: Proceedings of EICS 2011, Symposium on Engineering Interactive Computing System, pp. 311–314. Pisa, Italy (2011). DOI 10.1145/1996461.1996544
29. 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, Symposium on Engineering Interactive Computing System, pp. 69–78. Pittsburgh, USA (2009). DOI 10.1145/1570433.1570449
30. Nebeling, M., Mintsi, T., Husmann, M., Norrie, M.C.: Interactive Development of Cross-Device User Interfaces. In: Proceedings of CHI 2014, Conference on Human Factors in Computing Systems, pp. 2793–2802. Toronto, Canada (2014). DOI 10.1145/2556288.2556980
31. Nebeling, M., Teunissen, E., Husmann, M., Norrie, M.C.: XDKinect: Development Framework for Cross-Device Interaction Using Kinect. In: Proceedings of EICS 2014, Symposium on Engineering Interactive Computing Systems, pp. 65–74. Rome, Italy (2014). DOI 10.1145/2607023.2607024
32. Nebeling, M., Zimmerli, C., Husmann, M., Simmen, D.E., Norrie, M.C.: Information Concepts for Cross-Device Applications. In: Proceedings of DUI 2013, Workshop on Distributed User Interfaces: Models, Methods and Tools, pp. 14–17. London, UK (2013)
33. Paternò, F., Santoro, C.: A Logical Framework for Multi-Device User Interfaces. In: Proceedings of EICS 2012, Symposium on Engineering Interactive Computing Systems, pp. 45–50. Copenhagen, Denmark (2012). DOI 10.1145/2305484.2305494
34. 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, International Conference on Interactive Tabletops and Surfaces, pp. 45–54. Dresden, Germany (2014). DOI 10.1145/2669485.2669500
35. Rekimoto, J.: Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In: Proceedings of UIST 1997, Symposium on User Interface Software and Technology, pp. 31–39. Banff, Canada (1997). DOI 10.1145/263407.263505
36. Robertson, S.P., Wharton, C., Ashworth, C., Franzke, M.: Dual Device User Interface Design: PDAs and Interactive Television. In: Proceedings of CHI 1996, Conference on Human Factors in Computing Systems, pp. 79–86. Vancouver, Canada (1996). DOI 10.1145/238386.238408
37. Sanctorum, A., Signer, B.: Towards User-Defined Cross-Device Interaction. In: Proceedings of DUI 2016, Workshop on Distributed User Interfaces, pp. 179–187. Lugano, Switzerland (2016). DOI 10.1007/978-3-319-46963-8_17
38. Schreiner, M., Rädle, R., Jetter, H., Reiterer, H.: Connichiwa: A Framework for Cross-Device Web Applications. In: Extended Abstracts of CHI, Conference on Human Factors in Computing Systems, pp. 2163–2168. Seoul, Republic of Korea (2015). DOI 10.1145/2702613.2732909
39. Schreiner, M., Rädle, R., O'Hara, K., Reiterer, H.: Deployable Cross-Device Experiences: Proposing Additional Web Standards. In: Proceedings of Cross-Surface 2015, Workshop on Interacting with Multi-Device Ecologies in the Wild, pp. 17–20. Madeira, Portugal (2015). DOI 10.1145/2817721.2835067
40. Shen, C., Everitt, K., Ryall, K.: UbiTable: Impromptu Face-to-Face Collaboration on Horizontal Interactive Surfaces. In: Proceedings of UbiComp 2003, International Conference on Ubiquitous Computing, pp. 281–288. Seattle, USA (2003). DOI 10.1007/978-3-540-39653-6_22
41. Signer, B.: Fundamental Concepts for Interactive Paper and Cross-Media Information Spaces. Books on Demand GmbH (2008)
42. Signer, B., Norrie, M.C.: As We May Link: A General Metamodel for Hypermedia Systems. In: Proceedings of ER 2007, International Conference on Conceptual Modeling. Auckland, New Zealand (2007). DOI 10.1007/978-3-540-75563-0_25
43. 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, International Conference on Pervasive Computing and Applications. Alexandria, Egypt (2008). DOI 10.1109/ICPCA.2008.4783676
44. Signer, B., Norrie, M.C.: Active Components as a Method for Coupling Data and Services: A Database-driven Application Development Process. In: Proceedings of ICOODB 2009, International Conference on Object Databases. Zurich, Switzerland (2009). DOI 10.1007/978-3-642-14681-7_4
45. Signer, B., Norrie, M.C.: A Model and Architecture for Open Cross-Media Annotation and Link Services. Information Systems **36**(6), 538–550 (2011). DOI 10.1016/j.is.2010.08.002
46. 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: Proceeding of the CHI 1999, Conference on Human Factors in Computing Systems, pp. 120–127. Pittsburgh, USA (1999). DOI 10.1145/302979.303010
47. Tandler, P., Prante, T., Müller-Tomfelde, C., Streitz, N.A., Steinmetz, R.: Connectables: Dynamic Coupling of Displays for the Flexible Creation of Shared Workspaces. In: Proceedings of UIST 2001, Symposium on User Interface Software and Technology. Orlando, Florida (2001). DOI 10.1145/502348.502351
48. Tesoriero, R.: Distributing User Interfaces. In: Proceedings of DUI 2014, Workshop on Distributed User Interfaces and Multimodal Interaction, pp. 1–10. Toulouse, France (2014). DOI 10.1145/2677356.2677669
49. Villanueva, P.G., Tesoriero, R., Gallud, J.A.: Distributing Web Components in a Display Ecosystem Using Proxywork. In: Proceedings of HCI 2013, Conference on Human Computer Interaction, pp. 28:1–28:6. London, UK (2013)
50. Yang, J., Wigdor, D.: Panelrama: Enabling Easy Specification of Cross-Device Web Applications. In: Proceedings of CHI 2014, Conference on Human Factors in Computing Systems, pp. 2783–2792. Toronto, Canada (2014). DOI 10.1145/2556288.2557199
51. Zhang, T., Brügge, B.: Empowering the User to Build Smart Home Applications. In: Proceedings of ICOST 2004, International Conference on Smart Homes and Health Telematic, pp. 170–176. Singapore, Republic of Singapore (2004)