Vrije Universiteit Brussel

FACULTY OF SCIENCE AND BIO-ENGINEERING SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Fluid Cross-Media and Cross-Device Interaction in the Office of the Future

Master thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Audrey Sanctorum

Promoter: Prof. Dr. Beat Signer
Advisor: Sandra Trullemans

Academic year 2014-2015

Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN
VAKGROEP COMPUTERWETENSCHAPPEN

Fluid Cross-Media and Cross-Device Interaction in the Office of the Future

Masterproef ingediend in gedeeltelijke vervulling van de eisen voor het behalen van de graad Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Audrey Sanctorum

Promotor: Prof. Dr. Beat Signer
Begeleider: Sandra Trullemans

Academiejaar 2014-2015

Abstract

As the volume of information grows over time, new techniques, applications and devices are invented to cope with the management of all this information. New information management methods are needed in order to create, keep and organise large amounts of physical as well as digital information. Managing the digital information space has become particularly challenging since the rise of electronic devices and new online storage possibilities. Smartphones, tablets and touch tables that often allow information to be stored in the cloud, are becoming a natural part of our daily life. Using different devices to manage personal information results in fragmented information across devices, which led to research in the field of cross-device interaction. Furthermore, personal data is also spread across the digital and physical information space. This is due to the fact that a lot of people still use paper to manage their personal information. Consequently, the research in the field of cross-media interaction, which tries to combine the physical and digital information space, has been an active research area since the early nineties.

This thesis presents the eXtensible User Interface (XUI) framework, which contributes to both of these two fields of research. The objective of the XUI framework is to combine these fields to achieve a fluid cross-media and cross-device interaction environment. This is accomplished by analysing the related work in these two research domains. In addition, the potential of the XUI framework is illustrated by the construction of an Office of the Future environment. Our Office of the Future setup consists of multiple physical as well as digital user interfaces. The first interface is the Personal information management Visualisation (PimVis) interface, which runs on a multi-touch tabletop. We especially developed the PimVis interface to manage personal information using the expertise from related work in the domain of information visualisation. Next to the PimVis interface, two additional ubiquitous interfaces have been developed, namely the extended file system and the augmented ring binders application. The extended file system runs on a desktop computer. The augmented ring binders application consists of several ring binders, which have been augmented with integrated LEDs. The fluid cross-media and cross-device interaction between these three interfaces is achieved by using the XUI framework.

In order to assess our Office of the Future setting, we conducted a user study. To goal of this user study was to verify whether the fluid interaction between the different user interfaces of our setup grants a significant benefit compared to a setup without this interaction. By applying different statistical tests on the quantitative data, we could conclude that there was no significant

difference between the two setups. However, qualitative data was collected from observing the participants while performing their tasks and from a semi-structured interview that we conducted afterwards. Using this qualitative data, we present a few guidelines for developers of future fluid cross-media and cross-device interaction spaces.

Declaration of Originality

I hereby declare that this thesis was entirely my own work and that any additional sources of information have been duly cited. I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this thesis has not been submitted for a higher degree to any other University or Institution.

Acknowledgements

First of all, I would like to thank my promoter Prof. Dr. Beat Signer to give me the opportunity to write a thesis at the WISE lab and to provide feedback on my thesis and presentations. The next person I want to thank is Sandra Trullemans for her guidance and support during the year and during the writing of my thesis. I also want to thank my friends and fellow students for their support and help throughout the year. A particular note of thanks to Tim Reynaert, Ayrton Vercruysse, Tim Vereecken and Jasmien De Ridder, for their solidarity and presence at the WISE lab. Additionally, I want to thank my boyfriend, Guillaume Dekoninck for his encouragements during the year. Last but not least, I wish to express a special thanks for the support and love of my family. My mother for her tonic juices, which gave me a boost when needed. My father, for his advice and my grandfather for his interest in my research, which provided me with some interesting ideas. Finally, I would also like to thank my brother and my cat for clearing my mind when needed.

Contents

1 Introduction

Information has always played an important role in our society. Every piece of information is used in our daily life to gain knowledge and solve problems. Nowadays, we manage tons of physical as well as digital information sources. With the growing amount of these sources, the volume of information keeps getting bigger, which makes the efficient management of information more difficult. This increase of the amount of information is mainly because of the rise of the World Wide Web, which has made information more accessible and has led to an exponentially growing amount of information. The ability of retrieving, managing and organising information becomes more and more challenging for the human brain. The brain has some limitations, such as the storage and re-finding capacity. The latter is highly dependent on associations in the human brain, which can be easily lost when coping with large amounts of information. Forgetting things is just in the human nature [57]. Therefore, it is important to compensate this loss by storing information either physically or digitally. Since the fifteenth century, paper played an important role for storing, moving and placing information in some specific context [45]. A context represents the task on which a person is working on when handling the paper documents. This can, for example, be a *department meeting* or the *writing of a thesis*. Papers were often stored depending on the context in which they were used in order to facilitate its retrieval later on.

Nowadays, new technologies such as personal computers, smartphones, tablets, hard drives or the cloud can be used for managing information. Some of these technologies sometimes offer an almost unlimited amount of storage space. Most people are using these devices in combination with each other to complete certain tasks. During the completion of these tasks, people often need to consult some personal documents and maybe create new ones. These activities performed when working on a task, such as the keeping, organising and re-finding of documents, are the main subject in the field of Personal Information Management (PIM) [33].

While it is already known that these activities are often tough when dealing with a lot of information, it gets even more difficult when all this information is located on different devices. This leads to the need for *cross-device* interfaces and applications, which can facilitate the management of fragmented data across a user's devices. Furthermore, people still tend to use paper for a couple of these activities. As result, there is not only the need for fluid cross-device interaction, but also for fluid *cross-media* interaction. In other words, interaction between the physical and digital space.

1.1 Information Problems and Management

The continued increase of information results in the problem of *information overload*. Due to the overabundance of information, it is becoming more difficult to judge whether the information is truly relevant for a person's needs. Since the human memory can only handle a fixed amount of information at a certain time, it becomes harder to skip non-relevant parts. One could also miss some important parts because it is concealed by other information items of the same topic. Another problem that occurs as consequence of information overload is *information fragmentation*. This is because our personal information is now spread over various devices in the digital world, and different kinds of documents in the physical world, such as post-it notes, paper documents, books or pictures. Within this often immense personal information space, users have more and more difficulties to re-find their information. The Personal Information Management (PIM) research field deals with these problems and strives at having the right information at the right place [32]. In the physical world, people use different kinds of techniques to classify items, such as piling, filing and mixing to retrieve and find the right information.

To arrange papers on a desk, most people start by making piles of paper. Each pile could, for example, represent a certain task that needs to be done using the papers in this pile. This method of organising a desk is called

piling [38] and is illustrated in Figure 1.1. Each individual element of such a pile can be a piece of paper, a folder or another information carrier. The entire pile is presented as a group of individual elements by Malone [38].

Figure 1.1: On the left, piling method: a group of untitled and unordered elements. In the middle, filing method: a group of titled and ordered elements. On the right, mixing method: a group of titled and unordered elements.

Another method to organise documents is *filing*, which, for example, allows us to organise papers in a desk drawer, as shown in the picture in the centre of Figure 1.1. Again, elements can be any piece of information carrier. An entire file drawer represents one file group consisting of individual elements. The main difference between filing and piling is that file elements are labelled while pile elements do not necessarily need to be labelled. Next to that, elements of a filing organisation are ordered while pile elements are not. Moreover, file groups may be titled and/or ordered, while pile groups cannot be titled but may be ordered [38]. When combining both organisational methods, the *mixing* is obtained, which is neither just filing nor just piling. Mixing is often present in a work environment without knowing it [57]. The elements of a mixture can be labelled or not and may be ordered. An illustration of the mixing method can be found in the right picture of Figure 1.1.

Although these methods are used in the physical world, most of them are also used to manage digital information. The mixing strategy is, for example, often used in both the physical as well as the digital world. Digital piling and filing are also used as organisational methods in the digital world. According to a study by Trullemans and Signer, piling is applied to the same degree as the mixing strategy in both information spaces [58]. The physical and digital filing strategy is anyhow not applied in an excessive manner. However, when this method is applied, it highly facilitates information re-finding.

1.2 Problem Statement

Many distributed user interfaces exist in order to deal with data that is spread across different kinds of devices. They mainly focus on the interaction possibilities between these devices and how information can be shared among them. Since paper is still often used in our daily working environment, researchers also focus on combining the physical and digital space, thereby providing interaction between physical and digital items. This interaction aims at combining the affordances of paper and the advantages of digital documents. Both cross-device and cross-media interactions try to solve the many problems of personal information management that were explained in Section 1.1. However, to our knowledge, these two kind of interactions are not used in combination. The interaction amongst devices *and* between digital and physical information spaces has not yet been investigated in detail. We aim at combining these cross-device and cross-media interactions in a fluid interaction space.

1.3 Contributions

In this thesis we present the eXtensible User Interface (XUI) framework which contributes to the cross-device and cross-media interaction research fields. We investigated the existing research in both of these two fields in order to construct the XUI framework. The XUI framework aims at facilitating the management of digital as well as physical personal information, which is spread across devices and across the digital and physical information spaces. This has been achieved by combining both fields of research. Therefore, the XUI framework provides the possibility to build a fluid cross-device and cross-media interaction space.

In order to achieve this goal, the XUI framework has been built based on three requirements. The first one is *object-oriented user interfaces*, allowing properties to be given to documents and any other personal item. The second requirement is *lightweight data exchange*. Instead of passing the entire application state, only sending the needed information to the appropriate interface is a faster and easier way to communicate between interfaces. The last one is *ad-hoc interaction configuration* and allows the system to be flexible and extendible.

To illustrate the potential of fluid cross-device and cross-media interaction spaces, we present the Office of the Future setting. The setting contains different digital and physical interfaces which can fluidly interact with each other. The digital interfaces are PimVis and the extended file system interface, while the augmented ring binders interface is used as a physical interface.

We present the Personal information management Visualisation (PimVis) as the first example interface for our Office of the Future setup. The PimVis user interface has been created to work on a multi-touch tabletop. We designed the PimVis interface based on the OC2 framework [59] in order to follow an OOUI design approach. The goal of the PimVis interface is to make it easier for the users to collect, organise, find and visualise their personal information. PimVis has been built following the standard user interface design principles. Furthermore, it has been designed based on the background research we conducted in the field of information visualisation. Additionally, the PimVis interface provides a fluid interaction with the two other user interfaces of our Office of the Future setting. This interaction allows users to re-find their documents stored on their computer or in one of their ring binders by using a *re-find* button.

To complete the Office of the Future setting, two additional ubiquitous user interfaces have been developed, namely the extended file system and the ring binders application. The main purpose of these two user interfaces is to provide a fluid interaction between the different Office of the Future interfaces. When users want to find a digital document which is currently selected in PimVis in their file system, the extended file system interface will open the corresponding folder and select the document in the Windows File Explorer. The interface also allows users to open a document in the PimVis interface using a context menu in the Windows File Explorer on their computer. When users want to find a physical document open in PimVis in their ring binders, the augmented ring binders interface will indicate the right ring binder by lighting up the corresponding augmented ring binder's built-in LED. In addition, each ring binder contains augmented sheets of paper, which can be used to open the a paper document in the PimVis interface.

Finally, we conducted a user study in order to evaluate our Office of the Future setting. The goal of the study is to verify whether there was a significant difference in terms of user experience between the use of the setup with the integration of the fluid interaction between the user interfaces and without this interaction. Based on the quantitative results of the study, we could conclude that there was no significant difference between the two cases. However, the qualitative data collected by observing the behaviour of the par-

ticipants during the study and by the semi-structured interview conducted afterwards, provided us with valuable information about the setup. Last but not least, based on this collected qualitative data we present a few guidelines for developers of future fluid cross-media and cross-device interaction spaces.

1.4 Thesis Outline

The remainder of this thesis is structured as follows. First, related work in the field of cross-device and cross-media interaction is explored. For the cross-device interaction, interaction between tablets, smartphones, tabletops, smart boards and other electronic devices in different setups is reviewed. On the other hand, for cross-media interaction, the interaction possibilities between the physical and digital world are analysed.

Inspired by our investigation of related work, an Office of the Future setup is introduced that allows fluid cross-media and cross-device interaction. A couple of use cases are then presented in order to show the usefulness and potential of the setup.

In the next chapter, we discuss the PimVis interface which is a part of the Office of the Future environment. In order to design PimVis, additional background research in the field of information visualisation of document spaces has been done. Based on this background research, design requirements were composed and an iterative design process was started. Since PimVis has been created to be used on a multi-touch tabletop, it offers various touch interactions, which are explained in this chapter. Moreover, this chapter ends with an overview of the important parts of the final PimVis implementation.

In order to make such an office a reality, we present the eXtensible User Interface (XUI) framework, which is examined in the following chapter. First, the requirements of the XUI framework are established. Then, the architecture of the XUI framework is introduced together with all its components. In addition, important implementation specific features are also shown.

In Chapter 6, we explain how the Office of the Future setting has been evaluated. The experimental setup is reviewed together with the methodology that was used to let the participants assess the system. After that, the quantitative and qualitative results are analysed. Based on these results, a few guidelines and conclusions are presented.

Finally, we present some interesting future work possibilities, as well as some conclusions about the presented work.

2 Background

In this chapter existing work in the fields of cross-device and cross-media interaction will be discussed. The cross-device interaction field also leads to Distributed User Interfaces (DUI), which will also be reviewed. For the background research of the cross-media interaction, the focus has been set on paper-digital interaction. For both fields of research, many office settings are mainly analysed.

2.1 Cross-Device Interaction

With the emergence of all sorts of new interactive devices these last few years, a lot of research has been done in the field of cross-device interaction. This also led to increasing research in the field of Distributed User Interfaces (DUI). An excellent example of such a DUI can be found in Frosini et al.'s work [20]. The authors presented a framework for distributed and/or migratable user interfaces in multi-device and multi-user environments. Based on this framework, they introduce a working application that runs on mobile devices and has been used to enhance visits inside a museum. Their application allows visitors to get information via QR codes on their smartphone, which can then be filtered, showed and shared by displaying it on large screens that the visitors encounter during their visit. The state as well as the content of the application can be shared across devices.

Melchior et al. [40] went a step further by presenting a more decoupled framework. Besides multi-device and multi-user interaction, the framework also supports multiple operating systems, multi-display and concurrent interaction. By multi-display interaction, the authors mean that any user interface component can, not only be distributed among devices, but also among various user interfaces on the same device. However, these user interfaces still need to be located on different screens. This allows concurrent multi-tasking. An example of multi-tasking is presented by the authors in their first use case, where multiple users worked at the same time with certain office applications. Components of the applications could then be shared and modified by multiple users. This migration allows users to work together on the same application from different places at the same time. In the second use case, the authors show a setup to play pictionary with multiple devices, a PDA, laptop or desktop and a projector or large screen. As soon as every device runs the application the players can start playing. Certain components or widgets are distributed amongst the different user interfaces while others are not. The timer, for example, can be seen on each interface, while the label displaying which *word* the player has to draw, is only visible on the player's device, namely the PDA.

Figure 2.1: Deep Shot

While Melchior et al. [40] presented their work for any device, Chang and Li [10] primarily focussed on the interaction between a smartphone and a computer. They introduce Deep Shot, a framework to migrate tasks by using a smartphone's camera. The authors envisioned four scenarios that can be accomplished with their technology. The first one is *taking information to go*, which is done by migrating information from a PC to a mobile phone.

This could be used to capture a map on the screen of the computer, which can then be carried on the user's smartphone for later look up, as shown in Figure 2.1. The second scenario is *viewing or saving mobile phone content on PCs*, which works just as the previous scenario but now the other way around, information is carried from the smartphone to the computer screen. This can, for example, be useful to transfer a photo from the user's smartphone to the computer by simply aiming the camera at the PC screen. Just like the widely used USB flash drive, Deep Shot can be used to transfer information between computers. It even goes further by also allowing users to migrate application states. This led the authors to the third scenario which is, *using mobile phones as a bridge between PCs*. Finally, the last scenario is *sharing content between mobile phones*. When a user needs the contact information displayed on a friend's mobile phone, the user can simply take a picture of it and information will be automatically transferred. Via a usability analysis and a technical evaluation, the authors proved that their system was reliable and feasible for day-to-day tasks using a simple and natural gesture.

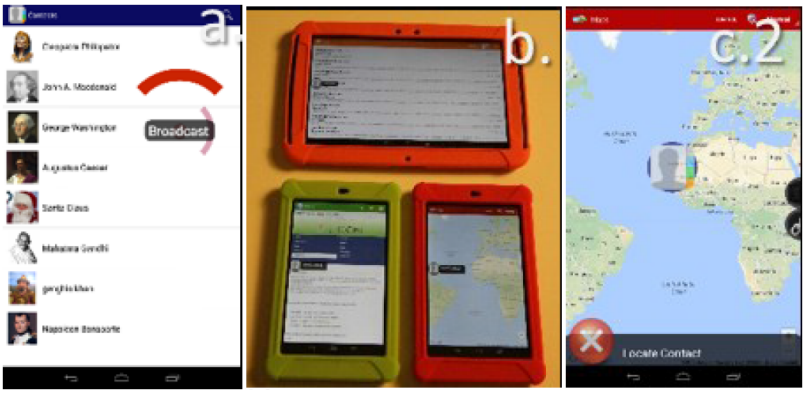

Figure 2.2: Conductor at use: (a) the user broadcasts a cue by choosing *broadcast* from a context menu; (b) all devices received the cue and displays it; (c.2) the user drag the cue to locate the contact in the current application.

While Chang and Li [10] are limited on the feature matching algorithm used to transfer information between the PC and the mobile phone, Hamilton and Wigdor [23] use another system to share information. They designed Conductor, a prototype framework to build cross-device applications. The communication between devices is done through a broadcasting system. When a user decides to share a piece of information across multiple devices, the *broadcast* option of the Conductor's context menu needs to be selected. Once this is done, every device will receive a cue containing the shared in-

formation. The user can then choose which device needs this information by tapping on the cue on the appropriate device. If a location has been broadcasted, dragging the cue to, for example, the Google maps application of the device will pinpoint the location on the map as shown in Figure 2.2. When a cue is left untouched for a certain amount of time or when the user touches another part of the screen, the cue will be hidden in a left panel of the screen and can be retrieved later on. Additionally, the Conductor system provides a task manager, which allows users to visualise the distributed space. The task manager shows a list of all devices running Conductor together with an up-to-date screenshot of each of the devices' screens. Furthermore, it allows to clone a session or application state between devices. Devices can be added to the interactive group of devices through different mechanisms. It can be added via a QR code, an NFC communication channel, the bumping with another device or via proxies containing the necessary identification information.

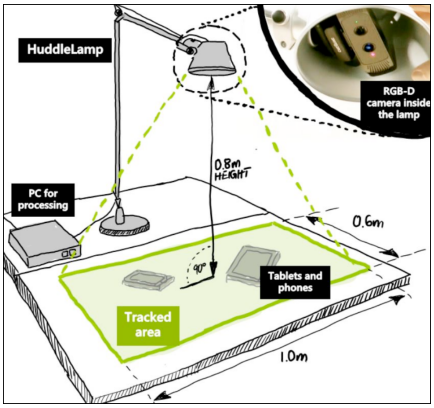

Figure 2.3: HuddleLamp setup

Yet another way to achieve cross-device communication has been presented by Rädle et al. [43] with the HuddleLamp system. This setup uses a lamp with an integrated RGB-D camera that can track hand movements and the position of any mobile device placed on the table. Figure 2.3 illustrates the technical setup of this lamp. Besides the RGB-D camera images to identify the mobile displays, the camera depth values can be used to detect the low IR reflectance from the screens and thus more easily recognise them. To verify the results, avoid false positives and distinguish the different screens, virtual fiducial markers on the display are used. Rädle et al. focus on the collaborative aspect of cross-device interaction in particular. Their setup allows different users to share information on their devices as soon as they are placed on the table without any additional configuration. The shared vir-

tual workspace also takes into account the position or spatial arrangement of all devices, allowing users to benefit from spatially-aware menus and modes based on the screen orientation or the distance between the screens. Additionally, intuitive multi-touch gestures can be used to communicate with different devices, like the drag, pinch-to-zoom-and-rotate and flick gestures. The flick movement is, for example, used for cross-device information transfer.

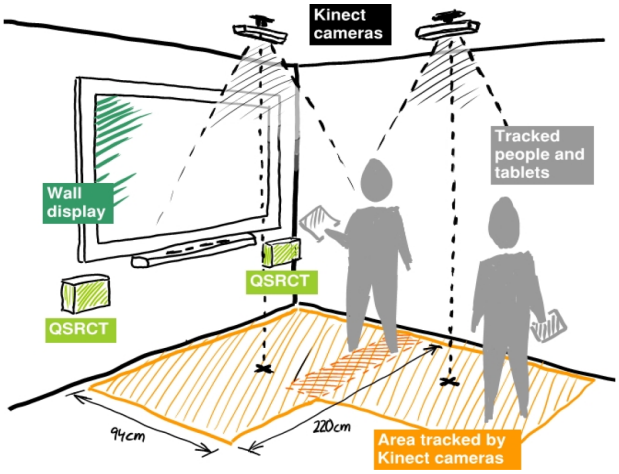

Figure 2.4: GroupTogether setup

Other authors, like Marquardt et al. [39] and Everitt et al. [15] also put more focus on user collaboration. After a design study to gain insights about how people interact and position each other while performing cooperative and competitive tasks, Marquardt et al. introduced GroupTogether. GroupTogether is a sensing system that analyses cross-device interaction based on the position and formation of people when interacting as a group (this study is called the F-formations) and the way they orient and tilt their devices towards each other to share information (which is called micro-mobility [37]). Inspired by the design study, specific interaction techniques have been implemented to share digital information across devices, like the tilt-to-preview, dragging and pinch-to-zoom interaction. Using a pair of overhead Kinect depth cameras, QSRCT radio modules and accelerometers, the authors setup a workspace for users to evaluate the different interaction techniques. Users could interact via their tablets and the wall display. Figure 2.4 schematically shows this setup.

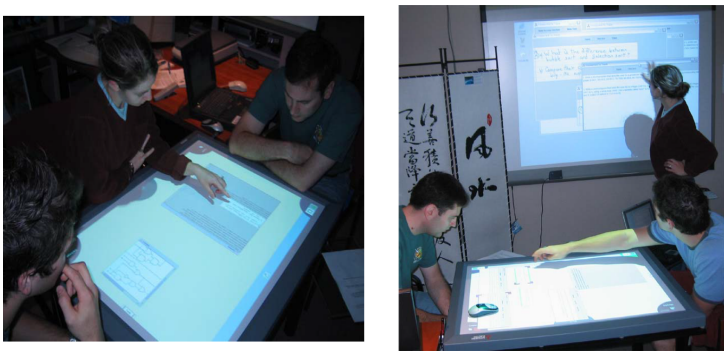

Figure 2.5: MultiSpace setup. Participants transition from tabletop interaction (left) to wall-and-table interaction (right).

While in Marquardt et al.'s [15] study all participants are standing and using a tablet, Everitt et al. present MultiSpace, a space where participants sit around a touch-sensitive tabletop, called DiamondTouch [12], while also having a touch-sensitive wall display at their disposal, which is linked over a wired network to the table. Apart from these devices, laptops and tablets can also be used by linking them to the table via Java Sockets (wireless LAN). This setup is illustrated in Figure 2.5. The authors focus on the role of the table as a central hub that can be used in meeting rooms to support collaboration across multiple devices. MultiSpace supports micro-mobility interactions that enables fluid and portal-based transfer of information across surfaces and devices. Portable devices can share information on the table via dynamic portals like the ones used by Shen et al. [46]. They present a setup where users can share information by wirelessly connecting their laptop on a touch-sensitive table, called UbiTable. The table can then be used as a public space to share information while the laptop is used as private space for personal information. Ownership of the documents on the table can be recognised via a distinctive colour on the border of each document.

Many other systems have been designed for multi-device environments. Johanson et al. [29] were one of the first to focus on augmenting dedicated meeting rooms with large displays and integrated portable devices. The interactive workspace prototype is called iRoom, while the software infrastructure for the environment is called iROS. The workspace contains three touch sensitive whiteboard displays, a custom-built high resolution display called the interactive mural and a table also containing a display. This setup can be seen in Figure 2.6. Additionally, cameras, microphones and wireless support are installed in the room. The authors came up with three important task characteristics that should be available in such interactive rooms. The first one is the *moving data* task, which allows users to move data from one screen

Figure 2.6: iRoom setup. The high resolution Mural on the far left mounted in a wall, the three whiteboards on the wall and the table in the middle. [55]

to another screen and across devices. The second one is the *moving control* task, users should be able to control the devices from their location to optimise collaborative sessions. The last task is allowing *dynamic application coordination*, which means that each application or tool should be coordinated with one another. All applications communicate and are coordinated via an Event Heap. They all send events and listens for events to which they can respond to. The Event Heap is also used by the i-Land group [54] with their own software.

Figure 2.7: i-Land setup. In the back the DynaWall and one CommChair, on the right the InteracTable and in the front left the ConnecTable.

The i-Land environment is quite similar to the iRoom setup, both use an interactive wall and an interactive table. The i-Land environment also contains mobile and networked chairs with integrated interactive devices, called *CommChairs*. All these components are called roomware components. Users can interact with this roomware via finger or pen gestures. *CommChairs* allow users to share information to other chairs and to the interactive wall, called the *DynaWall*, by, for example, making remote annotations. Information structures can also be passed around via a *Passenger*, which can be seen as a physical information carrier that can be moved to a new location by putting it in a device called the *Bridge*. The information assigned to the Passenger will then be displayed at the new location. Any physical object can be turned into a Passenger, as the information is not stored on the object itself but is only linked to it. Later on, the i-Land environment has been expanded by adding a new component called the *ConnecTable* [56], which has been designed for individual work and cooperation in small groups. The height and angle of the display can be adapted for optimal use of the roomware. The entire setup of the i-Land environment is shown in Figure 2.7.

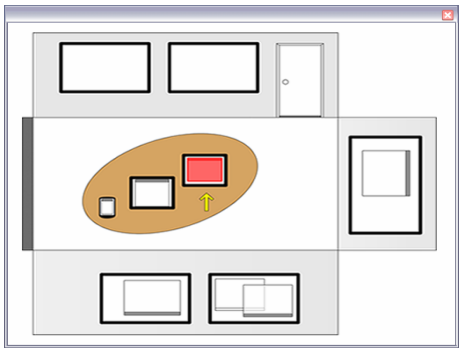

Figure 2.8: The ARIS interface, showing multiple screens along with the open windows of every screen in the room.

Various research continued in the field of smart, interactive or intelligent rooms, mostly focussing on communication between devices, management of multiple users and sharing of application states [30] [18] [11] [7]. They mainly use traditional desktop user interfaces and address the collaborative aspect of workspaces. While these approaches sound promising, a few authors follow another approach by developing a *user interface* particularly designed for optimal communication of devices in interactive spaces. Biehl and Bailey [5] designed ARIS, an interactive space window manager. It allows to manage applications across all sorts of devices through a common interface showing the state of every screen on an iconic map of the room. The ARIS interface is

illustrated in Figure 2.8. Via this interface, users can relocate an application window from one screen to another simply by dragging it to the corresponding screen on the map.

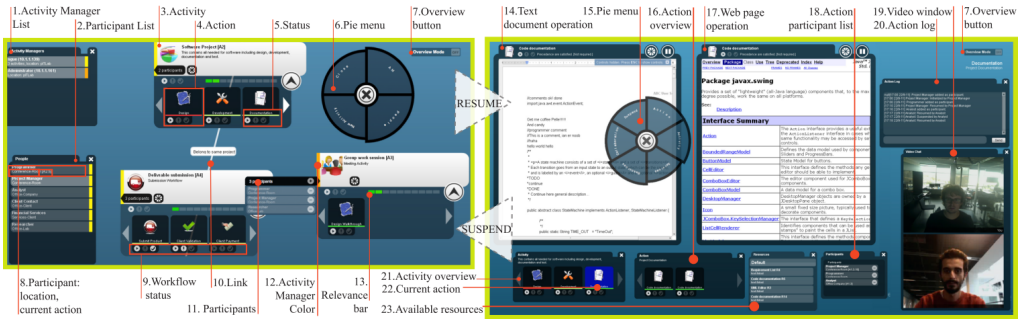

Figure 2.9: The ReticularSpaces interface, showing the Activity View on the left and the Action View on the right.

Bardram et al. [3] [4] went further down that path by introducing ReticularSpaces, an activity-based framework for distributed and collaborative smartspaces. The ReticularSpace interface or *ReticUI* proposes a completely novel user interface metaphor that relates to the activity of the users rather than to their desktop. ReticUI has two main views, the *Activity View* and *Action View*. The activity view shows a list of available activity managers, a list of all users and their location and all relevant activities together with their list of action and participants. The action view is used when the user resumes an action and shows all resources used when performing this action as well as an action log and a list of participants. These views are shown in Figure 2.9. The content and layout of certain relevant parts of the ReticUI are synchronised across local and remote smart space displays. This guarantees that users working on different devices but on the same activity will see the same view at all times.

Others, also aim at facilitating the design of a common user interface that could be used by a wide range of different devices and displays. Jetter et al. [27] present ZOIL, an software framework for *post-WIMP* distributed user interfaces in interactive spaces. The authors present several design principles too, for post-WIMP interactions, such as interaction with multi-touch walls, interactive tabletops, and much more roomware devices. They give insight about how to design interactions for interactive spaces and various concrete examples. The ZOIL framework facilitates the construction of such interactive spaces that integrates these design principles, which could be quite challenging to implement. It already provides functionality through a collection of components and classes.

Figure 2.10: GlueTK user identification.

Another interesting framework that abstracts the complexity of building an interactive room is presented by Camp and Stiefelhagen [60]. They introduce glueTK which also simplify the design of such interfaces and makes the distribution across devices transparent for the programmer and user. Unlike Jetter et al., Camp and Stiefelhagen focus more on persons and their position in the room relative to the screen. This is done by giving visual feedback once a person stands in front of a wall display, as illustrated in Figure 2.10. An example application is presented in order to show the functionality of the glueTK framework.

Although these mentioned systems sound promising, they show certain shortcomings for an Office of the Future environment. They mostly focus on DUI design on screen-based devices and/or interactive surfaces, ignoring the fact that a desk does not only contain electronic devices. It includes digital as well as physical ubiquitous user interfaces. Paper documents are still omnipresent in most offices and they can be seen as another user interface in the context of DUI. Unfortunately, none of these previously mentioned systems include DUIs supporting cross-media interaction across digital and physical information.

2.2 Cross-Media Interaction

With the growing amount of digital and physical documents, challenges arise about how to optimally classify and find all of these documents. Already in the early nineties, researchers started investigating possible solutions to combine the physical and digital information spaces, which led to hybrid surfaces. These surfaces combines different media, such as paper documents, voice, ambient sounds and all sorts of electronic devices. A good example

that illustrates cross-media interaction spaces can be found in Giaccardi's [22] paper which describes cross-media interaction for a virtual museum combining sounds and other media types. However, in this section, we focus on cross-media interaction which combines the affordances of paper and digital devices.

The first system combining both digital as well as physical media was introduced by Wellner [61], who presented the DigitalDesk. Wellner explains how his DigitalDesk could provide a Computer Augmented Environment for paper. The digital desk has a video camera mounted above it to allow extra features by reading the documents placed on the desk. An electronic projector is also mounted above the desk to allow projection of electronic objects on the desk surface, as shown in Figure 2.11. The use of the DigitalDesk has been demonstrated through several applications, such as using it as calculator, translator, paint program (PaperPaint) and remote sharing of the desk (DoubleDigitalDesk).

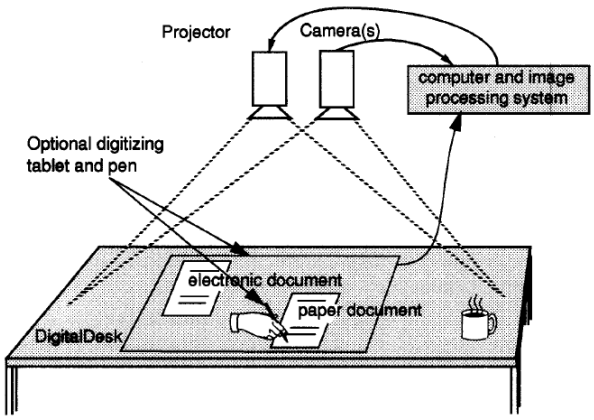

Figure 2.11: DigitalDesk setup

By combining paper and digital environments, Wellner aims at using the best of both worlds. When users interact with objects in the physical world, they use their natural skills to manipulate different objects simultaneously without needing to think about it. In contrast, when users physically interact with electronic documents, more skills and habits are needed to master each workstation since different skills are needed for different workstations. The classic approach to solve this problem is by using the *desktop metaphor* [50] [51] [31], where users can use direct manipulation to handle virtual objects [47]. This allows users to become more familiar with the workstation by using their knowledge of the physical world, this way less learning is required. That is why the DigitalDesk has three important features. The

first one is interaction with pens and bare fingers. The second one is projection of electronic images on the surface, while the last one is scanning and recognising paper documents placed on the desk. This novel approach of human-computer interaction techniques aims at making a normal desk more like an electronic workstation instead of making the workstation more like a physical desk. This makes it possible for the users to do certain tasks on paper that are more convenient that way, such as copying numbers, replicating sketches and remote sharing and editing of documents. The goal is to enhance paper by facilitating these tasks via the introduction of digital information.

An important paper that builds on the previous one is the paper from Everitt et al. [14]. This paper introduces the DocuDesk, a prototype of an interactive desk. It demonstrates interaction techniques between digital and paper documents. Once a paper document placed on the desk, a menu appears with different interactive options available for the user, as illustrated in Figure 2.12. A user can then send the document via email, see its digital counterpart, take a snapshot of it, link it with other documents or pin the document to the desk.

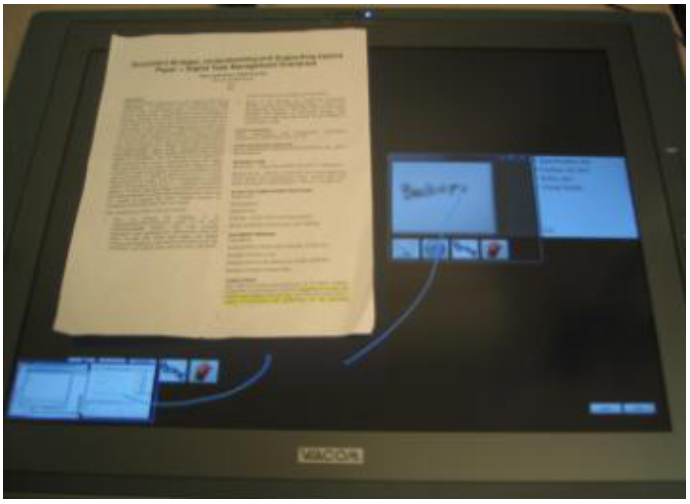

Figure 2.12: DocuDesk setup

One of the challenges of combining many digital and paper documents is that users often tend to have difficulties when resuming their task later on. This is, for example, because they do not remember where they put the documents they were using when they left. This challenge of trying to resume a task is called *task rehydration* or *task resumption*. DocuDesk cope with this problem by allowing users to link multiple documents that belong to the same task. Once a document is placed on the desk, the user can

resume the entire task by clicking on the "open all" button which opens all documents related to the paper placed on the desk.

A more recent study, focusses on the spatial usage of the documents on a desk [53]. The paper reveals the problem of physical occlusion when combining paper and digital documents, which happens for example when a paper document lies on a digital document. However, in this paper, they explain how to use this occlusion in the user's advantage. After studying different tasks like grouping, sorting, searching and comparison tasks on a tabletop including digital and physical papers, the spatial environment and how users cope with occlusion is analysed. When performing this task, it is interesting to see that when users are working with both paper and digital documents more space on the surface of the tabletop is used. While participants were asked to group different documents, a lot of them made hybrid piles, namely piles of physical and digital documents, hereby choosing willingly for occlusion of certain digital documents. This shows a perfect example of taking advantage of occlusion to have a visible representation of a relationship between different documents. Based on the results of their study, several design guidelines are given. A few of them can be of interest regarding this thesis, such as the support of hybrid grouping and zooming to focus on specific parts of a document.

Another study that focusses on piles on a desk, is the one of Kim et al. [34]. The authors present a system to track piles of paper documents on a desk and link them automatically to the corresponding documents in the user's computer using an overhead camera.

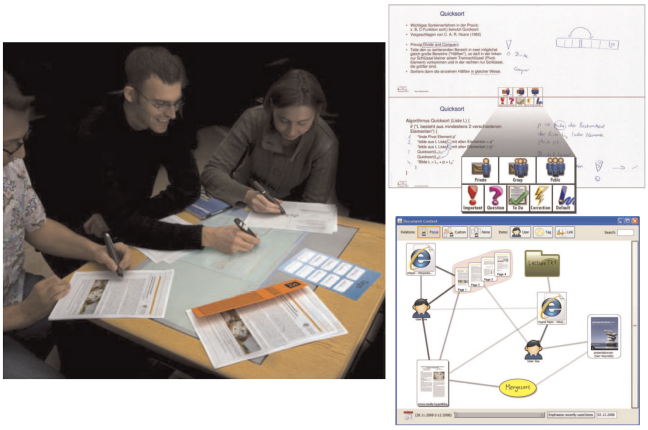

Figure 2.13: CoScribe

An interesting thing to consider and study is collaborative work on such a desk that combines paper and digital documents. Steimle et al. [52] wrote a paper about this, where they describe their prototype, called CoScribe,

which focusses on combining paper and digital documents in a way that it is easy to use for multiple users. Their main focus is that the system could be used in the context of learning and knowledge work. For example, taking notes during courses, review them later on and establish relationships between documents. Hereby, users are building structural knowledge which helps them better recall and comprehend information. The collaboration between users is mainly done by providing pen-and-paper-based interaction, which makes it possible for users to annotate, link and tag all kinds of documents. Intuitive multi-user visualisations were provided and tested via a user study. Their Pen-and-Paper User Interface (PPUI) has been implemented based on their analysis from related work. Figure 2.13 shows the main functionality of this user interface. To promote remote collaboration between users, their annotations can be shared. This is done by the authors via *button-based differentiation*. They print buttons on each paper which can be clicked on via the digital pen, allowing users to choose the visibility level and also allowing users to categorise annotations with semantic types. To link documents, hyperlinks are used, which are modelled as a symmetric association between a collection of documents, entire documents and part of documents. CoScribe also provides an entire overview of all documents, which focusses on relations between these documents and user activities on these documents. The visualisation is done via a graph, where each node is a thumbnail that represents a document, book, folder, tag or user. Tagging is done by using predefined categories and user defined tags. The user study evaluation indicates that CoScribe enhances the work performance. This is done by providing an efficient structuring and retrieval of documents, and user satisfaction, due to its seamless interaction technique between paper and digital documents. This seamless interaction is achieved via note taking and annotation with the digital pen.

Klemmer et al. [35] also focus on user collaboration, but this time by using a tangible user interface. They describe an interface for collaborative design of websites and call it *The Designers' Outpost*. The tangible interface combines physical and digital information by using post-it notes and images, and an electronic whiteboard, as shown in Figure 2.14.

Figure 2.14: The Designers' Outpost

To introduce structure and annotations an electronic pen is used. To capture movement a rear-mounted camera is used and to register annotations a front-mounted camera is used. By making use of these cameras the state of the board can be captured, which allows the transition to electronic tools like DENIM, a sketch based tool to design websites that can be used to further refine the design via a visualisation such as sitemaps, storyboards and page schematic representations. The Designers' Outpost supports different interaction techniques, namely adding notes by adding a regular post-it note on the board. Creating links by drawing lines between notes with the board stylus. Deleting notes and its links by taking the note off the board. Moving a note by physically moving it. Via the context menu that can be obtained by tapping on the note, it can be replaced by its digital counterpart or the note can be deleted. Finally, the state of the board can be saved by pressing the *Save* button on the board so that it can be opened later on in DENIM. The Designers' Outpost also provides three physical tools to manipulate information. The first one are the board styli, which are used to draw on the board. The second tool is a move tool, which can be used to move digital content on the board. The third one is an eraser, which removes ink on the board and is thus used to delete links between notes. The Designers' Outpost has been built to support many affordances of paper combined with the advantages of digital media. A user study performed with 15 professional designers showed that grouping of notes by colour automatically based on proximity, the note outline and the menu feedback were distracting for the users: "*Too many things flashing*". From this it can be concluded that only explicit user

actions should make the system perform visible (response) actions. A new functionality was added, namely making a digital counterpart for all post-it on the board, which seemed necessary to make it easier to move on to the next step of the design process. For future work, versioning of the system would be a nice extra feature. Many designers were enthusiastic when using The Designers' Outpost and when being able to import their work in a tool like DENIM on their computer.

Some of the mentioned systems might find their way into real office settings in the coming years. However, they all present *isolated* solutions in the sense that the used data and the functionality they provide is not accessible to third-party applications. Consequently, these solutions cannot be integrated in DUI settings. Moreover, some other authors present ubiquitous interfaces, such as LEDs that have been integrated into physical artefacts. This can be seen in the study of Jervis and Masoodian [26], where they present SOPHYA, a system to digitally manage physical documents placed in augmented artefacts. Such small augmented user interfaces should be re-usable by other applications. We can observe a gap between the work situated in the field of distributed user interfaces and cross-media user interfaces including physical user interfaces.

3 Experimental Setup

In order to define the requirements for the XUI framework, we have developed a few use cases in an Office of the Future setting. The defined use cases are all in the context of researchers and their prevalent activity of writing research papers and attending important meetings. In this chapter, the Office of the Future setup will first be explained. Further on, the possible fluid cross-device and paper-digital cross-media interactions that occur while using this setup will be demonstrated via the use cases. Finally, a short conclusion about the examined use cases will be given.

3.1 Office of the Future Setup

The setup for the Office of the Future environment includes a hybrid tabletop surface, a Windows File Explorer extension and a number of augmented ring binders in a bookcase. The tabletop has been built by combining a TablerTV 46" multi-touch frame with +32 touch points, a Samsung UN46F5000 TV and an IKEA table. First, the touch frame has been clipped to the TV screen. After that, a few holes were made into the table for the ventilation of the screen. Finally, the TV screen was fixed on the table, which completed the tabletop setup. This setup can be seen in Figure 3.1.

Figure 3.1: Tabletop setup

In order to complete the Office of the Future setting, we mounted a camera on the desktop next to the computer in order to recognise physical documents. The initial goal was to mount it above the tabletop, such as in the DigitalDesk [61] and DocuDesk [14] setup. Unfortunately, the multi-touch frame uses infrared detection, so when a sheet of paper is placed on the multi-touch tabletop it is recognised as multiple touch points. This prevents us from distinguishing the user's touch points from the ones generated by the document, hereby confusing the tabletop user interface. Therefore, the camera has been mounted on the desktop next to the tabletop. Placing the camera next to the tabletop should not influence the fluidity of the user's interactions since the user can simply put a document on the desk under the camera, which will recognise the document based on its fiducial marker. All documents are tagged with such a marker and are stored in the augmented ring binders in the users' bookcase. The ring binders are augmented with a Digispark board and an RGB LED as well as an infrared shield. This part of the setup is called the ring binders application and can be used to enable interaction between the tabletop and the physical world. An illustration of this part of the Office of the Future setup is shown in Figure 3.2.

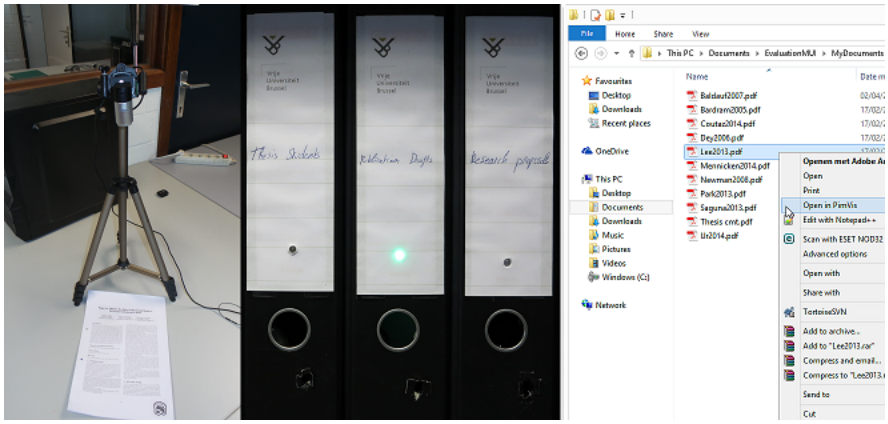

Figure 3.2: On the left the mounted camera. In the middle the augmented ring binders. On the right the extended File Explorer

For another office augmentation we added a desktop computer, which provides the interaction between the tabletop and the user's file system. By right-clicking on a document in their File Explorer, users can open the document on the multi-touch tabletop. This component is called the Windows File Explorer extension or extended file system and is illustrated in Figure 3.2. The final setup of the Office of the Future is shown in Figure 3.3.

Figure 3.3: Final setup of the Office of the Future — The tabletop showing the PimVis interface, the desk-mounted camera ready to recognise an augmented paper document, the extended file system on the desktop computer, the augmented ring binders in the bookcase

To use the multi-touch screen, we installed the TablerTV drivers on the laptop and connected both via a USB and HDMI cable. The tabletop has been integrated into the interaction space by using the Personal information management Visualisation (PimVis) user interface. This allows users to *explore* their personal information space and helps them in *re-finding* digital and physical documents. The design of PimVis is based on two principles. The first principle is that users make associations between documents to better remember them [33]. The second one is that users often use a contextual cue for re-finding activities. They often remember during which task or in which context they used a particular document [58]. PimVis has three different views in order to show the right amount of information to a user.

The first view is the *context view*, where all of the user's contexts are shown. The different contexts can be explored via the bubble visualisation, which is shown in Figure 3.4. Each bubble represents a context. Since a context can have sub-contexts, bubbles can also be packed within each other. For example, if certain users have a context *meeting*, it is likely that they will have certain sub-contexts such as *department meeting* or *family meeting*. These sub-contexts will then contain the documents used during the specific meeting. The size of a bubble representing a specific context depends on how many documents are in that given context.

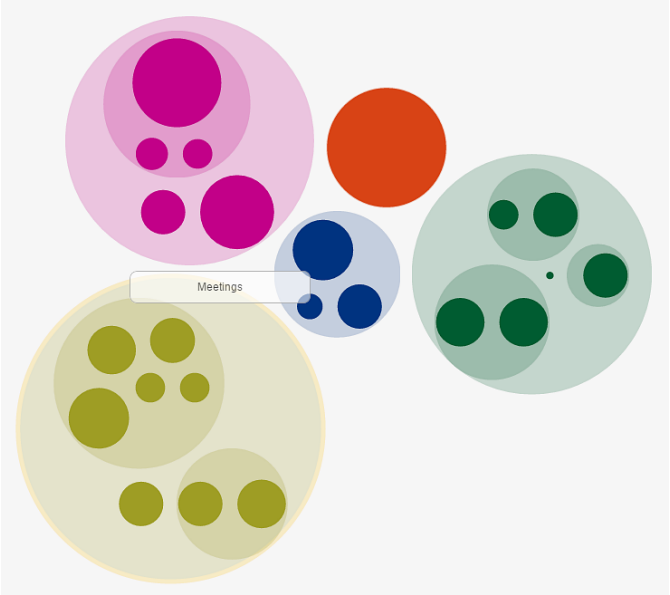

Figure 3.4: Context view

The next view is the *document view*, where all documents of a specific context can be explored via a graph visualisation. Each document of the context is represented by its preview surrounded by a coloured frame representing the importance of the document in that context. Documents can be related to each other, which is represented by a line between the related documents. Furthermore, documents can also be filtered using a control panel on the right and the timeline underneath the graph visualisation. This view can be seen in Figure 3.5.

Finally, the last view is the *focus view* which provides more details about a specific document. This view shows a zoomed-in view of a selected document, which is placed in the centre of the screen together with all its related documents. Via this view, users can find the selected document in the physical or digital space by using the *re-find* button. In case of a physical document,

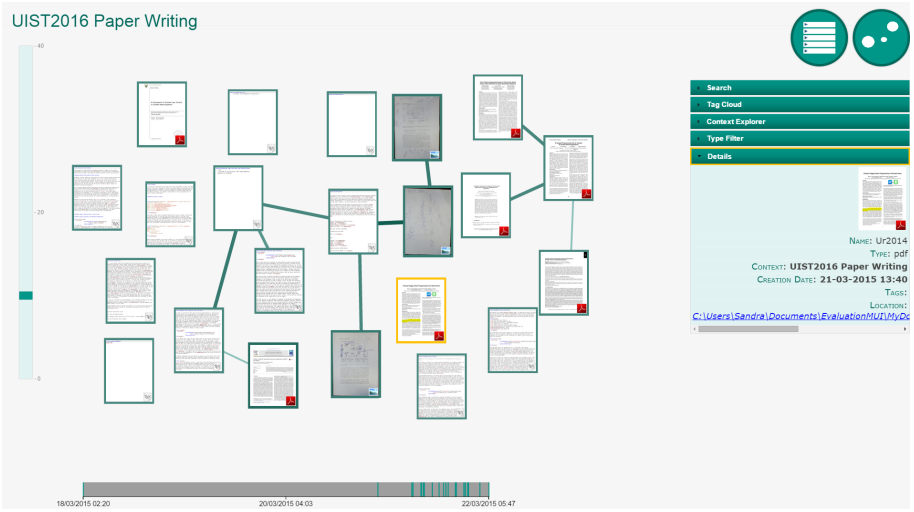

Figure 3.5: Document view

its location will be shown via the ring binders application, which will light up the LED of the ring binder in which the document is stored. This is done by sending an infrared signal via a PhidgetIR module. If the document is digital, its location will be shown via the extended file system application, which will open the folder where the document is stored in the Windows File Explorer on the user's computer. This last view can be seen in Figure 3.6.

Figure 3.6: Focus view

3.2 Use Cases

In order to illustrate the benefits of the setup, a few simple use cases are described. The first use case tells a story about researcher Alice who needs to write a paper for a specific conference. The second use case illustrates how an unexpected meeting with a thesis student can turn out.

3.2.1 The Paper Writing of Alice

The first use case, is about the writing of a paper which is a common task for a researcher. Imagine a scenario where Alice, a very talented researcher at the WISE lab, wants to submit a paper to the ITS conference[1]. To start writing the paper, she opens the File Explorer on her Windows computer and browses to her prepared ITS folder, which already contains a few papers that she needs to read. She opens a LaTeX editor and starts with writing an introduction for her paper. After a while, Alice needs a specific paper that she remembered using during a meeting with one of her students called Bob. In order to find this paper, she switches to the PimVis visualisation located on the tabletop next to her desk, goes to the *context view*, where all her contexts are grouped together, spots the *meeting with Bob* context and navigates to it. At a glance, she recognises the needed paper and wants to copy the paper to her ITS folder. By using the fluid interaction system, she can simply tap on the *re-find* button in the visualisation, which will open Alice's File Explorer in the right folder with the corresponding paper selected. Now, all she needs to do is to open the file and make a few annotations or copy/paste the PDF into her ITS folder for later use. Note that if these two applications did not work together, Alice would have to retype the whole file path provided by the document description in PimVis in her File Explorer or she would have to manually navigate to the document, which is not very practical. Furthermore, without PimVis she would have had to remember where she stored the document that she used during the meeting with Bob.

A while later, Alice is looking for related documents of a paper stored in her ITS folder. By right-clicking on this paper in her file system and selecting the option to visualise the paper on the tabletop, she can see the selected paper in the *focus view* in the PimVis interface. The paper is shown in the centre of the tabletop surrounded by all documents that are related to it. By selecting those documents she can see more information about each of them and find out more easily where they are stored. Suppose that this handy interaction between those interfaces did not exist, then Alice would have had

[1] http://www.its2015.org

to navigate to the *context view*, go to the context *ITS paper writing* and once inside this context, go to the *focus view* of the paper which she wanted to see the related documents of. This process would have taken much more time and if the PimVis visualisation did not exist, Alice would have had to remember these related documents and manually search for those in her File Explorer and hope she did not forget one of the related documents.

A few days later, Alice is about to write the *implementation* section of her ITS paper, but she does not remember her whole code architecture anymore. Luckily, she draws this architecture on her drafts of the system together with certain important details about the implementation. Unfortunately, Alice is not very organised, has a lot of ring binders in her cabinet and thus does not remember where she stored the draft. However, she remembers drawing it when working on the implementation of the system. Therefore, she turns to the PimVis visualisation and navigates to the corresponding context. There she sees the draft that she is looking for and opens it in the *focus view*. Once in this view, she taps the *re-find* button, turns to her bookcase and sees the LED of the ring binder containing the draft lighting up. Alice picks up the ring binder and finds the draft she was looking for together with several crucial notes she made about the implementation of her system. Without this interaction possibility between the tabletop and her ring binders, Alice would have to re-find the draft by reading the location in the detailed information of the draft in the PimVis interface and then look for the ring binder with the correct label in her bookcase. This scenario looks quite simple when there are 5 to 10 ring binders, but when someone has a cabinet full of them, the interaction system could prove to be faster. Moreover, if she did not have the PimVis interface she would have to go through all her ring binders in order to find what she was looking for.

3.2.2 The Unexpected Meeting

The second use case tells the story of the meeting of researcher Arthur with one of his thesis students called Morgana. The meeting started off one day too early as Morgana got the day wrong, causing Arthur to be unprepared for the meeting. During the meeting, they talked about the progress that Morgana made on her thesis these last few weeks. After a short explanation of what she did, Morgana asks Arthur whether he received the last version of her text that she sent via mail. Fortunately, Arthur had already printed and corrected the text, but he forgot in which ring binder he put it (given that he has 5 ring binders with the label *Thesis Students*). Luckily, Arthur has the Office of the Future setting installed in his office. In order to find out which

one contains the text, he simply goes to the context *Morgana's thesis* in the PimVis interface, double taps on the right document preview and taps the *re-find* button to see the LED of the needed ring binder lighting up. Once the text is found, he explains his remarks to Morgana.

Arthur had also written additional reading material for Morgana on a separate note. Since the note is related to the text, it appears as one of the related documents in the *focus view* on the tabletop. By a simple double tap on the note, he brings it to the centre of the view, making it possible for both Morgana and Arthur to read the content of the note without looking for it in the ring binders or on his desk. The note is linked to various related documents in the *focus view*. Those papers are described on the note as the extra reading material for Morgana. Arthur double taps on one of these papers and tap on the *re-find* button to locate it in his file system. Triggered by this event, the computer opens the File Explorer in the right folder with the needed paper selected. Arthur also selects several other papers located in this folder, which are also a part of the documents Morgana needs to read as extra reading material and send them to her via email.

Later on, Arthur asks her if she has additional questions about her thesis. As a matter of fact, she does have important questions concerning the planning she made in the beginning of the semester. To have an image of what she is talking about, Arthur goes from the *focus view* back to the context *Morgana's thesis* and double taps on the preview of the Gantt chart containing Morgana's planning. With the Gantt chart opened in the *focus view*, they discuss the changes she wants to make and the questions she has. After a while, they decide to modify the Gantt chart, so they needed to open in on the desktop. Due to the *re-find* button, this is done in a few seconds. When all modifications are completed, they save the file and conclude the meeting.

Notice that this unexpected meeting would have been much more chaotic without the PimVis interface and the fluid interaction system. It would have started off with at least 10 minutes of searching after the latest version of Morgana's text in the ring binders. Then Arthur would have needed to find the note with the additional reading material, which might still have been easy. However, finding the reading material on his computer would maybe have been a harder task, especially if the documents were not stored in the same folder. Even if they were, it would not always be easy to manually re-find in which folder it was located. Discussing the changes Morgana wanted to make to her planning would probably be easier to do when the Gantt chart is shown on the tabletop in *focus view*. Finding the Gantt chart would have been a difficult task without the PimVis interface, since it has been made a few months ago, which makes it more difficult to find in the computer's file

system. Due to the fluid interaction between the tabletop and the desktop, it was also simple to open a document on the computer without having to manually search in the file system.

3.2.3 Comments

The above-mentioned scenarios are two of the many scenarios that could be used to illustrate the benefits of a fluid cross-device and cross-media interaction space. The PimVis interface together with the extended file system and augmented ring binders interface have been built as example user interfaces to allow this fluid interaction system. Note that the connection between the desktop, tabletop and ring binders should not be hardcoded, so that users could easily extend their Office of the Future setting. If users buy new ring binders or an entirely new filing cabinet that is compatible with their current office setting, the added artefacts should also be able to be re-found via the *re-find* button of the PimVis interface. In the case that some PimVis functionality was hardcoded this would not be possible without redeploying the whole setup. The Office of the Future setting must be flexible in order to allow users to customise it and use it as they want.

The attraction of visual displays, when compared to textual displays, is that they make use of the remarkable human perceptual ability for visual information. Within visual displays, there are opportunities for showing relationships, [...] draw attention to certain items in a field of thousands of items, [...] rapid selection, and feedback is apparent. The eye, the hand, and the mind seem to work smoothly and rapidly as users perform actions on visual displays.

— Ben Shneiderman [48]

4

PimVis

In this chapter we introduce PimVis, a tabletop user interface which is a part of the Office of the Future setting presented in Chapter 3. The PimVis interface has been built to allow users to visualise, explore and manage their personal information space. Furthermore, it tries to cope with the problems encountered in the field of personal information management (PIM), which are information overload, information retrieval and task resumption. This is accomplished by using the Object-Concept-Context (OC2) [59] conceptual framework, which focusses on context-aware and cross-media personal information management. The OC2 framework also offers to possibility to develop an object-oriented user interface, where attributes can be assigned to every object in the information space.

In the previous chapter, the interaction possibilities of PimVis were demonstrated by using different use case scenarios. In contrast, this chapter illustrates the potential of the PimVis interface by going deeper into its creation, components, implementation and functionality. First, existing visualisations built to visualise large document spaces will be examined. Based on these visualisations, the user interface requirements are composed and an iterative design process is started. Finally, support for multi-touch gestures and the implementation of the interface is explained.

4.1 Existing Visualisations

In this section, different types of existing visualisations are discussed. Most of them are used to visualise large document spaces or simply large information spaces. These visualisations were used as an inspiration when designing the PimVis interface.

Wise et al. [63] proposed a spatial representation of document characteristics, which aims at reducing the mental overload when browsing and analysing the documents. The authors only focus on text documents, where the content of the text is used to form the characteristics of the documents in the visualisation. The authors also describe two major visualisation approaches to represent text documents, the *Galaxies* and the *Themescapes* views. These views were developed to make browsing and selection of documents in a large context space easier. They were introduced in the Spatial Paradigm for Information Retrieval and Exploration (SPIRE) application for Unix platforms and in the IN-SPIRE software for Windows users.

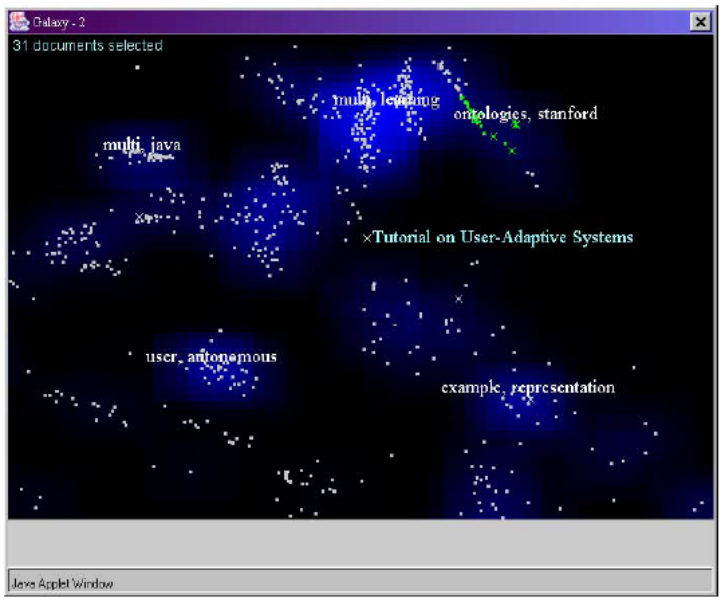

Figure 4.1: Galaxy visualisation

The Galaxy visualisation shows a scatterplot where points represent different documents as illustrated in Figure 4.1. Clusters of documents can be clicked, to have a more detailed view of specific documents within the cluster. These documents can then be annotated, analysed and grouped. A slider is used to incorporate a temporal notion in addition to the spatial one. The users can set years, months, days, hours or minutes as different granularity of

temporal units. By adding this component, temporal links can be discovered that relate to different topics in documents. Cluster patterns can then, for example, indicate some causal relationships related to some historical event.

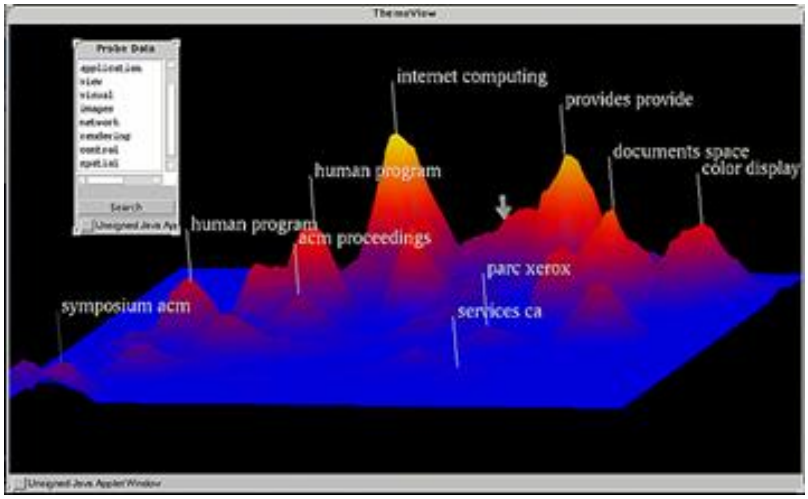

Figure 4.2: ThemeScape visualisation

The second visualisation is the Themescape visualisation. In contrast to the Galaxy visualisation which was in a 2D space, Themescape presents 3D landscapes of information as shown in Figure 4.2. Landscape surfaces provide relevant information about topics or themes, which results in a visual thematic terrain that shows primary themes. It gives an idea about how documents are related to certain themes through valleys, peaks and cliffs. ThemeScape is very useful for automated summarisation of one or multiple documents since it displays all topics in one visualisation. The analysis of documents is promoted through exploration of the document space by slicing through the thematic peaks to see their composition. Themescape has later been used to produce maps of international news[1].

[1]http://mappa.mundi.net/maps/maps_015

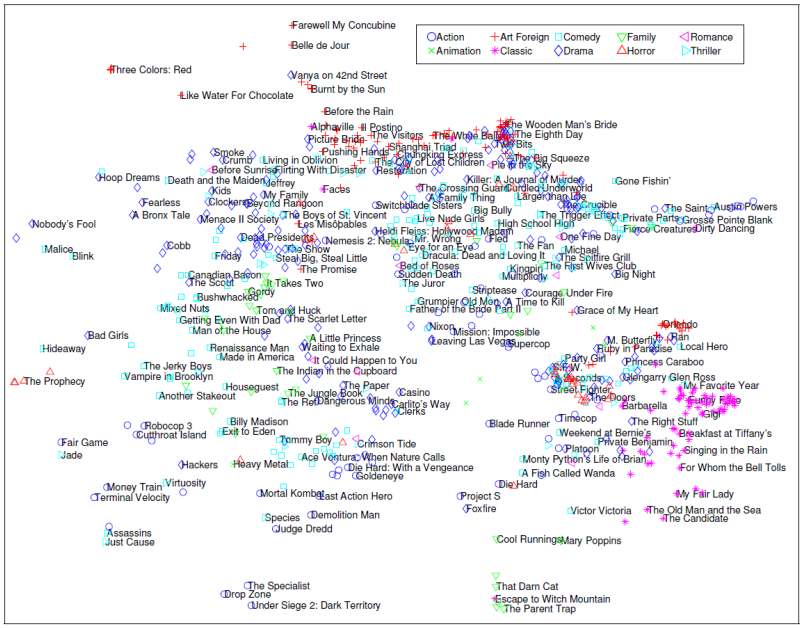

Figure 4.3: Movies visualisation by PLSV

In both visualisations, starfields and topographical maps were used as display metaphors. Starfields are point clusters that indicate patterns of interest, while topographical maps show peaks and valleys that are easy to spot, which offers another perspective of the same information. These metaphors provide an overview and details of text documents without needing to change the view. The more similar two documents are in terms of context and content, the more closely they will be located in the 2D or 3D space. Similar to Wise et al., Iwata et al. [25] propose a visualisation based on the topics of documents. The authors present the Probabilistic Latent Semantic Visualisation (PLSV), which is a probabilistic topic model made using an expectation-maximization (EM) algorithm. The resulting visualisation of this model, again, puts similar documents closer together based on similar topics. The visualisation can also be used for visualising other kinds of elements, such as movies, as shown in the representation of a movie database in Figure 4.3, each point represents a movie and the shape and colour of the point represents a movie genre.

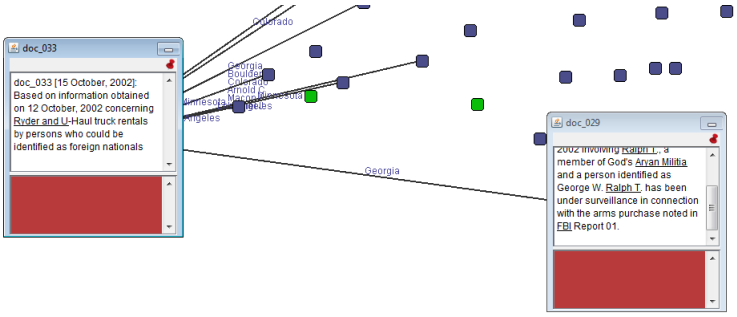

Figure 4.4: ForceSpire visualisation

Continuing on the path of dot-like visualisations, Endert et al. [13] propose a visualisation to spatially explore textual datasets, called ForceSPIRE. The ForceSPIRE visualisation represents the text documents as large dots and just as in the previous papers, similarity between documents is represented by spatial distance. While Wise et al. [63] used a slider to interact with the documents in the document space, this paper proposes a new interaction, called *semantic interaction*. Semantic interaction combines two primary parts of sensemaking [42], which is the process that allows users to make sense of complex datasets through interactive exploration. These two parts are called foraging and synthesis. During the first part, users filter and gather collections of information. In the second part, they use and process this information. This interaction allows users to interact directly within the visual metaphor. Basic interaction techniques like searching, highlighting, annotating and repositioning are supported. In order to show semantic interaction, the ForceSPIRE prototype shown in Figure 4.4 is introduced. By clicking on the document points, links appear between the points forming a directed graph representing the relationship between the selected document and its related documents, which are the documents with at least one matching keyword. The connection strength between different documents depends on how many keywords they have in common. The highlighting of terms creates a keyword for the document and increases the importance value of this keyword. Searching is done via a searchbox, resulting in a highlighting of the searched terms or keywords. The annotation of documents can result in new links between them since a link is created as soon as documents have keywords in common. Finally, the repositioning of documents can be done by just dragging and dropping a document point. Additionally, users can also pin a document, forcing it to remain at the same location and add new documents to the visualisation.

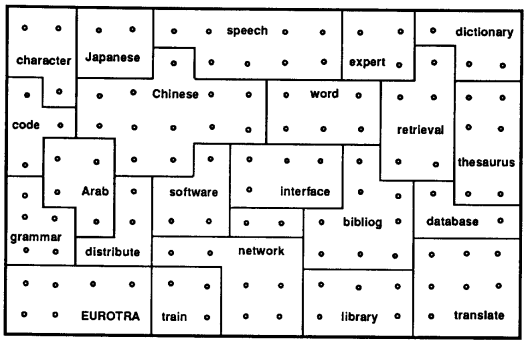

Figure 4.5: The map display for a document collection on multilingual information retrieval

Another kind of document space representation is proposed by Lin [36], who uses a map display to allow users to retrieve information more easily. Again, visual cues like dots, links, clusters, area and spatial arrangements are used to show relations between documents. This visualisation is illustrated in Figure 4.5.

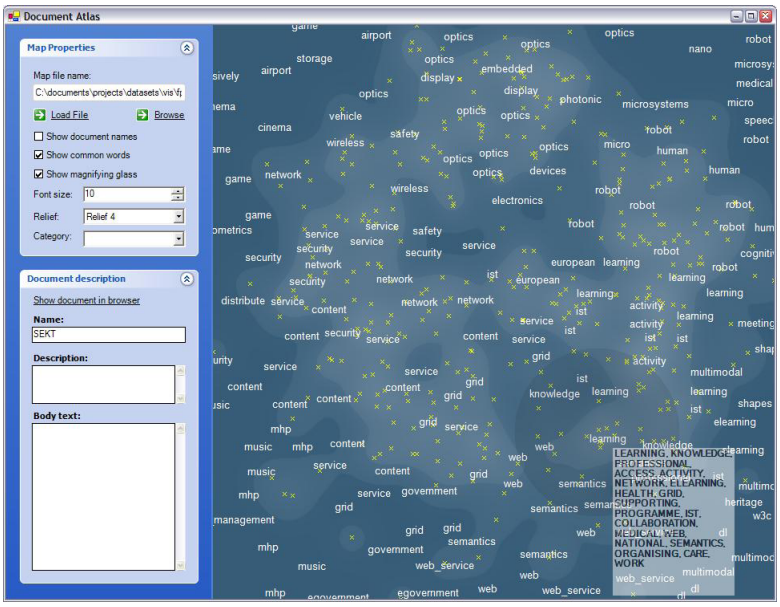

Figure 4.6: The visualisation of European IST projects from 6th framework.

Fortuna et al. [19] also use points in a two-dimensional space to represent documents. In addition, they also use the background to convey information, as illustrated in Figure 4.6. This is done by changing the colour saturation according to the degree to which a document has a specific content.

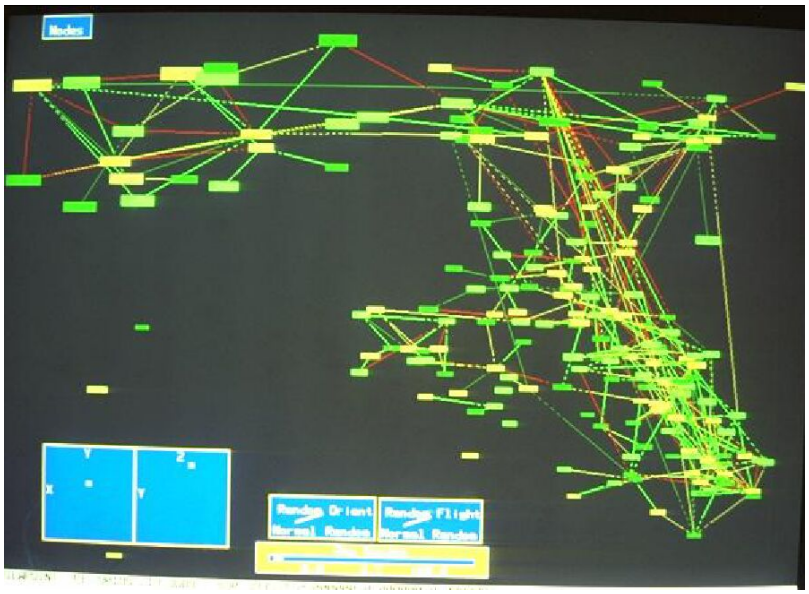

Figure 4.7: The SemNet visualisation of a semantic network

Fairchild et al. [16] put the problem of coping with large knowledge bases on the table when providing their 3D graph representation. They explain that displaying a graph, with more and more nodes and edges, can become quite overwhelming. Firstly, because of the limitation in speed to process and show all the information and secondly because of a user's inability to distinguish and pay attention to all objects on the display. The authors try to cope with this issue by introducing SemNet, which reduces the total amount of information by displaying only useful subsets of the information. To do that, they integrate selection by the user. For example, when different types of elements are displayed, the user can select a relevant type which will then be the only type displayed in the visualisation. Using the spatial metaphor of SemNet, other strategies can be used. For example, a consequence of a 3D view is that some elements are not shown when viewing the information from a certain angle, helping users to examine only the subset of elements which can be seen. Furthermore, elements nearer to the viewpoint of the user appear larger, allowing users to focus on local neighbourhoods. Other strategies are derived from the "Generalised Fisheye views" [21]. *"These views display details near the focal point and only more important landmarks further away"*. These views are useful for showing a balance between local details of objects and their surrounding context. This technique is often referred to as *focus+context* and can, for example, be achieved with the fisheye view technique, which is a special kind of distortion. The Semnet visualisation can be seen in Figure 4.7.

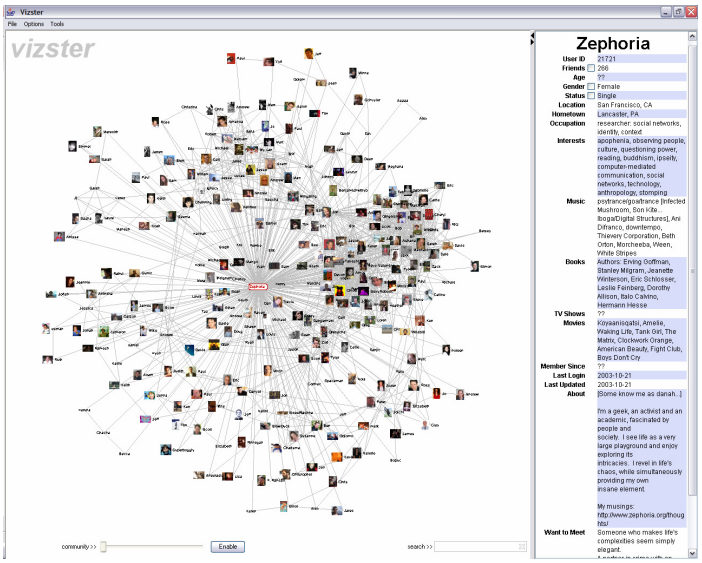

Figure 4.8: The Vizster visualisation

Since most of the systems that propose a visualisation of the document space do often not focus on the explicit relationship between documents, the Vizster [24] visualisation has been analysed. This visualisation is used to analyse social networks by means of a graph representation. The same visualisation techniques could be used on relationships between documents instead of relationships between people. The goal of Vizster is to provide a visualisation where users are able to easily explore, discover and increase the awareness of their network of friends. Vizster also aims at supporting an easy access to search and group patterns. Searching through the network is done via keywords. Matching nodes to the keyword(s) are indicated with "auras" while non-matching nodes are de-emphasised through desaturation. To get more information of a specific node, the user can click on it and more detailed information will appear on the right, as shown in Figure 4.8. To give a clear overview of which node is selected, a radial layout is proposed with the selected node in the centre.

4.2 Visualisation Design

In order to avoid the many problems that can be encountered when designing user interfaces, the pitfalls and basic rules for building a successful visualisations are examined. We start by investigating the Visual Information-Seeking Mantra proposed by Shneiderman[48]: *Overview first, zoom and filter, then details-on-demand*. Based on this mantra, Shneiderman suggests seven tasks that should be supported in interactive visualisations.

The first one is the *overview*, which allow users to get an overview of the entire data collection. The amount of content details that can be viewed can be controlled via zooming possibilities. This technique is called *overview+detail* or *context+focus*. Overview strategies are essential in order to design high quality user interfaces. The second task is the *zoom*, which is used to zoom into an item of interest. Users often are interested in some specific part of a collection. Therefore, they need to be able to control the zoom focus and zoom factor. Smooth zooming into starfield displays has been proven to be useful for the user experience [28]. *Filtering* is the next interaction that needs to be supported. It allows users to filter non-important items. Applying dynamic queries to items of a collection is one of the main concepts in information visualisation [2] [62]. By filtering content on the display, users can focus on what they are interested in. The goal is that users could instantly filter via control widgets, such as sliders, search boxes or buttons. The fourth task is *details-on-demand*, which is used to get details of a selected item or a group of items. When parts of a collection have been selected, users should be able to skim into the details of these selected elements. The most common approach is to give users the possibility to click on an item in order to get more information about its attributes. The fifth task is showing *related* elements. This allows users to see the relationships between items. For example, in the FilmFinder [1] interface, users can see all movies directed by a certain director by performing the right interactions with the system. In Vizster [24], a visualisation of social networks, users can ask the system to see all relations of a certain person. The sixth task is *history*, which keeps a history of a user's actions in order to be able to support the undo, replay and progressive refinement actions. Since information exploration is a process of many steps, keeping a history of actions could allow users to go back in order to re-find important information. The last task is *extraction*, which allows the extraction of some parts of a collection. This could be useful for a user to extract a set of items and save them to a file. The user could then send these items by email, print it or run some statistical algorithms on it. Not many systems support this interaction. However a version of this task is presented in the Designers outpost [35], where the visualisation could be saved and opened in another program on the user's computer.

Next to these tasks, Shneiderman [48] proposed seven core data types and their visualisation models to organise information spaces. Two of them were particularly interesting for the design of the PimVis visualisation. The first one is the network visualisation, which is convenient for capturing relationships between items by linking them together. In addition to the basic

interactions that can be applied to items, users often want to know more about the strength of the relation between these items. Since the goal of the PimVis interface is to visualise a user's personal information, which contains various relationships, the network visualisation seems to be well suited for this purpose. The second model is the timeline, which is used to visualise temporal data. Temporal data is often present when working on some projects and in general when working with personal documents. Frequently, someone is looking for some documents created at a certain point in time or during some specific time period. Consequently, certain systems focus only on how to project a collection of documents in time. Fertig et al. [17], for example, presented lifestreams, which shows a stream of documents over time. Time is also an important aspect that provides an overview of a user's information space and should therefore be present in the PimVis interface.

4.3 Iterative Design

Previous work in the domain of document space visualisation, together with Shneiderman's advices to design user interfaces, provided many ideas about how to tackle the problem of visualising large collections of documents. First, the design requirements were defined to get an idea of which interactions were needed by the PimVis user interface. After that, user interface prototypes were drawn on large pieces of paper. The focus was put on how to clearly show the important information of the documents and the relations between them. The prototypes were later used to implement a first draft of the visualisation. Since the PimVis interface needs to be designed for a multi-touch tabletop, different touch possibilities were explored. In order to evaluate the PimVis visualisation an iterative design process has been used.

In this section an overview is given of the design requirements followed by a description of every iteration of the design process of the PimVis interface.

4.3.1 Design Requirements

The main goal of the PimVis interface is to provide a clear overview of the user's personal information space by allowing users to visualise and explore their documents in an easy and convenient way. To do this, information will be shown based on how it is stored in the user's brain. This means that the hierarchy visualisation between files, like in the traditional file system, will not be used.

Instead, the interface will present information by using the associative trails between documents, as presented by Bush in his Memex a few decades ago [9]. In this subsection, the different types of requirements of the PimVis interface are described in order to achieve its goal.

Accessible Information

The PimVis interface should allow users to access a certain amount of information about documents, their relations with other documents and their contexts. Furthermore, users should be able to get information about a *document* name, type, contexts, importance in the current context together with its relations to other files in that context, its creation date and its tags or keywords. The visualisation should show the *links* between documents when they are related to each other in a specific context. The weight of these links along with the documents they connect should be clearly indicated and must be seen by the user at first glance. Any *context* in the visualisation should contain some documents which can be accessed. The number of files of a context should be visible.

General Requirements

The general requirements for the PimVis interface are based on the accessible information mentioned earlier. In order to get an overview of the information space and allow users to access details about the different documents in that space, different views or layers must be defined. By using multiple views, the right amount of information can be shown at once to the users. This system provides an *overview+detail* approach, as suggested by Shneiderman [48]. Furthermore, this technique has also been used by Wise et al. [63] in their visualisation which has been discussed in the related work section.

The PimVis interface should allow users to explore the information space via search queries in order to find a document based on its name. Moreover, users should be able to filter the content of the visualisation based on the tags, contexts, types and the creation date of a document. Users should also be able to focus on a selected document and get more details about that document. Switching between the three different views should be possible and easy, as well as switching between the different user contexts. Additionally, re-finding documents located in other user interfaces should be possible by making use of the XUI framework.

Usability and Navigational Requirements

The PimVis interface should be easy to understand given the appropriate documentation. Moreover, learning to manipulate the PimVis interface should be intuitive and take less than five minutes. Therefore, it is important that the interface elements and interactions should be consistent. The interaction between PimVis and the other interfaces must be fluid and easy to comprehend. The interface should also be attractive for users, which can be achieved by being consistent and by following a specified style guide. The style guide defined for the PimVis interface has been created by following the principles of Redmon-Pyle and Moore [44] and can be found in Appendix A.

Easy navigation between the different views must be straightforward for the users. In addition, users should be able to easily re-find any document within the PimVis user interface by navigating through the views, considering that users must be able to quickly resume their work after finding a document.

Design Constraints

Before designing PimVis, several important design constraints were noted. First of all, every document and programming code documentation will be written in English as well as the information provided by the visualisation to the users. Additionally, the interface should be usable without prior system configurations. The design must be modular in such a way that modifying or extending the programming code with new interactions does not have a negative impact on the proper functioning of the system.

Since PimVis is built in order to work on a tabletop, the visualisation needs to support multi-touch interaction in a fluid manner. The visualisation will show the user's personal information, which keeps getting larger over time. Therefore, PimVis needs to be scalable and be able to cope with a large amount of information items. Furthermore, the visualisation must support at least the standard file types such as PDF, JPEG, DOC and many more. Since all this information will be located on a centralised server, the PimVis interface needs some Internet connection. In the case that the Internet connection is lost, the system needs to stay functional.

4.3.2 Iteration 1: Graph Representation

Based on the defined requirements, the first prototypes were sketched. As mentioned in the previous section, the PimVis visualisation will use different layers in order to provide the right amount of data at once to the users. During the first iteration, the visualisation was composed of two main views. The first view has been called the *document view*, while the second one is called the *focus view*. The views can be seen in Figure 4.9 and Figure 4.10. The *document view* draft gives an overview of all of a user's documents represented as nodes of a graph visualisation, which is compatible with the graph-like structure of the OC2 framework. Each node represents a document, the node's shape depends on the type of the document. Documents can be seen in function of time via the time slider at the bottom of the page and can be filtered based on tags, context and type via the search box in the upper right corner. When searching for the keyword *WISE*, for example, all documents with the name, context, or tag *WISE* will be highlighted by putting their contour lines in bold and change their colour to amber, while the other documents will be faded out as illustrated in Figure 4.9. This interaction has been inspired by the Vizster visualisation [24]. The documents' relations to each other can be seen by the blue links. The darker the link, the stronger the relation between documents. When a document is selected, its contour colour becomes amber and an information panel appears on the right border of the screen containing information about the selected document. Note that Fortuna et al. [19] and Heer et al. [24] also used an information panel on the side of the screen to provide additional details about an element.

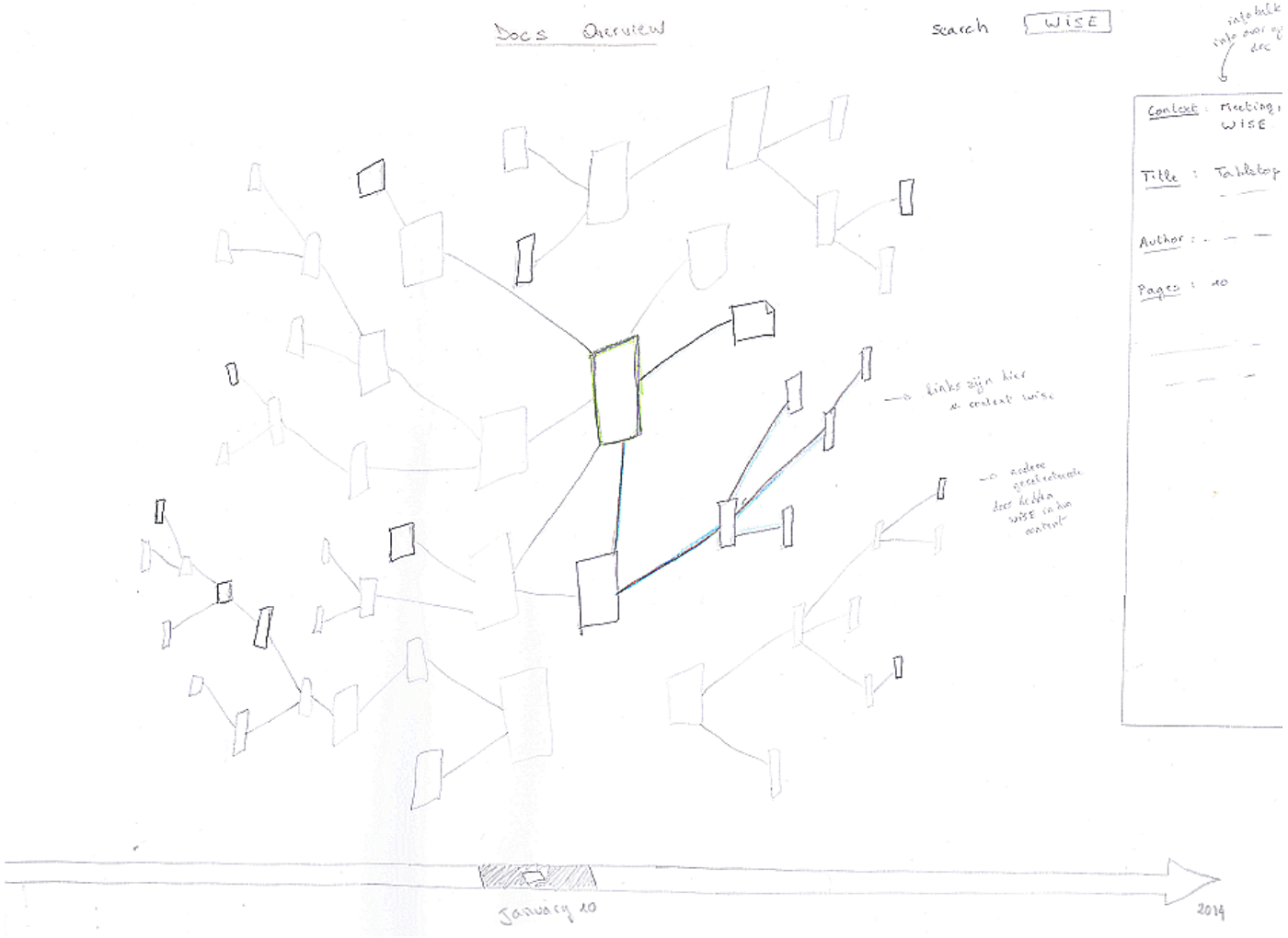

Figure 4.9: Document view draft 1 - with selected document in WISE context

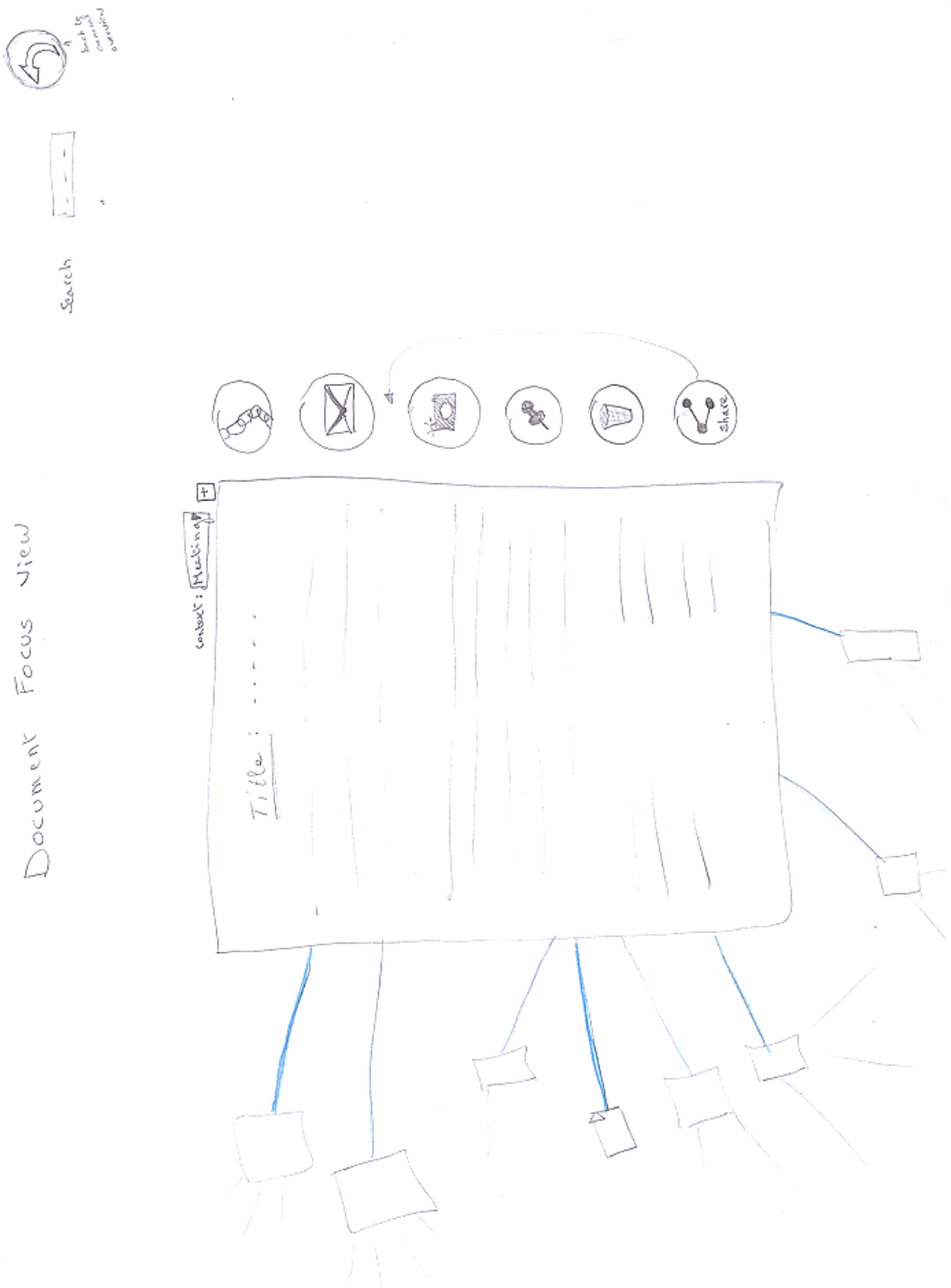

Figure 4.10: Focus view draft 1

4.3.3 Iteration 2: Bubble Representation

In the second iteration, an extra layer, called the *context view*, has been added to the visualisation. This layer is represented by means of the bubble layout. It has been added to get a better overview of all the user's contexts. Each bubble represents a different context. Since a user can have multiple sub-contexts within a context, we defined a hierarchy between contexts. This hierarchy is represented by bubbles inside a bubble, which is called a pack layout. A user can, for example, define a context *thesis* with as sub-contexts *thesis meeting* and *thesis presentation*. Allowing such structure can be convenient to classify related contexts together. A draft of the bubble visualisation can be found in Figure 4.11. On the figure we can see that one of the leaf bubbles contains a graph. The idea is that each leaf context has such graph, which represents all the documents within a certain context. When zooming into the bubble, the *document view* will appear, which shows the document of the zoomed-in context. To switch between the *context view* and the *document view*, a button in the upper right corner of both views has been added. Links between the bubbles show how many documents they have in common. Again, the darker the links, the more documents two contexts have in common.

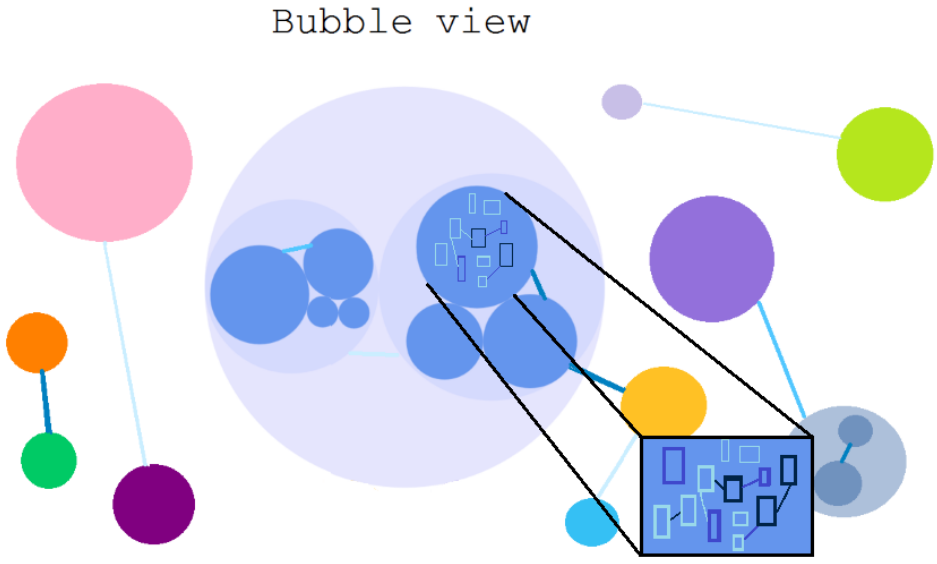

Figure 4.11: Context view draft 1

4.3.4 Iteration 3: Search Components

In the third iteration, a few new components have been added to the *document view* to improve the exploration of the document space. The tags of all the documents of a certain context are represented in a word cloud. The goal is that the user could select a tag, which will result in the highlighting of all documents containing the corresponding tag. The tag cloud can be seen in Figure 4.12 on the left. Next to this tag cloud, a context picker view has also been added, which shows all contexts of the documents in the graph representation. When selecting a context, all documents containing this context will be highlighted. This context picker view can be seen in Figure 4.12 on the right.

Figure 4.12: Left: Tag cloud, Right: Context Picker view

The last component that has been added, is the thumbnail dropdown. Each thumbnail represents a type of document, such as PDF, DOC or JPG. The goal of this component is to filter documents based on their type, which means that when a user taps on a thumbnail, all documents of the corresponding type will be highlighted. These thumbnails can be seen in Figure 4.13.

Figure 4.13: Thumbnail list. Images from windowsico.com

All three components are represented in an *accordion list*[2], so that users can choose to hide the components. Since the previous iteration, the time slider component has changed into a timeline brush component, which enables users to select a certain time interval on the timeline. The PimVis interface would then highlight all documents created in this time interval. This component can be seen in Figure 4.14.

Figure 4.14: Timeline

Additionally, the way to represent the documents in a graph has changed. The documents now all have a rectangular shape. Each rectangle contains a preview of the document and the document type can be seen in the bottom right corner of the rectangle. Furthermore, the importance degree or weight of the links and documents, is from now on also represented by the width of the line. The colour of the links and contour lines of documents has also changed to teal, which is a less flashy kind of blue. So, a document is more important when its contour lines are big and dark teal and less important when its contour lines are thin and light teal as illustrated in Figure 4.15. The same reasoning is used for the thickness and colour of the links.

[2]https://jqueryui.com/accordion/

Figure 4.15: Document importance degree representation, left: important document, right: less important document

4.3.5 Iteration 4: Link Deletion

During the fourth iteration, the *context view* has been changed. The decision has been made to delete the links between the bubbles since these links had another meaning than the links between the documents in the graph visualisation, which could confuse the user. Links are used in the *context view* to show how many documents the contexts have in common, while the links in the *document view* show the relation between two documents. Moreover, using links in the pack layout defeats the purpose of such a layout, which has the main purpose of giving a compact overview to the users about their contexts. Putting links between these bubbles could make the visualisation quite cluttered and unclear. The new *context view* is illustrated in Figure 4.16.

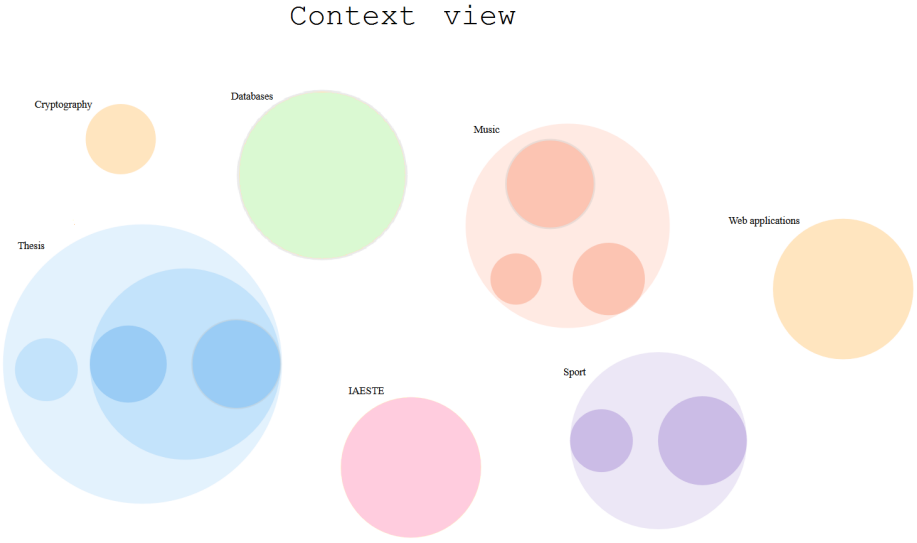

Figure 4.16: Context view draft 2 - bubble representation without links

4.3.6 Iteration 5: Embellishing the Interface

In the fifth iteration, the focus has been set on making the interface more attractive for users. As can be found in the style guide in Appendix A, *teal* —some variant of blue— has been chosen as the main colour for the visualisation. In order to remain consistent, the same colour has been used for the titles, buttons and all the different components as well as for the control panel.

Figure 4.17: From left to right: the re-finding, control panel, *document view* and *context view* button.

The layout of the buttons that are used to navigate through the different views are shown in Figure 4.17. The names of the components in the control panel also changed to be more understandable for the user. The context picker became the *context explorer*, the thumbnail dropdown became the *type filter* and finally, the node information became *details*. The new layout of these four main components is shown in Figure 4.18. A general overview of the *document view* is illustrated in Figure 4.19.

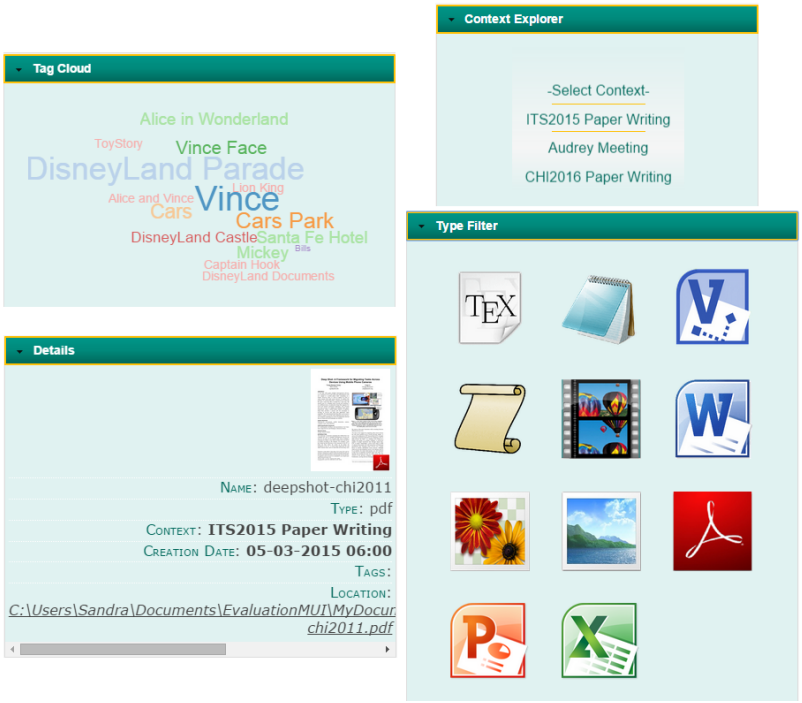

Figure 4.18: The four main components of the control panel. Top left the Tag Cloud, beneath it, the Details of a document called *deepshot-chi2011*. On the right top corner, the Context Explorer to filter based on context, beneath it the Type Filter to filter based on type.

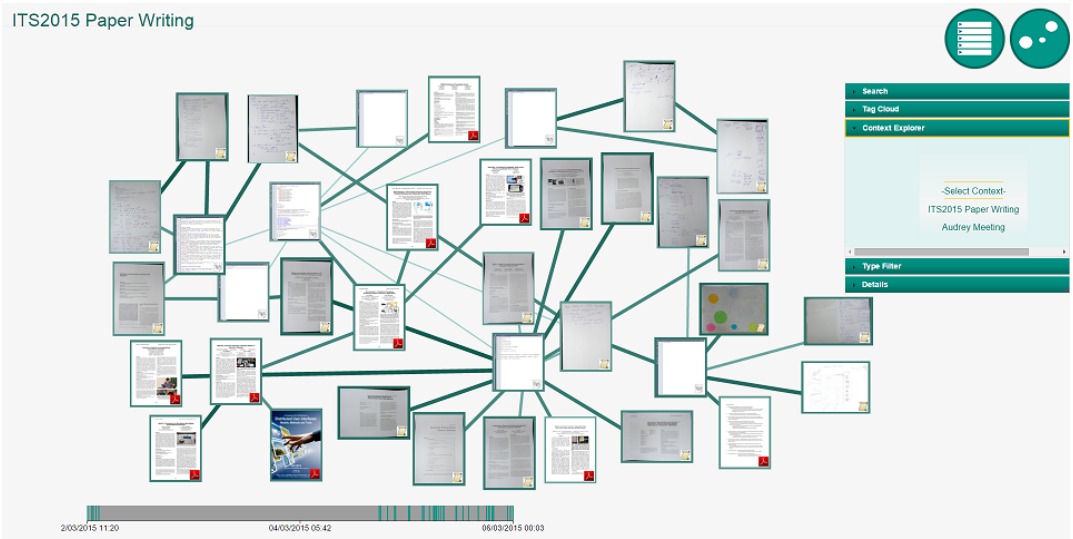

Figure 4.19: Document view - new layout

4.3.7 Iteration 6: Cross-Device Interaction

In order to re-find a document in the user's file system or bookcase, depending on whether the document is a digital or a physical one, a *re-find* button has been added to the PimVis interface. The *re-find* button is only accessible from the *focus view*. When users press this button when a digital document is selected, the desktop computer will open the folder containing the corresponding document and select it. If the selected document is a physical one, the LED located in the ring binder containing the right document will light up, indicating the document's location in the bookcase. Besides the cross-device interaction, a new component has also been added to the *document view*, namely the zoom slider. From this iteration on, the user can choose to zoom with the zoom slider or with the double tap gesture on a document. The zoom slider is shown in Figure 4.20.

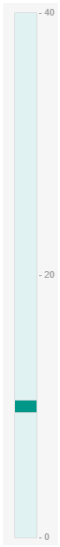

Figure 4.20: Document view - zoom slider

4.3.8 Final Iteration

Based on the user's feedback during the conducted user study, some minor modifications were made during this last iteration. Many users did not think about using the *re-find* button, when it came to find a document located in the file system. A few users just typed over the location of the document in the file explorer of the desktop computer. That is why, during this iteration the location description of the details component was made clickable, allowing users to simply click on it as if it was a URL leading to a website. To remain

consistent, we also made the location of a physical document clickable. The users can now use two ways to re-find documents in their file system and ring binders.

4.4 Touch Gestures

Since the PimVis interface has been created to work on a multi-touch table-top, a few gestures that can be used to interact with the visualisation were selected. An overview of the gestures can be seen in Table 4.1, basic touch gestures have been chosen so that they could easily be remembered by the users. In this section, the function of each of these gestures in the PimVis interface will be explained.

Image	Movement	Description
	Tap	Tap with one finger on the surface
	Double tap	Tap twice with one finger on the surface
	Drag	Press one finger on the surface, move it around the surface and then release
	Press and hold	Press and hold two fingers on the surface for about one second

Table 4.1: Supported touch events, created with one or two finger input. Images from gestureworks.com

Tap

The tap gesture is mostly used for navigation and exploration interactions. When tapping on the *document view* and *context view* button, the user can go to these two views. Exploring the documents of the *document view* can be done by tapping on some document, as result the details about the document will be shown in the control panel. By tapping on some tag in the *tag cloud* or

on a thumbnail in the *type filter*, further exploration can be done. Depending on what has been tapped on, the documents having this property will be highlighted. The tap gesture can also be used in the *context view* to quickly get a glimpse at the name of the context represented by a bubble.

Double tap

The double tap gesture can be used to zoom into the current visualisation. This gesture can only be performed on bubbles or document nodes. This technique has been chosen because the zooming level has also been used to let users switch between the *context*, *document* and *focus view* depending on which element they zoom into. For example, zooming twice into a document in the *document view* will bring the user to the *focus view* with the corresponding document in the centre of the screen.

Press and hold

When a user presses on an element with two fingers and holds for at least one second, the user can zoom out in the current visualisation. Again, this gesture can only be performed on some element, like a bubble or a document node. Depending on the zooming level, this gesture also allows the users to navigate through different views. For example, if the gesture is used on a document in the *focus view*, it will bring users to the *document view*.

Drag

The drag gesture can be used to move document nodes to another position on the screen. Moreover, the drag movement can be used to select a time period on the brush component in the *document view*. As a result of this interaction, the documents that were created in this time period will be highlighted.

4.5 Implementation

In this section, the overall implementation of the PimVis interface is reviewed. The interface has been implemented in JavaScript. The different types of visualisations used in the views of Pimvis are made using the D3 library[3], which will be introduced in the next subsection. In the following subsections, the implementation of the different components of the visualisation is presented.

[3]http://d3js.org/

4.5.1 D3

As mentioned earlier, the different visualisation types are mainly constructed using the D3 library. D3 is a Data-Driven Document-based JavaScript library, which allows to bind data to a Document Object Model (DOM). It provides different data-driven visualisations that can be applied to this DOM. For example, given a dataset, D3 can generate an interactive choropleth map or produce some dynamic line charts or even other visualisations all using the same dataset. Based on the requirements, sketches and the available D3 layouts, four different D3 layouts were used. The first one is the pack layout, which is used to show a hierarchy using bubble packing. This is used to build the *context view*. To implement the *document view*, the D3 force layout has been used. Next to that, the brush SVG Control element was employed to implement the timeline component. Finally, the tag cloud has been implemented using a D3 cloud layout[4] library, which is based on the D3 force layout. To integrate multi-touch events, *d3.touches* has been used.

4.5.2 Context View

The *context view* shows all of a user's contexts. Each context is represented by a bubble. A context can have sub-contexts, for example, if the user has a context *meeting*, this context can have sub-contexts *student meeting* and *department meeting*, which in turn may have some other sub-contexts. To visualise such a data structure, the D3 pack layout is used, which is very convenient to show hierarchical data using circle or bubbles packed in a certain manner. The pack layout implementation can be seen in Listing 4.1. This code snippet creates a new pack layout with several specific settings, such as the `size` of the pack layout, the `padding` between adjacent bubbles, the `value` that determines the size of each bubble depending on the number of documents in the bubble's context and the final parameter is the `nodes` which holds the data of the pack layout. This data has typical tree structure.

```
pack = d3.layout.pack()
        .size([width, height])
        .padding(50)
        .value(function(d) { return d.size; })
        .nodes(data);
```

Listing 4.1: JavaScript pack layout initialisation

The *context view* is not a simple pack layout visualisation. Each leaf bubble or context of the view shows a preview of the documents inside it. When the bubbles are drawn in an SVG container, the distinction needs to

[4] https://github.com/jasondavies/d3-cloud

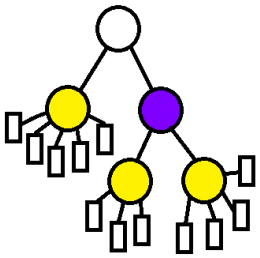

Figure 4.21: Data example JSON structure

be made between the leaf contexts and the other ones. This is done as shown in Listing 4.2. Since the data given to the pack layout contains the contexts as well as the documents within each of these contexts, there is no need to go down to the leaf nodes of the JSON structure, as they are documents and not contexts. Consider a JSON structure as shown in Figure 4.21, where contexts are represented as circles and documents as rectangles. To find the leaf contexts of the JSON structure, we need to first verify if the node has children. If this is the case, we verify whether the node's child has any children. If it does not have children, like the yellow circles on Figure 4.21, it represents a leaf context. In order to draw a preview of the documents inside such a leaf context, the function `drawPreview` is called. If the node's child does have children, it is an internal node, like the purple one, so only the corresponding circle hierarchy needs to be drawn using the `drawCircle` function. Note that the root node has been removed from the data, since no *upper* context needs to be represented.

```
container.datum(data).selectAll(".node")
        .data((nodes).filter(function(d) {
                    return !(d == root); //filter out root node
         }))
        .enter()
        .append("g") // g is a circle element
        .attr("class", "node")
        .attr("id", function(d) { return d.id; }) //to find by id
        .each(function(d) {
            if (!d.children) //if node has no children, it is a doc not a
                context
              return;
            if (d.children[0].children) { // if child has a child then it is
                 a middle node
              drawCircle(d3.select(this));
            }else{
              drawPreview(d3.select(this));
            }
        });
```

Listing 4.2: JavaScript used to distinguish internal and leaf contexts

4.5.3 Document View

The *document view* contains five different components. The main component is the graph visualisation. Next to this, there are the tag cloud, context explorer, type filter, brush timeline and the zoom slider component. In this section we will explain how these different components are implemented.

Graph Visualisation

As mentioned earlier, the main component of the *document view* is the graph visualisation, which is implemented via the D3 force layout. The graph visualisation shows all documents of a specific context. The initialisation of this force layout is done as shown in Listing 4.3. Concretely, various parameters are added to the layout in order to initialise the graph. These parameters are the `size` of the graph, the `nodes`, the `links`, `link distance`, `charge` and `tick` function. The charge of the nodes in a graph indicates the nodes attraction or repulsion relative to each other. We put the charge on a negative value (-900), so that nodes will repulse each other. The tick function is used to initialise the position of the nodes. It is run once the `start()` is called.

```
force = d3.layout.force()
        .size([width, height])
        .nodes(nodes)
        .links(links)
        .linkDistance(360)
        .charge(-900)
        .on("tick", tick)
        .start();
```

Listing 4.3: JavaScript force layout initialisation

Tag cloud

The tag cloud component has been implemented by making use of the *d3.layout.cloud.js* library which is a library that requires D3. This library can be used to construct a tag cloud as shown in Listing 4.4. The `words` parameter takes an array of elements that have a tag name and a frequency. Depending on the frequency of a certain name, the size of the tag will be bigger or smaller in the tag cloud. This can be interpreted by the parameter `fontSize`. The `rotate` parameter is set to 0, which indicates that the words of the cloud will not be rotated and thus will be shown horizontally. There is one event attached to the cloud, called *"end"*, which is executed when all tags are placed. When this happens, the `draw` function is called which *draws*

the different tag elements inside an SVG container. Inside this function the colour, position and events coupled to each word or tag are defined.

```
cloud = d3.layout.cloud()
        .size([widthCloud, heightCloud])
        .words(tags)
        .rotate(0)
        .fontSize(function(d) { return wordScale(d.frequency); })
        .on("end", draw)
        .start();
```

Listing 4.4: JavaScript cloud layout initialisation

Context Explorer

The context explorer is based on the spinning-wheel or slot-machine metaphor. To achieve such a picker view layout, the mobiscroll library[5] shown in Listing 4.5 has been used. First, a context list in the right HTML form using *option* tags is built, then the mobiscroll library is used to change the layout of this list to a *picker view*. Using CSS, the layout has been changed to the defined colour scheme. This component is shown in Figure 4.18, which has been presented earlier in this chapter.

```
// make html options' tags for each context in context Array
  for(var i = 0; i < contextsMap.length; i++) {
        var opt = contextsMap[i];
        var el = document.createElement("option");
        el.textContent = opt;
        el.value = i;
        selecty.appendChild(el);
    }

    $('#contextsPicker').mobiscroll().select({
                    theme: 'android-holo-light',
                    display: 'inline'
                });
```

Listing 4.5: JavaScript context picker layout

Type Filter

The third component of the *document view* is the type filter. The implementation of the type filter is based on this JavaScript snippet[6], which has been used to create a dynamic table. This means that when the table's dimensions change, the elements inside the table are rearranged to fit the new dimensions. The table has been filled with thumbnails representing the different

[5] https://github.com/acidb/mobiscroll
[6] http://stackoverflow.com/questions/17662698/filling-a-table-with-javascript

types of documents. Each picture or thumbnail is associated with a click function that highlights the documents of a certain type when clicked on.

Brush Timeline

Another component of the *document view* is the brush timeline, which is implemented via the D3 brush SVG Control. The initialisation of the brush component can be seen in Listing 4.6. The brush's x-scale depends on the creation date of all documents that can currently be visualised in the graph. Thus, the minimal x-value will be the earliest creation date while the maximal x-value will be the latest creation date of a document. Once this is set, the brush can be put inside a group element situated in the SVG container.

```
var brush = d3.svg.brush();
brush.x(scale);
//draw brush into the g element
brush(g);
```

Listing 4.6: JavaScript brush component initialisation

4.5.4 Zoom Slider

The last component of the *document view* is the zoom slider. To implement this component, we used the jQuery Slider Pips[7] library. Listing 4.7 shows the initialisation of this component. The orientation has been set to vertical and the minimum and maximum values of the slider has been currently set on 0 and 40. This configuration has been chosen based on how much zooming a user could use, while keeping a clear view of one or more documents.

```
//  Build a vertical slider for zooming
  //  Min value is 0 and max value is 40

  $(".slider").slider({
     min: 0,
     value: 10,
     max: 40,
     orientation: "vertical"
 })
  .slider("pips", {
     rest: "label",
     step: "20"
 });
```

Listing 4.7: JavaScript zoom component initialisation

[7] http://simeydotme.github.io/jQuery-ui-Slider-Pips/

4.5.5 Focus View

The *focus view* shows a zoomed-in view of a specific document surrounded by all its related documents, even the ones from other contexts. This view has been implemented in the same manner as the graph view, thus using the D3 force layout. The main difference is that the position of document located in the middle is fixed. Once in the *focus view*, the user can start locating the document in the physical or digital space depending on the type of the document. This is done via the *re-find* button, which detailed functionality is explained in Section 5.5. Unlike in the *document view*, in the *focus view* the control panel is only used to show details about the selected document, the exploration components are not available.

4.6 Conclusion

The PimVis user interface has been built to manage the user's personal information space. It is based on two principles. The first one is the principle that associations between documents are used to better remember them [33], which is shown in PimVis by links between documents. The second principle is that contextual cues are often used for re-finding documents [58]. Therefore, we present a view of all the user's contexts. The PimVis interface uses different views and display metaphors, in order to give the users the right amount of information at once on the screen. The first view is the *context view*, where all context information is shown. The second layer is the *document view*, where all the documents of a specific context can be seen. The last view is the *focus view* which shows more details about a specific document. The contexts are represented by bubbles packed within other bubbles. This is since a context can have different sub-contexts. The documents are represented by rectangles containing a preview of the document together with an indication of its type. These rectangles are nodes of a graph visualisation, which is used to represent the relation between the documents. By using the OC2 framework, the interface has been built following an object-oriented design approach. This means that every element of the information space is represented as an object with certain attributes.

In order to populate the PimVis interface, the OC2 server has been used, which contains all the personal information of the users. More about the use of the OC2 framework will be explained in Section 5.3. The final version of the different views of the PimVis interface is illustrated in Figure 4.22.

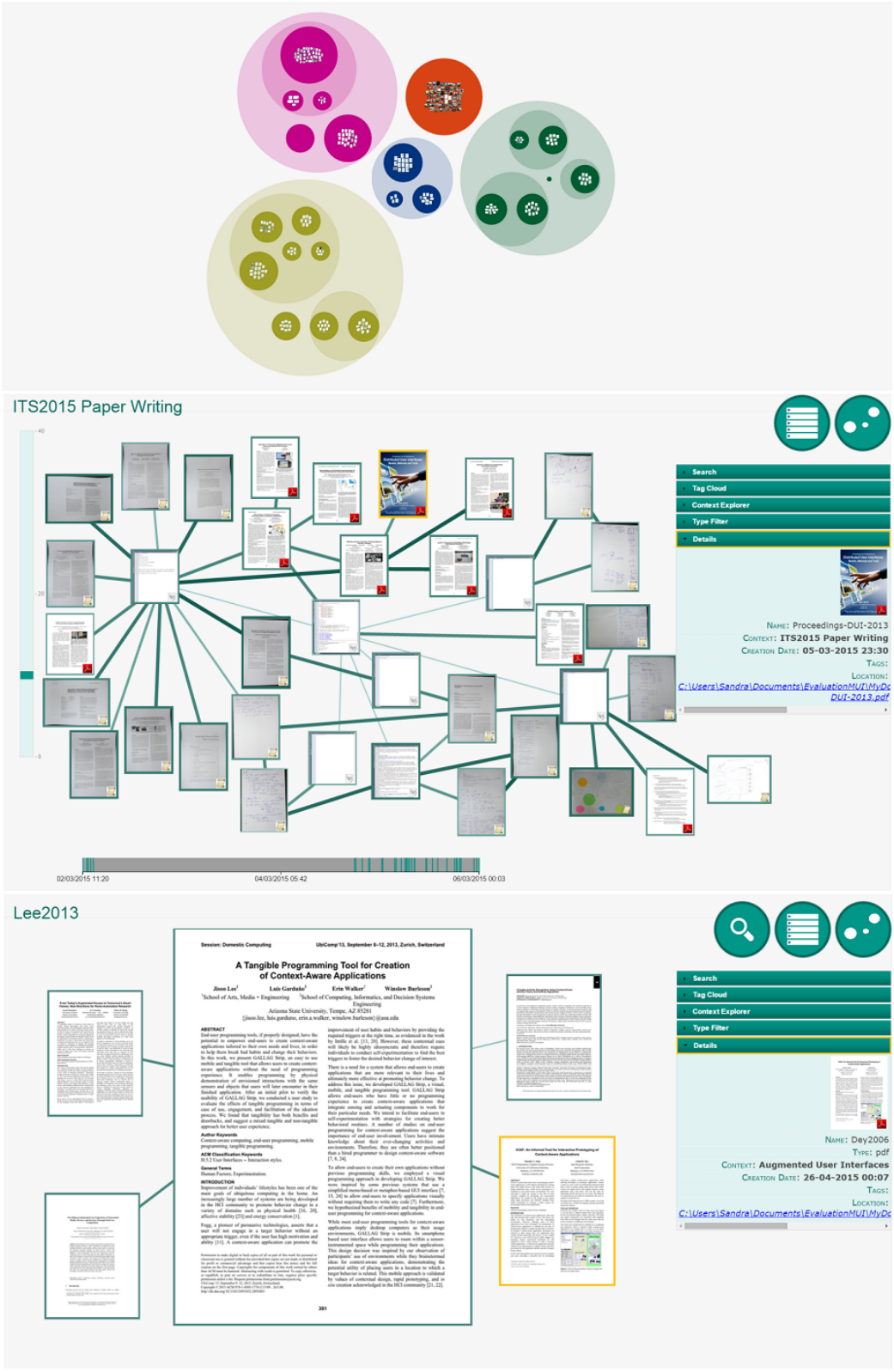

Figure 4.22: From above to the botom, the *context view*, *document view* and *focus view*

5

The eXtensible User Interface Framework

The research that we reviewed in the background chapter points out the importance and usefulness of cross-device and cross-media interaction. The emergence of cross-device interaction offers a way to solve some problems encountered from disconnected devices. Cross-media interaction can be used to bridge the gap between the physical and digital space. We investigate fluid interaction spaces in an Office of the Future environment, where desktops, tabletops as well as digitally augmented bookcases form a part of the user's natural workflow. We present the eXtensible User Interface (XUI) framework which can be used to develop such fluid interaction spaces. In this chapter, the components of the XUI framework will be explained. First, the requirements are introduced. After that the architecture is discussed. Finally, the implementation of the different components of the framework are examined.

5.1 Requirements

Based on the scenarios discussed in Chapter 3, we present three main requirements for achieving a fluid cross-device and cross-media interaction space. These are the use object-oriented user interfaces, lightweight data exchange and ad-hoc interaction configuration and will be explained in this section.

Requirement 1. Object-Oriented User Interfaces

A user interface should be seen as a possible *view* of the user's personal information space. The natural interaction a user has with personal documents, such as exploration, manipulation, storage and retrieval of information, is an important part of the user's workflow that must often be achieved by using multiple applications. By making the information independent of the application it needs to be used in, documents can be accessed from any application. Properties can then be assigned to documents, which can be used in interfaces of different devices and media types. Documents might, for example, have some location property, which either indicates with a URI where a digital document is located in the filesystem or indicates with a semantic label the position of physical document in the bookcase. The property can then be accessed by any user interface of the Office of the Future setting.

Requirement 2. Lightweight Data Exchange

An Office of the Future environment tends to contain ubiquitous light-weight user interfaces. An example of this would be a Windows File Explorer extension and LED-based interfaces, which can be used to pass data across interfaces. This data does not always need to contain the entire application state, but only parameters expected by the interface to which it is sent to. If a user needs information about some specific document on the tabletop, it is sufficient to just send the ID of the document to the appropriate interface. The same goes for the ring binders application, which should only ask the XUI framework the number of the LED that needs to be light up, depending on the physical document that needs to be retrieved.

Requirement 3. Ad-hoc Interaction Configuration

Due to the increase of augmented office supplies, it is important to have an easy and fast mechanism to add new furniture. Designers also need a way to explore the available user interfaces of the office environment. This way, they are able to provide a fluid interaction between them and integrate new functionality in their programming code. Users often like to be able to reconfigure their office environment without too much effort. This dynamic character of the office setting, entailing temporary disconnection, relocation and the addition of devices, is only possible with no hardcoded references between the user interfaces. That is why the setting must be generic.

5.2 Architecture

In this section we present the architecture of the Office of the Future setting. An illustration of this architecture can be found in Figure 5.1. The two main components are the *XUI framework* and the *User Interface Management Web (UIMW) service*. Both are implemented as a GlassFish server and use the Atmosphere framework to enable communication via different protocols. The latter is done by using the Jersey JAX-RS reference implementation framework for REST interfaces. While the XUI server is a local server, the UIMW service runs as an online server.

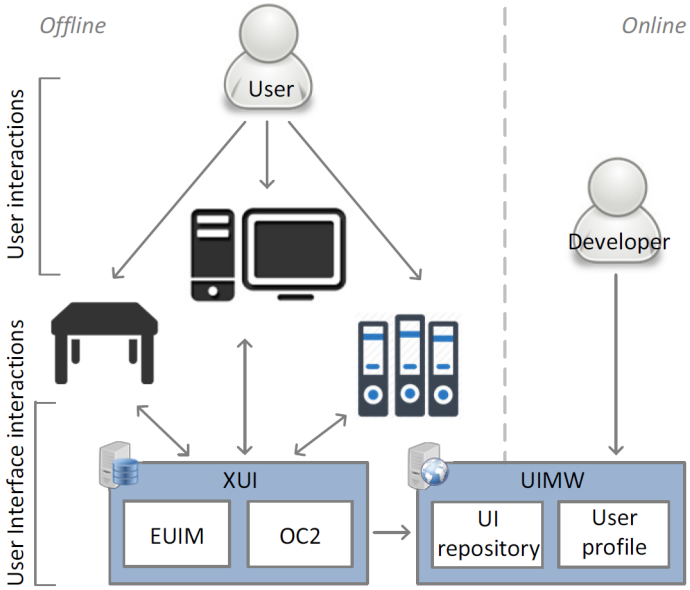

Figure 5.1: The XUI architecture

The XUI server ensures the communication between the available user interfaces. These currently are the tabletop interface, the computer's file system interface and the ring binders interface. The XUI framework is implemented following the client-server model and has to be installed on the user's local network. It includes the Event-based User Interface Mechanism (EUIM) as well as the Object-Concept-Context (OC2) Personal Information Management system. The EUIM server is a publish/subscribe mechanism for events, while the OC2 framework, on the other hand, manages all the user's personal documents, the links between them and their properties.

The UIMW service, makes it possible for developers to explore the functionality of the interfaces. By providing this service, developers have the opportunity to integrate the existing interfaces to their new user interface.

Additionally, the Web Service keeps track of the different user interfaces in a user's office setting by means of user profiles. These profiles can be quite convenient if a user, for example, wants to extend the Office of the Future setting with some new artefacts. Before buying the new user interface, the user can then verify whether or not it is compatible with the current setting.

To better understand these different components we go deeper into each one of them in the following sections.

5.3 XUI Server

As mentioned before, the XUI offline server communicates and passes data to the various user interfaces in a user's setup. There are two types of interactions: the *user interaction* and the *user interface interaction*. The first one comprises the different kind of interactions that the user can have with the user interfaces. The user can use the tabletop, the extended file system and augmented ring binders application. These artefacts in the user's office are all listening to the XUI server. When a user interacts with the tabletop, for example, by pressing the *re-find* button, they trigger an interaction between the PimVis interface and the XUI server. This last interaction is part of the *user interface interactions*. If the *re-find* button was used on a digital document, an interaction with the extended file system in order to show the corresponding document on the desktop computer will be triggered. On the other hand, if the document is a physical document, an interaction with the ring binders application is triggered to light up the LED of the corresponding ring binder, by giving the application the correct information. The file system can also trigger an interaction with the XUI server. This happens, when users select the *Open in PimVis* in the context menu of the File Explorer. As result, the PimVis interface will get the correct information from the XUI server, allowing PimVis to show the right document in its *focus view*. Finally, the ring binders application can trigger an interaction when users place a sheet of paper from one of these ring binders under the camera mounted on the desktop, the detection of the fiducial marker on the paper will trigger an interaction with the XUI server. Consequently, the PimVis interface will again receive the needed data to visualise the document in the *focus view* from the XUI server. These interactions are illustrated in Figure 5.1.

When data is passed around between these different user interfaces, the event mechanism of the EUIM component is used. This data containing information about the documents of the user, is stored in the OC2 component of the XUI server. The components of the XUI server will be discussed in the following sections.

5.3.1 OC2

In order to follow an object oriented user interface (OOUI) design approach, we used the Object-Concept-Context (OC2) framework [59]. The OC2 framework has been implemented on top of a db4o object database and developed specially for the PIM domain. It is a domain-specific extension of the Resource-Selector-Link (RSL) metamodel, which has been introduced by Signer and Norrie for hypermedia systems [49]. The model was created using the semantic, object-oriented data model, called the OM model [41], which combines the concepts of ER and OO-data models. Therefore, it offers the concept of a collection of objects and associations between them. The RSL metamodel's core components are shown in light grey in Figure 5.2, while the extensions are coloured in blue. Rectangles represent collections, whereas the oval shapes are the associations between collections. Each association also has cardinality constraints.

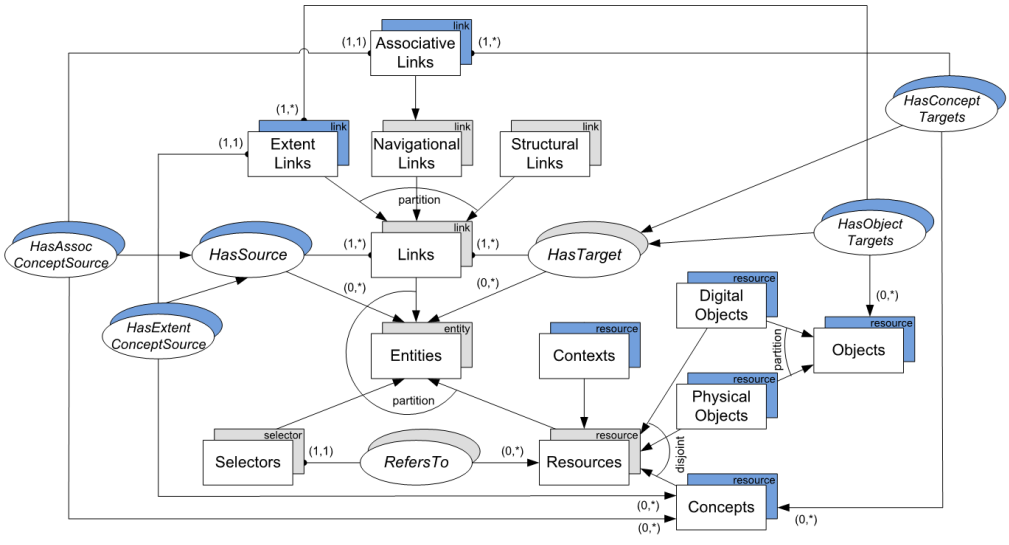

Figure 5.2: The OC2 metamodel

A collection can contain three types of elements, which are `Resources`, `Selectors`, and `Links`. All of them are subtypes of the `Entities` collection.

The `Resources` type represents a complete information unit. Since this resource is an abstract concept, a plug-in mechanism has been provided in order to add a specific extension to the `Resource` type. Hence, when different kinds of resources, such as music, web pages or even physical paper, needs to be managed, they are instantiated and added to the `Resource` collection. Rather than linking an entire resource, specific pieces of a resource can also be linked by using a `Selector`. A `Selector` is always associated with precisely one resource via the association `RefersTo`. The last subtype of the `Entities` collection is the `Links` collection. `Links` are always directed and have one or more sources and targets. Since the `HasSource` and `HasTarget` associations are a part of the entity domain, links can be defined between the three `Entities` subtypes. In the original RSL model links are partitioned into two types, the `navigational` and `structural links`. The first one is used to show navigation possibilities between the different entities. A `structural link` can be used to define the structure of any entity of the `Entities` collection. More details of the RSL model are described in [49].

The OC2 framework added three main extensions to the RSL model. The first one extends the `Resource` collection with `Digital Objects`, `Physical Objects` and `Concept` subcollections to make the distinction between objects and concepts. This distinction is important since users perceive concepts as internal to their mind, while objects are external to their mind [8]. Objects are also subdivided into physical and digital objects. This is due to the fact that a user's memory often uses this additional information as refinding cue. A resource can only belong to one of these three subcollections. This can be inferred from the disjoint constraint between them as shown in Figure 5.2. Moreover, the `Contexts` collection is also added as subcollection of the `Resources` collection. Contexts can be linked to other contexts and to various entities. The OC2 framework also introduces two new link types, the `Extent` and `Associative Links`. The `Extent Links` are a direct subcollection of the `Links` collection, while the `Associative Links` are a certain type of `Navigational Links`. `Extent Links` are used to link `Objects` to a semantic concept. This is achieved by the `HasExtTarget` and `HasExtSource` associations. The `HasExtTarget` represents the objects, while `HasExtSource` indicates the concepts of the extent link. `Associative Links` are used to represent navigation between `Concepts`. Similar to the `Extent Links`, this is done by using the `HasAssocTarget` and `HasAssocSource` associations. Both associations are representing specific contexts of the associative link.

The OC2 framework has mainly been used to get an OOUI design, but it also has another added value, namely supporting the context-awareness of its objects. Objects are often used by the users in a specific context. When

an object is lost, users tend to remember the context in which they used it. By allowing users to classify objects by their contexts, it is easier for them to re-find their location. An object can have a different weight depending on its context. For example, a certain *PDF document* might be more important in a context *meeting* and less important in the context *paper writing*. Besides `Objects`, all elements from a subcollection of the `Entities` collection can have some weight depending on a context. This context relevancy is modelled via the `InContext` association and is illustrated in Figure 5.3.

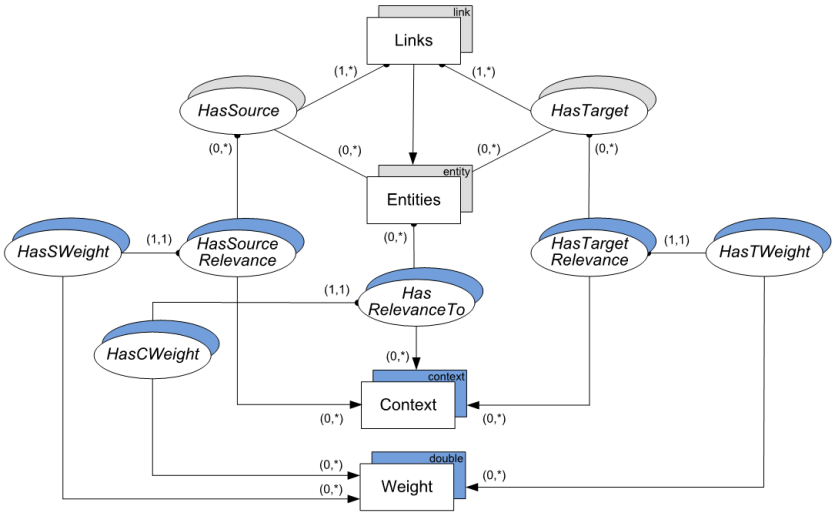

Figure 5.3: Context elements in the RSL framework

The weight of an entity within a context is defined by the association `InContext` and the `Weights` collection via the `HascontextWeight` association. Since links are elements of a subcollection of the `Entities` collection, they can be assigned with a weight depending on their context. However, links in the RSL metamodel are many-to-many links, which means that only this link can have a weight. To support context relevancy for each association between the target and source element, the `HasSourceRelevance` and `HasTargetRelevance` corresponding to `HasSourceWeight` and `HasTargetWeight` has been added to the model. This makes it possible to define a weight for each source and target of a link given a specific context.

Such entity properties can be added by developers for their specific user interface. This can be done by using the RESTfull API of the OC2 framework. Some of the most used REST endpoints are shown in Listing 5.1. The first two lines are used to add additional properties to objects. The next endpoint is used to get some property of an object, such as its weight.

The one after that is used to get the linked documents of an object, while the last one is used to get all documents of some specific context.

```
../property/getNewProperty/{key}/{value}
../object/addProperty/{objectId}/{propertyId}
../object/getPropertyWithKey/{objectId}/{key}
../object/getAllNavigationalLinks/{objectId}
../context/getAllRelatedObjects/{contextId}
```

Listing 5.1: Most used REST endpoints

5.3.2 EUIM

The Event-based User Interface Mechanism (EUIM) allows different user interfaces to publish events, as well as to subscribe to events published by other user interfaces. The EUIM component also includes a part responsible for the communication with the UIMW online server, in order to share the user's interface configurations. Researcher Alice, for example, has three user interfaces configured in her Office of the Future setup. The three interfaces are the PimVis tabletop, the extended file system and the ring binders interface. When she interacts with them, events are published, which can be intercepted by interfaces subscribed to those events. Through this event mechanism of the EUIM component, data is passed around between interfaces. This data can be the entire application state or just some part of OC2-related information about digital or physical objects, concepts and contexts. If a document needs to be re-finded, passing the document's ID would suffice.

To implement the EUIM component the publish-subscribe pattern has been used, which guarantees lose coupling between publishers and subscribers, since the publishers do not even need to know the existence of their subscribers. This pattern ensures a better scalability of the system, for example, through parallel operations. Each user interface can define its own channel on which it will publish specific data. User interfaces can even publish to existing channels. To notify all user interfaces subscribed to a certain channel whenever data is published, a broadcasting mechanism is used. The data received on a channel will be broadcasted to all subscribed user interfaces of that channel.

Imagine that Alice wants to re-find a certain document stored in one of her ring binders. The document is currently opened in the *focus view* of her PimVis interface. By pressing the *re-find* button, the right information is published on the *selectInSpace* channel. This information is similar to the data shown in Listing 5.2.

```
1  {"sender":"PimVis",
2   "objectId": 29,
3   "state": "selected"
4  }
```

Listing 5.2: Published data by PimVis

Any user interface subscribed to this channel can use the `objectId` and query OC2 if extra information is needed, such as a certain property of the document or the documents related to it. In Alice's current Office of the Future setting, only the ring binders interface is subscribed to the *selectInSpace* channel. So when such an event occurs, the ring binders application will receive this data and query the OC2 framework to know the led property of the object. The latter is done via the endpoint shown on line 3 of Listing 5.1. The endpoint will return the number of the LED that has to be lighted up. As result, the ring binders application will light up the LED of the ring binder containing the document. This interaction scenario is illustrated by the sequence diagram in Figure 5.4.

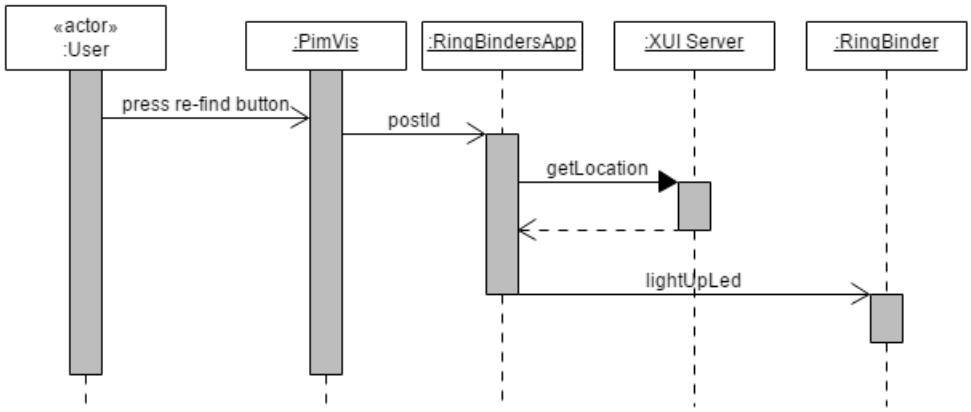

Figure 5.4: Sequence Diagram representing the interaction between the user and the PimVis interface, in order to find a physical document

If instead of a physical document, Alice needs to find a digital document in her file system, she can again just press the *re-find* of the PimVis interface, given that she opened this document in the *focus view*. Internally, something else happens than in the previous case. The PimVis interface will publish on the *selectInFS* channel instead of the *selectInSpace* channel. Consequently, the extended file system, which is listening to that channel receives an event notification.

Depending on the published data, the extended file system will open the File Explorer on Alice's computer in the folder containing the document she needed. This scenario is shown in the sequence diagram in Figure 5.5.

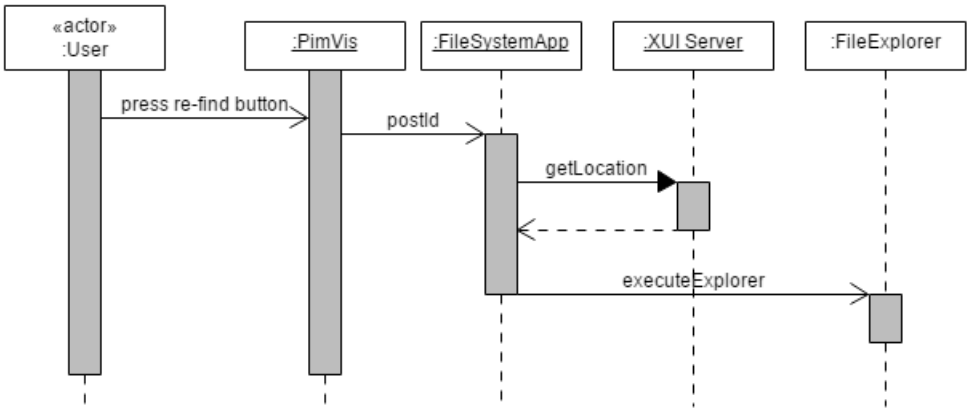

Figure 5.5: Sequence Diagram representing the interaction between the user and the PimVis interface, in order to find a digital document

Due to the use of an OOUI design approach, the Pimvis application does not need to send all the document's information to its subscribers. In this case, only the document ID suffices. This has the major advantage that small and simple user interfaces, such as the ring binders interface, do not need to process a large amount of data, which leads to a better performance and easier development.

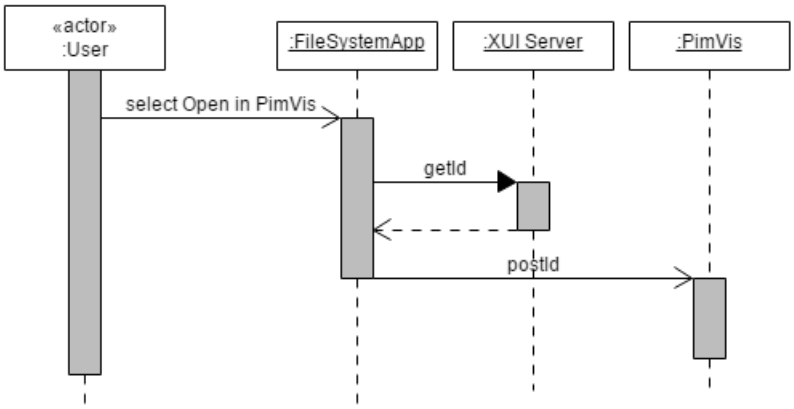

Figure 5.6: Sequence Diagram representing the interaction between the user and the file system application, in order to open a document in PimVis

When Alice interacts with the extended file system and selects the option *Open in PimVis* in her File Explorer in order to open this document in PimVis, the file system interface will publish data on the *openPimVis* channel. Since the PimVis interface is listening to this channel, it will get the required information to be able to open the document in its *focus view*. This interaction is illustrated by the sequence diagram in Figure 5.6.

If Alice had tried to show a physical document in the PimVis interface by using the augmented ring binders, the PimVis interface would have received the same data structure. This because the ring binders interface is also publishing on the *openPimVis* channel. This interaction scenario is shown by a sequence diagram in Figure 5.7.

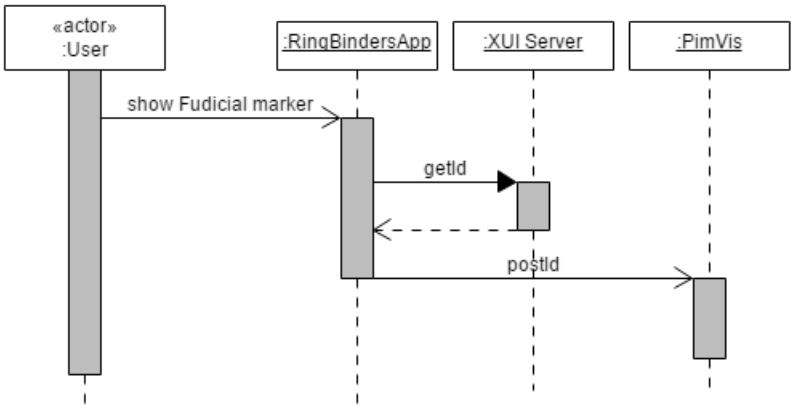

Figure 5.7: Sequence Diagram representing the interaction between the user and the ring binders application, in order to open a document in PimVis

Thus, in order to receive data from a specific channel, the user interface needs to subscribe to this channel. A user interface can be subscribed to different channels. When needed, user interfaces can define their own channel to publish data on, instead of publishing on existing channel. Developers must specify these channels via the UIMW service and the JSON format in which information will be encoded must be defined. This is required in order to allow a flexible Office of the Future environment where any compatible interface can be added.

In the current setup, data is published through the XUI REST interface. Subscribing to channels in order to receive events is done by establishing a WebSocket connection. To manage this connection, the Atmosphere framework is used, making it also possible for clients to establish long polling or HTML Server-Sent Events next to the WebSocket connection. Furthermore, a fallback mechanism is included in the case that some connection type would not be available.

5.4 UIMW Server

To extend or modify the Office of the Future setting, developers will need to know which third-party user interfaces are already integrated to the XUI framework. The User Interface Management Web Service (UIMW) will ensure that they get all the necessary details about these different user interfaces. Those comprise the publishing channels, the data they send and to which channels they are subscribed. Again, this component is based on a REST interface, which integrates the Atmosphere framework to deal with the client-server communication. The UIMW Web Service contains two important components, the UI Repository and the User Profile component. In the remaining of this section the function of each these components will be explained.

5.4.1 UI Repository

The user interface repository, keeps track of all user interfaces of the fluid interaction space. When a new user interface needs to be used, it must first be registered with the UIMW service. The registration is done via the POST `http://.../uimw/register` endpoint of the UIMW REST API. This endpoint expects a JSON body including the general user interface information together with the publishing and subscribing channels that will be used. An example of an interface registration can be seen in Listing 5.3, which is the registration of the PimVis interface.

```
1  { "name":"PimVis",
2    "description": "visualisation for tabletop",
3    "pubChannels":[{"name":"showInFileSystem"}, ...]
4    "newChannels": [
5    { "purpose":"indicate the location of a physical document",
6      "channelName":"selectInSpace",
7      "jsonFormat": "{ "sender": String,
8               "objectId": int,
9               "state": String }"
10   }, ... ]
11   "subChannels":[{"name":"tagRecognised"}, ...]
12 }
```

Listing 5.3: Registration of PimVis

By using the `pubChannels` key, user interfaces can publish on the existing channels. To have a list of those different channels, developers can simply query the UIMW service. They can also check which user interfaces publish and are subscribed to each channels. New channels can be created via the UIMW REST API or by filling in the `newChannels` key as shown in Listing 5.3. A channel has three properties that needs to be specified. The first one is

its purpose, which can be *indicate the location of a physical document*, for example. This means that any subscribed interface, such as the ring binders interface, will perform this action when receiving data from this channel. A channel also needs a name which can be specified by the `channelName` key. Finally, the last property is the JSON data format, which shows how data will be passed in the event stream. This last one is very important for the subscribers, since it allows them to know exactly what they will receive when something is published on the channel and in which format they will receive it. In the PimVis example, each subscriber will receive events from the newly defined *selectInSpace* channel which will contain the ID of a physical document (`objectId`) and the state of the document, such as the *selected* state. The unique document ID is specified by the OC2 framework and is fixed for each document. Note that the JSON format that will be passed to the other user interfaces can vary, depending on what the developers want it to look like.

The last key that needs to be filled in to register a user interface is the `subChannels` key. This key allows the new user interface to subscribe to channels by giving their specific channel name. The UIMW service only gives a mechanism to check the functionalities offered by each user interface. Considering that the standardisation of the JSON format depends on the publisher and can be found in the UI repository component of the UIMW service, building cross-media and cross-device interaction spaces can be done with a high flexibility.

5.4.2 User Profile

The user profile component of the UIMW service takes care of the available and installed user interfaces of a specific user. For example, suppose that Alice decides to buy a new augmented filing cabinet, she first wants to know if her already installed user interfaces in her office environment will be able to interact with it. Once that Alice bought the new filing cabinet, it should be automatically added to her profile and registered with her local instance of the XUI framework. The user profile repository is managed via a REST interface. In order to add new users in the repository, a username, password and their current user interface setup must be provided. This is done via the function `addUser` illustrated in Listing 5.4.

```
    @POST
    @Path("/addUser")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces("application/json")
    public Response addUser(String input){
        JSONObject  obj = new JSONObject();
```

```
        JSONObject in = new JSONObject(input);
        String name = in.getString("userName");
        if(!UIMWCore.get().checkUserName(name)){
            String pass = in.getString("password");
            JSONArray uisJson = in.getJSONArray("uis");
            ArrayList<UI> uis = new ArrayList<UI>();
            for(int i = 0; i<uisJson.length(); i++){
                JSONObject uiObj = uisJson.getJSONObject(i);
                String uiName = uiObj.getString("nameUI");

                UI ui = UIMWCore.get().getUI(uiName);
                if(ui!=null){
                    uis.add(ui);
                }
            }
            User user = new User(name, pass);
            user.setUis(uis);
            UIMWCore.get().addUser(user);
            User us = UIMWCore.get().getUser(name);
            System.out.println("in rst " + us.getUsername());
            obj.put("status", "ok");
        }else{
            obj.put("status", "error");
            obj.put("error", "exists");
        }
        String json = obj.toString();
        return Response.status(201).entity(json).build() ;
    }
```

Listing 5.4: Registration of a new interface

Adding a new interface to a user's profile is done via the `addUiUser` function. When a PUT request is sent to the UIMW server to add a user interface, the username and the name of the new interface must be provided. The UIMW server will verify if the user is already stored in the *User Profile* repository. When this is the case the profile of the user will be updated with the newly added interface.

```
    @PUT
    @Path("/addUiUser")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces("application/json")
    public Response addUiUser(String input){
        JSONObject  obj = new JSONObject();
        JSONObject in = new JSONObject(input);
        String name = in.getString("userName");
        String uiName = in.getString("uiName");
        User user = UIMWCore.get().getUser(name);
        if(user!=null){
            UI ui = UIMWCore.get().getUI(uiName);
            if(ui != null){
                user.addUI(ui);
                UIMWCore.get().updateUser(user);
            }
        }
        String json = obj.toString();
        return Response.status(201).entity(json).build() ;
    }
```

5.5 The Interaction in the Office of the Future

In this section, the implementation details about how data is retrieved by the different user interfaces are examined.

5.5.1 PimVis

In order to start the PimVis interface, data needs to be retrieved from the XUI server. This data contains all the personal information of the user. To retrieve this data, which is received in JSON format from the XUI server, a few GET requests were made. The specific XUI endpoints that were used are listed in Listing 5.5. First, the context structure is retrieved from the server, which outputs it in a tree structure and is shown on line 1 and 2. After that, the tree is traversed to find the leaf contexts. For each leaf context, the PimVis interface will send a request to the XUI server in order to get the documents within a certain context. Lines 4 to 18 illustrates how to get all the properties of a document, being its preview, location, contexts, creation dates, tags and type. The links between documents are retrieved by executing the GET requests of lines 20 to 22. While collecting this information, a tree structure is built, which contains all the data needed to construct the *context view* and *document view*. Once the data retrieved, these views can be constructed.

```
1  ../extended/contextStructure/getAllStructures
2  ../extended/contextStructure/getData/{contextStrucId}
3
4  ../core/context/getAllRelatedObjects/{contextId}
5  ../extended/general/getResourceInfo/{objectId}
6  ../core/object/getProperty/{objectId}/imageUrl;
7  ../core/object/getProperty/{objectId}/imageIconUrl;
8  ../core/object/getProperty/{objectId}/width;
9  ../core/object/getProperty/{objectId}/height;
10 ../core/object/getProperty/{objectId}/loc;
11 ../core/object/getAllRelevantContexts/{objectId};
12 ../extended/general/getResourceInfo/{contextId};
13 ../extended/object/getLifeLine/{objectId}/Interactions;
14 ../extended/lifeLine/getLifeLine/{lifeLineID};
15 ../core/object/getExtLinksITarget/{objectId};
16 ../core/extLink/getSource/{extLinkId};
17 ../extended/general/getResourceInfo/{conceptId};
18 ../extended/object/getType/{objectId};
19
20 ../core/object/getNavLinksITargetInContext/{objectId}/{contextId}
21 ../core/navLink/getSources/{navLinkId}
22 ../core/navLink/getWeightInContext/{navLinkId}/{contextId}
```

Listing 5.5: GET requests to the XUI server done by PimVis

As mentioned before, the PimVis interface publishes on two channels, the *selectInFS* and *selectInSpace* channel, depending whether digital or physi-

cal documents need to be re-find in the file system or in the ring binders respectively. Publishing to these channels is done via a POST request containing the ID of the document that needs to be retrieved. These requests are illustrated in Listing 5.6 and 5.7.

```
$.ajax({ url: "http://192.168.0.21:8080/ReViTaFinal/pubsub/mui/selectInFS"
      ,
   type: 'POST',
   contentType: "application/json",
   data: '{"objectId":"'+nId+'"}' });
```

Listing 5.6: Re-finding a digital document POST request

```
$.ajax({ url: "http://192.168.0.21:8080/ReViTaFinal/pubsub/mui/
    selectInSpace" ,
   type: 'POST',
   contentType: "application/json",
   data: '{"objectId":"'+nId+'"}' });
```

Listing 5.7: Re-finding a physical document POST request

5.5.2 Windows File Explorer

The Windows File Explorer has been changed by adding the extended file system application. This file system listens to the channel *selectInFS*, where the PimVis interface publishes data. When this data is published, the file system can use it in order to open the folder in the File Explorer containing the right document. To do this, the location of the document needs to be retrieved from the XUI server. Listing 5.8 illustrates how the path of a document can be found using a GET request to the RESTful XUI API.

```
HttpResponse<JsonNode> request = Unirest.get("http://"+host+"/ReViTaFinal/
    oc2/extended/object/getLocationProperty/"+ objectid +"/loc").asJson();
JSONObject urlRes = request.getBody().getObject();
String url = urlRes.getString("value");
if(!url.equals("")){
    Runtime.getRuntime().exec("explorer.exe /select, " + url);
}
```

Listing 5.8: Locating a digital document in the file system

When a document of the File Explorer needs to be opened in PimVis, the extended file system application has to find the ID of this document. The selected document's location will be used in order to find it on the XUI server. The GET request to the server using the location property is shown in Listing 5.9. Once the document object retrieved from the server, the file system application publishes its ID on the *openPimVis* channel to which the PimVis interface is listening. This is done by using a POST request. As

result, the ID of the document will be intercepted by the PimVis interface, which will open the corresponding document in its *focus view*.

```
HttpResponse<String> reqGetObjects = Unirest.get("http://"+host+"/
    ReViTaFinal/oc2/extended/object/getObjectsWithLocationProperty/"+
    propid).asString();
  JSONObject resultObjectsList = new JSONObject(reqGetObjects.getBody());
JSONArray arrObjects = resultObjectsList.getJSONArray("objects");
  if(arrObjects.length()==1){
    JSONObject nodeObjId = arrObjects.getJSONObject(0);
      String pub = nodeObjId.toString();
      Unirest.post("http://"+host+"/ReViTaFinal/pubsub/mui/openPimVis").
          body(pub).asString();
}
```

Listing 5.9: Opening a document in PimVis

5.5.3 Ring Binder

The ring binders application listens to the *selectInSpace* channel in order to light up the LED of the correct ring binder when a document shown on the PimVis interface needs to be re-find. When this occurs, the PimVis interface publishes the ID of the document to be re-find on this channel. Based on the intercepted id, the location of the document needs to be retrieved. This is done by sending a GET request to the XUI server, as shown in Listing 5.10. Once that the location is known, a signal will be transmitted to an infrared transmitter, which will be intercepted by the corresponding IRPhidget located in the ring binders. As result, the LED located in the correct ring binder will be lit up.

```
HttpResponse<JsonNode> request = Unirest.get("http://"+host+"/ReViTaFinal/
    oc2/extended/object/getLocationProperty/"+ objectid +"/led").asJson();
  JSONObject urlRes = request.getBody().getObject();
String url = urlRes.getString("value");
    if(!url.equals("")){
    search.find(Integer.parseInt(url));
  }
```

Listing 5.10: Registration of PimVis

Physical documents can be opened in the PimVis interface by placing them under the camera mounted on the desktop. This interaction is also a part of the ring binders application, since these documents are taken from a ring binder. Once the document placed under the camera, the fiducial marker will be recognised by a TUIO listener. This marker will be used to find the property ID needed to make a GET request to the XUI server based on this property. The GET request corresponds to the one of Listing 5.9, which was used to open a digital document in the PimVis interface. Again, once the

document object retrieved from the server, a POST request can be made in order to publish the document ID on the *openPimVis* channel. This ID will be received by the PimVis interface, which will open the corresponding document in its *focus view*.

6 Evaluation

In this chapter, we present the user study that was conducted in order to investigate the benefits and pitfalls of fluid cross-media and cross-device interaction spaces. Additionally, attention is paid to the natural and intuitive interaction of the participants while working in our Office of the Future setting. First, the use case of the study is discussed based on the use cases introduced in Chapter 3. After that, the participants and the methodology are described. Finally, a quantitative and qualitative analysis of the results of our study is presented.

6.1 Use Case of the Study

Evaluating a system dealing with personal information management data is not trivial since the information should be the participant's personal information. Therefore, we followed the best practice of our community and populated the Office of the Future environment with a co-workers personal information. The organisation of the documents is based on Boardmann's guidelines [6].

At the beginning of the user study, before executing any tasks, the participants were given some time to get familiar with the documents forming part of the setup. As explained in Chapter 3.1, our Office of the Future setup contains three user interfaces. The first one is the PimVis user inter-

face, which is accessible on the tabletop. The next one is the extended file system interface, which is located on the desktop computer. The last user interface is the ring binders application, which comprises the augmented ring binders containing sheets with fiducial markers. Documents in the PimVis interface, the artificial file system as well as the physical documents in the four augmented ring binders were shown together with a short explanation of how to navigate through these user interfaces.

During the study, participants had to perform tasks that are common for researchers. Concretely, the *paper writing* use case has been used. This means that participants were given tasks related to the process of writing a paper. These tasks were similar to the ones described in use cases of Chapter 3. The participant begins on the desktop computer with the File Explorer open in the right folder. This is the folder in which all the documents needed so far for the paper writing are located. The PimVis interface on the tabletop shows the bubble representation of the different contexts of the user. The general contexts were *Meetings*, *Thesis Students*, *Current Research Projects*, *Conferences* and *Holidays*. A typical task is, for example, to look for a paper used during a certain meeting. Other tasks could be finding specific notes taken while working on the implementation of the paper that needs to be written or searching for related papers of a document that could be used in the related work section of the paper.

6.2 Participants

Eight participants were recruited to participate in the experiment which took place at the WISE lab. All participants had some experience in the writing of research papers or Master's theses and used LaTeX before. All of them were Master students in their final year or PhD students. This specific population was chosen due to the fact that the participants would be more familiar with the given use cases.

6.3 Methodology

Participants were introduced to our Office of the Future environment. An explanation about which devices could be used during the experiment was given to the users. Each participant was given two sheets of paper with the tasks that had to be performed. In order to observe a difference between the setting without fluid interaction, called the *isolated* setting, and the one with fluid interaction, a multicase case study was conducted. The first set

of tasks had to be performed in the *isolated* setting, while for the second set of tasks the participants could use the fluid interaction approach. In the *isolated* setting, the user can use the PimVis application, the Windows File Explorer and the ring binders separately. The three user interfaces are not connected to each other and the participant does not know about any connection between them. In the fluid interaction approach, the XUI framework can be used, so users were shown how to use the *re-find* button in PimVis. Additionally, the Windows File Explorer extension as well as the LED-augmented ring binder user interface were revealed.

Research instruments

During the execution of these tasks each participant was observed and timed. After completing all tasks, participants had to fill in a 26 items User Experience Questionnaire (UEQ)[1] for the quantitative results. This questionnaire can be found in Appendix B. In addition, a semi-structured interview was conducted for qualitative results. The quantitative results were analysed based on certain usability and user experience factors. These factors are *attractiviness*, *perspicuity*, *novelty*, *stimulation*, *dependability* and *efficiency*. In order to construct these six factors, we did a factor analysis. All factor sets have a Cronbach's alpha value higher than 0.7. After that, normality has been tested with the Shapiro-Wilk test so that a t-test or Wilcoxon-Rank test could be used to test for a significant difference between the two cases. The qualitative data was examined and resumed in a few design guidelines. This will be explained in more detail in Section 6.4.

Participant's tasks

Once the participants were more familiar with the setup, they begun with the *isolated* case, which contained six tasks. As previously mentioned, each individual task was a small activity that a researcher might perform while writing a conference paper. The six tasks are shown in Table 6.1. For the second case, similar tasks were used but with different documents to prevent a learning effect. These tasks can be found in Table 6.2.

[1] http://www.ueq-online.org/

Case 1: Without Integration	
Task 1	You want to read the paper "Reality Editor". This paper is related to the paper called "Objectop", which you used in your current task and which is stored in the "papers" folder in the current directory.
Task 2	You want to store a paper in your "papers" folder which is about "DocuDesk". You remember that the paper was used during a meeting with Audrey. The paper is also stored in her folder. Copy the paper to the "papers" folder in the directory in which you are currently working.
Task 3	Find the picture of a "LED" that you took while working on the Cool Office project. Which documents are related to this image? Open the image on the desktop.
Task 4	Find the note on which you wrote "general contexts" at the top. You have used it previously when writing this paper. Go back to the desktop and google "Holidays".
Task 5	You need to cite another paper which is about "PaperSpace". This paper is related to the paper called "WatchConnect" which you used in your current context and is stored in the "papers" folder.
Task 6	You want to add a comment to your physical copy of the paper about "StackTop". The paper was used while writing the current paper. Find the paper in the bookcase.

Table 6.1: User study Case 1 - Without fluid interaction integration

Case 2: With Integration	
Task 1	You want to read the paper "iCap". This paper is related to the paper called "Lee2013" about a "Tangible programming tool" which you have used in your current context and is stored in the "paper" folder which is in the current directory.
Task 2	You want to store a paper which is about "Improving Intelligibility and Control in Ubicomp" by Jo Vermeulen. You remember that the paper was used during a meeting with Wouter. The paper is also stored in his folder. Copy the paper to the "papers" folder in which you are currently working.
Task 3	Find an image with Vince where his face is shown and is taken in DisneyLand. Open the image on the desktop.
Task 4	Find your note with written "Layers of Abstraction" on the top. You have used it previously when writing this paper. Find the paper in the ring binders. Go back to the desktop and google "End users".
Task 5	You need to verify some details which are in the "Drools Manual", you have used it during the development of the CMT tool. The manual is related to the paper called "Park2013" which you have used in your current context and is stored in the "papers" folder.
Task 6a	You want to add a comment to your physical note where you have written "Corrected Schema" at the top. The note was used when writing this paper. Find the note in the bookcase.
Task 6b	Find the note with "Working with the Tools" printed at the top in the same ring binder. In how many contexts/tasks did you use this document?

Table 6.2: User study Case 2 - With fluid interaction integration

In both cases, the tasks were split into two categories. The first category made it possible for users to stay on a single device to perform the tasks, while for the second category the tasks could hardly be performed using only one device. The first three tasks are from the first category, meaning that, for example, the participant did not need to switch from the tabletop to the desktop environment. Each of these tasks could be performed either on the tabletop or on the desktop computer. An example of one of these tasks is: "You want to read the paper *Reality Editor*. This paper is related to the paper called *Objectop*, which you used in your current context and which is stored in the *papers* folder in the current directory" (see Task 1 in Table 6.1).

Considering that the remaining three tasks belong to the second category, users could not solve them by only using a single user interface. They had to use a combination of them, since the needed information was spread across multiple interfaces. Additionally, for this last range of tasks the desktop computer was always used as starting point. An example of one of these tasks is: "Find the note with "Working with the Tools" printed at the top in the same ring binder. In how many contexts/tasks did you use this document?" (see Task 6b in Table 6.2). This design allowed us to investigate whether users naturally switch back to their main workplace (i.e. desktop computer) after each task. Additionally, with the last range of tasks the experience of participants could be examined while switching between interfaces in the *isolated* case in contrast to the fluid interaction case.

6.4 Results

This section provides an overview of the results of the conducted user study. First, the quantitative results will be analysed using statistical tests. After that, the qualitative results will be presented, which provides a few guidelines for designers of fluid interaction spaces.

6.4.1 Quantitative Analysis

The quantitative results have been calculated through the User Experience Questionnaire (UEQ) that each participant had to fill in after finishing both set of tasks. The UEQ allows to measure the user's experience while using the Office of the Future setting. It is based on six different categories, going from usability aspects, such as efficiency and dependability, to user experience aspects, such as originality and stimulation. Each question in the questionnaire belongs to one of these categories.

Factor	Keywords
Attractiveness	Pleasing, Enjoyable, Good
Perspicuity	Understandable, Easy, Clear
Novelty	Innovative, Creative, Inventive
Stimulation	Exciting, Interesting, Motivating
Dependability	Predictable, Supportive, Secure
Efficiency	Fast, Practical, Organised

Table 6.3: User Experience Questionnaire factors with keywords based on the questionnaire's questions

The first category is *Attractiveness*, which measures the overall impression of the users about the system. The second category is *Perspicuity*, which verifies whether the users can get easily familiar with the system. The next category is *Efficiency*, which checks if users can do their tasks without performing unnecessary actions. The fourth category is *Dependability*, which measures if users feel in control when interacting with the system. The fifth category is *Stimulation*, which verifies how excited and motivated participants are when using the system. The last category is *Novelty*, which asks the users about how innovative and creative they think the setup is. An overview of these factors along with a couple of their properties can be seen in Table 6.3.

Reliability of the data

To verify whether the data retrieved from the questionnaire is reliable, the Cronbach's alpha value per category for each case is calculated. The data is considered reliable enough if the Cronbach's alpha value is higher than 0.7. When this value is lower than 0.7, one or more questions which have the most impact on the value must be removed. By applying this process on the data, we get Table 6.4. The table shows all the remaining questions per category and per case. For example, `Attractiveness1` represents the data obtained from the *isolated* case for the attractiveness factor, while `Attractiveness2` designates the data from the fluid interaction case for the attractiveness factor. For this factor, it is apparent that question 24 was removed in the second case in order to obtain a Cronbach's alpha value higher than 0.7. In addition to the factors and their remaining questions, the final Cronbach's alpha value is given. By looking at the remaining questions of certain factors, interesting things can be seen. The `Dependability` factor, for example, is originally measured by four questions, being 8, 11, 17 and 19. Notice how the remaining questions of each case are disjunct sets of the original questions. In

the first case, questions 17 and 19 remain, while for the second case, questions 8 and 11 are kept. This is a typical dataset that could provide misleading results when running statistical tests if not pruned correctly. Therefore, calculating the Cronbach's alpha value of each question set is in fact an important task.

Factor	Questions	Cronbach's Alpha
Attractiveness1	1, 12, 14, 16, 24, 25	0.81
Attractiveness2	1, 12, 14, 16, 25	0.75
Perspicuity1	2, 4, 13, 21	0.70
Perspicuity2	4, 13, 21	0.75
Novelty1	3, 10, 15, 26	0.83
Novelty2	3, 15	0.73
Stimulation1	5, 6, 7, 18	0.86
Stimulation2	5, 6, 7, 18	0.80
Dependability1	17, 19	0.72
Dependability2	8, 11	0.71
Efficiency1	9, 20, 22, 23	0.74
Efficiency2	9, 20, 22	0.79

Table 6.4: Cronbach's Alpha of factors with the remaining questions

Normality of the data

Knowing that the datasets are reliable, the significant difference between the two cases for each factor can be verified using the paired two-sample t-test. The paired-samples t-test is used to compare the means of two samples of repeated measures. Since the question sets of each factor have been collected once in the *isolated* setting and once afterwards in the fluid interaction setting, this statistical test can be performed on the user study's dataset.

In order to perform a t-test, the datasets must be normally distributed. To test for normality, there are two well-known tests, the Kolmogorov-Smirnov test and the Shapiro-Wilk test. Since the latter is more powerful for small sample sizes, this one has been used for testing the normality of the samples. The hypotheses for this test has been defined as follows.

The null hypothesis *The data is normally distributed.*

The alternative hypothesis *The data is not normally distributed.*

The results of the Shapiro-Wilk test are shown in the fifth column of Table 6.5. Based on the significance values (p) of this test, we can assume

that all samples except for one factor, namely the `Efficiency1` factor, have a normal distribution. This can be affirmed, since all datasets exceed the significance level α of 0.05, except for the *efficiency*-dataset of case 1. This last one rejects the null hypothesis, since the p-value is lower than 0.05, which means that this dataset is not normally distributed. Consequently, the t-test cannot be used on the *efficiency* factor datasets. So, to verify if there is a significant difference in the two cases of this factor, we used the Wilcoxon signed-rank test. This test can be used as an alternative for the paired t-test when the dataset cannot be assumed to be normally distributed. The following hypotheses were used for the t-test and Wilcoxon signed-rank test.

The null hypothesis *There is no significant difference between the two cases.*

The alternative hypothesis *There is a significant difference between the cases.*

Quantitative results

The results of the t-tests and Wilcoxon signed-rank test are shown in Table 6.5 along with the *mean*, the *mean's confidence interval* and the *standard deviation* of the datasets. Notice that every p-value of the results of the t-test has a value higher than 0.05. This means that the null hypothesis is not rejected. Therefore, it cannot be said that there is a significant difference between the two cases for the factors *attractiveness*, *perspicuity*, *novelty*, *stimulation* and *dependability*. By looking at the results of the Wilcoxon signed-rank test of the *efficiency* factor, the same conclusion can be made, since the p-value is higher than 0.05, namely 0.21. This means again, that the observed difference between the two *efficiency* datasets are not significant, since the null hypothesis cannot be rejected. However, to have a better understanding of the results obtained from the datasets of the different factors, an overview of each factor's dataset characteristics is given in the next few paragraphs.

Attractiveness In Figure 6.1, the different means per factor for each case are shown, together with their error bars. The mean of the *attractiveness* factor of the second case is quite higher than the one of the first case. However, as mentioned earlier, there is no significant difference between them. The questions are on a scale of 7. As illustrated in the bar chart, the mean of this factor is in both cases positive, since it is higher than 3.5. The fact that

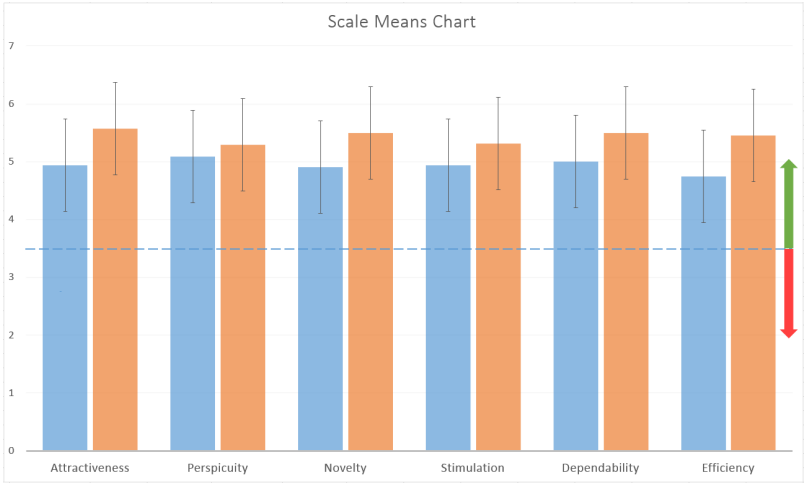

Figure 6.1: Bar chart comparing the means of each factor per case. The dotted line indicates the separation line between a positive and negative mean. This is also illustrated by the red and green arrows.

there is no significant difference between both cases may result from the fact that the Office of the Future setup was already appealing to the participants right from the beginning. Adding the fluid interaction to the setting did not significantly make it more pleasant or enjoyable to interact with.

Perspicuity The second factor shown in Figure 6.1 is the perspicuity factor. This factor has, for the first case, a mean higher than the attractiveness mean, which means that the setup was already clear and easy to learn. The *perspicuity* factor has the highest mean for the *isolated* case. It also has the smallest difference between the means of the two cases, which shows that adding the extra interaction possibility did not make the system more understandable. The standard deviation of the questions for the *perspicuity* factor is relatively high, namely `1.10`, which is higher than the standard deviation of most factors. This signifies that the participants had diverging opinions.

Novelty When looking at the mean values of the *novelty* factor, the second case has again a higher value than the first one. As can be seen in Table 6.5, the standard deviation of both cases' mean is relatively high, with both values above one. This means again that participants had different opinions about the novelty of the setup. Four participants gave very positive feedback about the novelty of the system while the others answered around average. The fact that the setup of the Office of the Future may already be creative and innovative for the users can play a role why there is no

significant difference between the cases for this factor. Many people are not used to interact with a graphical user interface on a multi-touch tabletop.

Stimulation The *simulation* factor, just as the others, does not show a significant difference while using the *isolated* case or the second case. The mean of the second case is however a little higher than the one of the first case. Both are higher than 3.5, which is positive and means that users felt motivated when interacting with the setup. The Office of the Future may already be interesting and valuable for the participants without the integration of the fluid interaction, explaining why there is no significant difference between the cases for this factor.

Dependability After the *perspicuity* factor, the *dependability* factor has the highest mean for the *isolated* case. For the second case, it also has the second highest mean, together with the *novelty* factor. This suggests that the setup was meeting the expectations of the users and that it provided a predictable output when performing their tasks. The fact that there is no significant difference between the two cases of this factor, may be because users already felt in control of the system while using it without the fluid interaction.

Efficiency The efficiency factor has the highest difference in means between the two cases. One could expect the second case to be significantly faster and more practical than the first one, since the fluid interaction made it possible to use the *re-find* button, which is supposed to facilitate interaction between the interfaces. However, from the conducted statistical tests, no significant difference was detected. This could be due to the fact that participants did not always think about using the *re-find* button. It is possible that by spending more time with the system, users would remember to use this button.

Factor	Mean	Confidence Interval	Standard Deviation	Shapiro-Wilk (p)	T-Test		Wilcoxon Signed-Rank Test	
					t	p	Z	p
Attractiveness1	4.94	0.71	0.84	0.69	-2.08	0.08	-	-
Attractiveness2	5.58	0.58	0.70	0.79				
Perspicuity1	5.09	0.65	0.78	0.43	-0.43	0.68	-	-
Perspicuity2	5.29	0.92	1.10	0.50				
Novelty1	4.91	1.01	1.20	0.94	-1.26	0.25	-	-
Novelty2	5.50	0.87	1.04	0.49				
Stimulation1	4.94	0.84	1.01	0.10	-1.47	0.18	-	-
Stimulation2	5.31	0.83	1.00	0.82				
Dependability1	5.00	0.92	1.10	0.69	-0.86	0.42	-	-
Dependability2	5.50	0.59	0.71	0.27				
Efficiency1	4.75	0.84	1.00	0.03	-	-	-1.26	0.21
Efficiency2	5.46	0.71	0.85	0.15				

Table 6.5: Statistical test results by factor and by case

6.4.2 Qualitative Analysis

By observing the participants executing their tasks, interesting behaviour could be noticed. Additionally, by conducting the semi-structured interview, important comments from the users were captured. An overview of the basic questions asked during this interview is shown in Table 6.6. With this acquired knowledge, we present a few benefits and pitfalls for the design of cross-media and cross-device user interfaces forming a fluid interaction space.

#	**Questions**
1	Would you buy the individual application? — What is the reason you will/will not buy it?
2	Which version of the two setups would you like to have at home? — What are the advantages and disadvantages? — Can you think of a situation in your life when you would use this setting?
3	Do you think it would become chaotic if when you push a certain button, such as the re-find button in PimVis, and that various applications would do something? — What if you had the possibility to configure this?
4	What do you think about the online registration system?

Table 6.6: General questions of the semi-structured interview

The interaction's starting point is the current device

While participants were performing their tasks, in both cases, the *isolated* and the fluid interaction case, most of them always started to execute a task on the device with which they just solved the previous task with, even if doing this was not the fastest way to solve the current task. For example, when participants finished the previous task interacting with the desktop computer, they would start the next task searching in the Windows File Explorer. On average, participants spent 25 seconds searching without results before deciding to switch to the PimVis interface, where they could explore the documents based on context (which cannot be done on the desktop). The same phenomenon was observed the other way around, when participants needed to search for an image in the picture folder and open it via the Windows File Explorer. Since they finished the previous task on the tabletop, most of them traversed the PimVis context hierarchy instead of just navigating to the picture folder on their desktop computer.

Digital and physical mental models in cross-media interactions

During the semi-structured interview, the participants were asked which of both cases they preferred. All participants favoured the second case with the integration of the XUI framework, which they found much faster. They also had a notably more natural experience while performing the second set of tasks. For example, when they had to find related documents of a specific document in the file system, they could easily find the document via the PimVis interface and press the *re-find* button to open it on the desktop, instead of manually retyping the path. Nevertheless, a significant difference was observed in the interaction when users needed to re-find a physical document or when a digital one was needed. While all participants used the *re-find* button to find a physical document by lighting up the LED in the right ring binder, not all of them used the button to find a digital document in the file system. Six of the eight participants navigated through the file system of their desktop computer following the path that was indicated in the document's detail panel in the PimVis interface. They did not use the *re-find* button, which would have opened the Windows File Explorer in the right directory and selected the needed document in a matter of seconds. This shows that users have different mental interaction models for these two media types. By changing the description of the location in the PimVis interface into a hyperlink pointing to the file system's folder, it might be more natural for most users to make use of this interaction. Instead of pressing the *re-find* button, they could follow the hyperlink when trying to re-find a digital document.

Differences in hot and cold documents

The small ubiquitous user interfaces such as the Windows File Explorer extension and the ring binders interface and the way they are integrated in the Office of the Future were quite popular and appreciated by most of the participants. During the interview, they came up with some interesting cases of how they could use these types of ubiquitous user interfaces in their own workplaces. Some participants stated that it would be nice to have small Windows File Explorer extensions for documents that they often use during short term tasks. These kinds of documents are called hot documents. For such documents, related documents could, for example, be shown in a side panel of the file system when a file is selected. This could be more convenient for the users than opening it in the PimVis interface on the tabletop. Opposed to these kinds of documents, there are the documents that are archived and not often used during the user's daily tasks. These documents are called

the cold documents. According to our interview with the participants, an application such as PimVis could be favoured to handle those documents. This might mean that for physical augmentations, indicating the corresponding ring binder is maybe not sufficient, the exact location in the ring binder might also be needed. When developing ubiquitous user interfaces, designers should take into account which kind of documents will be used while interacting with their interface.

Self-development and configuration

As mentioned before, several participants came up with interesting real-life use cases. Going beyond the application of the XUI framework, they were inclined to spend some time augmenting their personal artefacts such as books, shoes or toys for children. For most participants the ideal setup would be a customisable augmented environment where they could augment arbitrary artefacts. Additionally, they would like to be more in control of the triggered actions when pressing the *re-find* button. They suggested to change the action depending on the type of the document. For example, if the document is a PDF document, the user suggested that the *re-find* button could trigger the opening of the document in the standard PDF reader on the user's computer instead of just opening the right folder in the file system. Others also proposed to trigger different actions depending on the task they were working on. If the user is, for example, in a meeting context, pressing the *re-find* button might directly open the email client on the desktop of the user in order for the document to be sent by email immediately to the person the user is meeting with.

However, the level control over the fluid interaction space configuration is somewhat user dependent. Certain users are prepared to spend a lot of time in configuring the whole interaction space by defining each and every action while others prefer to just have some standard settings or spend a limited amount of time on personalising these settings.

6.5 Conclusion

To summarise, we can conclude from the quantitative results that there is no significance difference between using the system in the *isolated* case and when using it in the fluid interaction case. However, by observing the participants performing their tasks and by conducting a semi-structured interview, interesting behaviour, suggestions and comments could be collected. Based on the collected information, we presented design guidelines for developers

of user interfaces designed for fluid cross-device and cross-media interaction space. These guidelines also illustrate the benefits and pitfalls of designing interfaces for such fluid interaction spaces.

7 Conclusion and Future Work

In this chapter, we discuss the PimVis interface as well as the XUI framework and outline some future work. Finally, some conclusion about the work performed in this thesis is given.

7.1 Discussion

In this section, we discuss the PimVis user interface and the XUI framework. The PimVis interface was one of the example user interfaces in our Office of the Future setting, which has been used to illustrate the potential of the XUI framework. In the following, we compare the PimVis interface to related work in the domain of information visualisation. In contrast, the XUI framework is compared to related work in the fields of cross-media and cross-device interaction.

7.1.1 PimVis

The PimVis interface is mainly based on different existing visualisations for large document spaces. The ForceSpire visualisation of Endert et al. [13] represented the documents as dots and related documents were illustrated by a line between the dots. Relations between documents or other entities are also illustrated via links by Fairchild et al. [16] in the SemNet visualisation

and by Heer and Boyd [24] in their Vizster interface. Other authors, such as Wise et al. [63], Iwata et al. [25], Lin [36] and Fortuna et al. [19] use the spatial distance between elements to represent how related they are to each other. The PimVis interface adopted the link representation to show relations between documents, which has also been suggested by Shneiderman [48]. Just as Fairchild et al. [16] noticed, when a lot of links are present, the overview can rapidly be lost. The PimVis interface tried to cope with this problem in the *document view* by showing only the links that are relevant to the current context. However, when a document is opened in the *focus view*, all its related documents are shown including those of another context. Links have weights associated to them, to indicate how important they are in a specific context. This weight system has also been used by Endert et al. [13], where the weight of the links depends on the number of common keywords in related documents.

Different search and filter techniques to explore the data collection were applied in PimVis. A search box is used to search based on the title of the documents. This technique was often used during our user evaluation and is common in visualisation tools [13] [19]. In order to provide more information about a specific item in the information space, PimVis shows a side panel or control panel containing the most important information about the selected element. The Vizster [24] visualisation also uses this technique to show information about different persons in a social network. Furthermore, Fortuna et al. [19] as well, use a side panel, to provide more information about the information space and to display the description of a document. In contrast to PimVis and Vizster, this side panel was located on the left part of the screen instead of on the right. Based on the document's properties, PimVis users can use the tag cloud, context explorer and type filter which are located in the same control panel. Additionally, the timeline can be used to filter based on the creation dates of the documents. Filtering based on tags or keywords is also used in ForceSPIRE by Endert et al. [13], but they used a search box to do so, we opted for a tag cloud for novelty and also because tag clouds become more and more popular on the World Wide Web. The context explorer and type filter are not often seen in existing visualisations, but the timeline component has been used many times when elements involve the notion of time [63] [2] [1] [17]. The PimVis interface uses the D3 brush component, which can be compared to rangesliders used in the FilmFinder interface [1]. This type of slider allows users to define a range in which the data should belong instead of just indicating a single point in time, such as for normal time sliders. These normal sliders requires precision, which is not easy to have on multi-touch devices.

The *context view* of the PimVis interface represents the different contexts by using bubbles. Bubbles forming a part of the same general context belong to the same bubble hierarchy. This means that contexts that are closely related to each other are spatially also closer to each other. Proximity is thus used to illustrate similarity between contexts, such as [63], Iwata et al. [25], Lin [36] and Fortuna et al. [19] did to show similarity between documents. Different types of contexts are shown by using different colours. Using colours to differentiate between various element types is also done in Iwata et al.'s [25] visualisation of a movie database.

PimVis uses different views, such as the *context*, *document* and *focus view*, to visualise the data in order to get an overview of all the user's documents. Furthermore, it allows us to show the right amount of data at once on the screen. Wise et al. [63], also present three primary display types or views to provide another insight of the information space. These are the primary information view, called the backdrop, the workshop and the chronicle view, which all represents information differently.

7.1.2 XUI Framework

The XUI framework combines the work of two research domains, the cross-device interaction and cross-media interaction field. It supports digital as well as physical user interfaces with cross-media and cross-device interactions. In contrast to related work, the XUI framework has the objective to incorporate an Object-Oriented User Interface (OOUI) design with lightweight data exchange and the ability to have ad-hoc configurations of the dynamic Office of the Future setting. This is done by integrating the OC2 PIM system and by deploying an online UIMW service that manages a user's local office configuration. These three main requirements are necessary in order to make the interaction between user interfaces more fluid, flexible and extensible. The benefit of this approach is that it creates the opportunity to build more natural user interfaces, which go beyond the *isolated* functionality offered by the single components. To illustrate the potential of the XUI framework, the Office of the Future setting has been created. The setting includes three interfaces. The first interface is the PimVis tabletop visualisation, which is used to explore the user's personal information space. The second one is the extended file system, which added a context menu to the Windows File Explorer to interact with the PimVis interface. The last user interface is a physical one, which contains several augmented ring binders with an integrated LED to indicate the location of a document.

Many systems allow interaction with specific devices, but they often do not allow new types of devices to be added, as it is supported in the XUI framework. The Deep Shot system [10], for example, only focusses on the interaction between a smartphone and a computer. Other systems, such as the Conductor [23] and the HuddleLamp systems [43] essentially use tablets. Devices in both of these systems can be added in an ad-hoc manner. The Conductor system allows devices to be added to the interaction environment by bumping, NFC tags or QR codes. In the HuddleLamp setup, a device becomes part of the interaction environment by opening a URL on the device and placing it on the table. In contrast, to add new devices to the XUI framework, users should buy user interfaces compatible to their setup. Once our interface has been bought, it will be added to the user's profile via the UIMW registration system. The interface will be usable as soon as the user gets home without any necessary configuration.

The configuration of the different user interfaces forming a part of the Office of the Future setting is stored in the *user profiles* repository. All users can access their profile in order to see the interfaces within their interactive setup. The ARIS [5] and Conductor [23] interfaces also allow users to visualise the various devices of the user's interaction space, but often in existing systems this configuration cannot be saved. The Conductor system allows the state of a device to be saved to a list of sessions, but the system does not allow to save the entire configuration of devices in the interactive environment. The XUI framework automatically saves the setup of its users and even allows them to compare their settings.

The communication between devices of the XUI framework is done via a publish/subscribe mechanism. This means that an interface can publish data on different channels. Additionally, interfaces that are subscribed to channels will be notified when data is published on these channels. A similar approach has been used by Hamilton and Wigdor [23] in their Conductor system. The system allows devices to broadcast data to other devices, thereby users have the possibility to select a device to which the data needs to be sent or they can just broadcast the data to every device of the interactive space. The current interaction possibilities in the Office of the Future environment are defined by the developers, but future work can be done to give this communication options to the users as this will be further explained in Section 7.2. The interaction between the digital devices included in the current setup is done via the *Open in PimVis* option in the extended file system and via the *re-find* button in the PimVis interface. The cross-media interaction is realised via fiducial markers on the sheets of paper. In addition, by using the *re-find* button when a physical document is selected, the LED of the

ring binder containing the document will light up. Fiducial markers or other kinds of markers, such as QR codes, were also used by Frosini et al. [20] to convey information across digital devices. Augmenting the interactive environment with overhead cameras, radio modules and infrared shields is also often done to allow communication between user interfaces. The Office of the Future setting uses an overhead camera to recognise the fiducial markers and a infrared shield to light up the LEDs attached to the ring binders. These artefacts are often used to augmented non digital objects, such as a paper documents [61] [14] [52], tabletops [43] or entire rooms [39]. For cross-media interaction, a desk-mounted camera is often used. The DigitalDesk [61] presented by Wellner and the DocuDesk [14] introduced by Everitt et al. are some examples using this system to enhance the experience of the users while working in their office. The example setup of the XUI framework also uses a desk-mounted camera to recognise the different documents of a user. In this setup, augmented paper is only used to open the document in the PimVis visualisation. However, the potential of paper-digital interaction is much bigger. Again, this will be explained in more details in Section 7.2.

In contrast to systems such as CoScribe [52], ReticularSpaces [3] [4], GroupTogether [39] and MultiSpace [15], which focus more on the collaborative aspect of the interactive environment, the example setup of the XUI framework does not focus on that aspect. However, the XUI framework is perfectly suited for collaborative setups where data needs to be transmitted across multiple interfaces. In the current Office of the Future setting, the tabletop could be used for collaboration up to a certain point. For example, it can be used during a meeting where two persons are seated next to the tabletop, as mentioned in Chapter 3.

7.2 Future Work

The PimVis user interface can still be improved, especially in terms of interaction possibilities. Until now, users can visualise and explore their contexts and documents via the different views provided by PimVis. During the conducted user study, participants often commented that it would be convenient to have a legend in the *context view*. This legend would then show the colour of the bubbles together with the name of their context. Until now, users had to tap on a bubble in order to know the name of its context. We originally chose not to add the context names because users were supposed to know which colour is associated to which context since it is supposed to be their own personal information space. Nevertheless, as shown by the semi-structured interview during the evaluation of our setup, the PimVis inter-

face is more likely to be used with cold documents that are not often used. Therefore, we concluded that a legend would improve the user experience when using the interface.

Additionally, more interaction possibilities could be added to the PimVis interface, such as adding, modifying and deleting contexts, documents and links between them. Up to now, we focussed more on the visualisation and exploration of the information space by providing different views and filtering techniques to filter the data. Furthermore, we aimed at providing fluid interaction possibilities between the PimVis interface and other user interfaces in the setup.

A few participants mentioned that when a lot of links are present between documents, it became less clear which documents are related to each other. A possible solution is to use the same technique as in the ForceSPIRE visualisation introduced by Endert et al. [13], where they show links between documents only when a document is selected. However, an additional user study should be conducted in order to evaluate the usability of the PimVis interface on its own. Furthermore, it would also be worth investigating if PimVis enhances the user's work process.

Finally, allowing users to interact with the paper documents on the tabletop might improve the natural interaction between physical and digital documents. In order to allow such interactions, touch points generated by a sheet of paper must be filtered out, so that it does not interfere with the touch points generated from the user's fingers. More paper-digital interaction options can then be explored. A physical document can, for example, be placed on the tabletop in order to compare it with a digital document and make some annotations, which was not possible in the current setup. Furthermore, these documents placed on the tabletop could also be recognised by an overhead camera. This could even lead to more collaborative possibilities, between users sitting around the tabletop. In future setups, more relevance could be given to the collaborative aspect. Even though an office environment is often used by a single person, people often have meetings on a regular basis in their offices where input from different persons might need to be considered. Moreover, by using the XUI framework, we are not limited to office environments. An interactive environment could be set up at home and could even use another information space. Participants suggested that the system could be used to organise other artefacts, such as books, shoes or a child's toys.

Future refinement and extensions can also be done with the XUI framework. In the current XUI framework, developers decide about the interaction possibilities between the different user interfaces. Some participants of our user

study suggested that it would be handy to let the users define parts of the interactions between the interfaces. It could be worth investigating how much control a user can be given to model a user interface's behaviour. Imagine that Alice, a user of the Office of the Future, wants to change the behaviour of the *re-find* button, which allows interaction between the PimVis interface and the extended file system or the ring binders application. Alice wishes that when pressing this button when a PDF document is selected, the document should be directly opened in the PDF reader on her computer instead of only opening the folder in the Windows File Explorer. Other users, such as Arthur, may want another behaviour. Therefore, it would be interesting to let the users model this interaction. Furthermore, users may also want to define from which device an interaction should be triggered. For example, Alice might want to use her smartphone to re-find a document in the augmented ring binders instead of using the PimVis interface. This customisations could lead to a more natural and fluid interaction space and improve the overall user experience even more.

In addition, during the user evaluation we noticed that participants tend to switch between devices dependent on their current task. For this reason, extending the XUI framework with a context-aware framework could be interesting. This framework will then automatically determine on which task the user is working, with other words, the user's context. Based on this context the behaviour of the user interfaces and their content can be adapted when performing cross-media and cross-device interactions.

7.3 Conclusion

We presented an innovative system that supports fluid cross-device and cross-media interaction in an office environment. The cross-device interaction field focusses on providing interaction possibilities between different devices in an interactive environment. In contrast, research in the cross-media interaction domain focusses on the interaction between the physical and digital information space, allowing the interaction between physical and digital documents. A lot of research in those fields has been done, but to the best of our knowledge, none of the existing systems combine both research fields.

We presented the eXtensible User Interface (XUI) framework, which contributes to both of these fields by combining them in order to achieve a fluid cross-media and cross-device interaction environment. An example setup called the Office of the Future setting has been built to illustrate the potential of the XUI framework. The setup consists of a desktop computer, a cam-

era, a multi-touch tabletop and ring binders. The file system of the desktop computer has been extended with a context menu, the tabletop runs a dedicated interface called PimVis and the ring binders have been augmented with integrated LEDs and contain documents augmented with a fiducial marker. The extended file system allows users to open a document located in the Windows File Explorer in the PimVis interface. The PimVis interface has been built to manage a user's personal information and is one of the contributions of this thesis. It has been constructed based on the OC2 framework and inspired by related work in the field of information visualisation. The PimVis interface can be used to locate a digital document in the extended file system or to find a physical document in the augmented ring binders. The papers located in the ring binders can be opened in the PimVis interface by placing them under the desk-mounted camera, which is used to recognise the fiducial marker. This fluid interaction between the different digital and physical user interfaces is managed via the XUI framework.

In order to evaluate our Office of the Future setting, we further conducted a user study. The user study has been done to control whether or not the fluid interaction between the different user interfaces of our setup grants a significant benefit opposed to the setup without this interaction. Participants of the study had to fill in a user experience questionnaire after performing a series of tasks, first without the fluid interaction then with the integration of interactions between the user interfaces. Furthermore, the candidates participated in a semi-structured interview to get qualitative results. Using the data from the questionnaires, different user experience factors were measured in both setups, by applying the t-test to data which was normally distributed and the Wilcoxon Signed-Rank test to non-normally distributed data. We could deduce from the quantitative data that there was no significant difference between the two setups. However, from the semi-structured interview with each participant, qualitative data could be collected. The qualitative results were very interesting, since every participant preferred the setup with the fluid interactions. Last but not least, we used the remarks and suggestions from the participants to define a few guidelines for developers of future fluid cross-media and cross-device interaction spaces.

Great Design Always Means
Great Style

— Don Norman

Appendix A: Style Guide Definition

A.1 Standard Window Layout

- Windows
  - Should contain 7 components or less.
  - Should always be horizontal.
  - Only show essential information.
- Context view
  - Should give a clear overview of the users' contexts.
  - Should be easily accessible from within other views.
- Document view
  - Should give a clear overview of the users' files and documents in a certain context.
  - Should clearly indicate within which context the user is situated.
  - Should be accessible from the *context view* and *focus view*.

- Focus view
  - Should give the user a clear and detailed overview of a specific document together with its related documents.
  - Should indicate on which document the focus is put on.
  - Should provide an easy way to go back to the other views.
  - Should be accessible from the *document view*.
- Help windows
  - Button or link to help pop-up window should be clearly indicated and stand out from the rest of the user interface.
  - Should be easily accessed.
  - Should be easily exited.
- Error windows
  - Should be presented in the centre of the screen.
  - Should only contain a confirmation button that closes the error window.

A.2 Standard Menus and Push Buttons

- Navigation buttons
  - Should be labelled with a comprehensive and uniform label, which indicate the buttons' destination view.
  - Should always be clearly visible.
  - Should always be of a reasonable size i.e. at least three times the average fingertip surface
- Help buttons
  - Should always be clearly visible.
  - Should contain either the *?* or *i* label
- Dialog buttons
  - Should be labelled explicitly depending on the dialog text
  - Should have an option for confirmation and negation, optionally a cancellation when necessary

A.3 Standards for Use of Window Controls, and Mapping of Data Types to Window Controls

- Input fields should be drop down boxes if the value is not free-form.
- Input fields should not allow improper value types.
- If applicable, an input field should auto complete.
- If applicable, when using an input field a virtual keyboard should appear.

A.4 Standard Use of Colour, Type, Fonts

- Standard colour scheme with teal as basic colour:
  In RGB:

Figure A.1: Basic colour scheme

Colour 1: (178, 223, 219)	Colour 6: (89, 150, 142)
Colour 2: (160, 208, 204)	Colour 7: (71, 135, 126)
Colour 3: (142, 194, 188)	Colour 8: (53, 121, 111)
Colour 4: (125, 179, 173)	Colour 9: (36, 106, 95)
Colour 5: (107, 165, 157)	Colour 10: (18, 92, 80)

- Standard colour for highlighted items is the amber colour:
  In RGB:

Figure A.2: Highlighted colour

Colour: (255, 193, 7)

- Standard colour scheme for other items:
  In RGB:

Figure A.3: Additional colour scheme

Colour 1: (0, 92, 49)
Colour 2: (0, 51, 128)
Colour 3: (194, 0, 136)
Colour 4: (158, 157, 36)
Colour 5: (216, 67, 21)

- Standard title font
  - 36 pt
  - Arial
- Standard content font
  - 18 pt
  - Verdana
- Hyperlinks should always be in blue and underlined

A.5 Standard Representations for Common User Objects

- Documents should be displayed via their example preview if it is appropriate to do so.
- Documents' types should be displayed by the corresponding thumbnail.
- Documents' context should be visible via the bubble colours in the *bubble layout* or via a context explorer in the *graph layout*.
- Documents' tags should be displayed via the tag cloud

B

Appendix B: User Experience Questionnaire

Please make your evaluation now.
For the assessment of the product, the Office of the Future, please fill out the following questionnaire. The questionnaire consists of pairs of contrasting attributes that may apply to the product. The circles between the attributes represent gradations between the opposites. You can express your agreement with the attributes by ticking the circle that most closely reflects your impression.

Example:

attractive	○	⊗	○	○	○	○	○	unattractive

This response would mean that you rate the application as more attractive than unattractive.

Please decide spontaneously. Don’t think too long about your decision to make sure that you convey your original impression.

Sometimes you may not be completely sure about your agreement with a particular attribute or you may find that the attribute does not apply completely to the particular product. Nevertheless, please tick a circle in every line.

It is your personal opinion that counts. Please remember: there is no wrong or right answer!

Please assess the product now by ticking one circle per line.

	1	2	3	4	5	6	7		
1 - annoying	○	○	○	○	○	○	○	enjoyable	1
2 - annoying	○	○	○	○	○	○	○	enjoyable	1
1 - not understandable	○	○	○	○	○	○	○	understandable	2
2 - not understandable	○	○	○	○	○	○	○	understandable	2
1 - creative	○	○	○	○	○	○	○	dull	3
2 - creative	○	○	○	○	○	○	○	dull	3
1 - easy to learn	○	○	○	○	○	○	○	difficult to learn	4
2 - easy to learn	○	○	○	○	○	○	○	difficult to learn	4
1 - valuable	○	○	○	○	○	○	○	inferior	5
2 - valuable	○	○	○	○	○	○	○	inferior	5
1 - boring	○	○	○	○	○	○	○	exciting	6
2 - boring	○	○	○	○	○	○	○	exciting	6
1 - not interesting	○	○	○	○	○	○	○	interesting	7
2 - not interesting	○	○	○	○	○	○	○	interesting	7
1 - unpredictable	○	○	○	○	○	○	○	predictable	8
2 - unpredictable	○	○	○	○	○	○	○	predictable	8
1 - fast	○	○	○	○	○	○	○	slow	9
2 - fast	○	○	○	○	○	○	○	slow	9
1 - inventive	○	○	○	○	○	○	○	conventional	10
2 - inventive	○	○	○	○	○	○	○	conventional	10
1 - obstructive	○	○	○	○	○	○	○	supportive	11
2 - obstructive	○	○	○	○	○	○	○	supportive	11
1 - good	○	○	○	○	○	○	○	bad	12
2 - good	○	○	○	○	○	○	○	bad	12
1 - complicated	○	○	○	○	○	○	○	easy	13
2 - complicated	○	○	○	○	○	○	○	easy	13
1 - unlikable	○	○	○	○	○	○	○	pleasing	14
2 - unlikable	○	○	○	○	○	○	○	pleasing	14

1 - usual	○	○	○	○	○	○	○	leading edge	15
2 - usual	○	○	○	○	○	○	○	leading edge	15
1 - unpleasant	○	○	○	○	○	○	○	pleasant	16
2 - unpleasant	○	○	○	○	○	○	○	pleasant	16
1 - secure	○	○	○	○	○	○	○	not secure	17
2 - secure	○	○	○	○	○	○	○	not secure	17
1 - motivating	○	○	○	○	○	○	○	demotivating	18
2 - motivating	○	○	○	○	○	○	○	demotivating	18
1 - meets expectations	○	○	○	○	○	○	○	does not meet expectations	19
2 - meets expectations	○	○	○	○	○	○	○	does not meet expectations	19
1 - inefficient	○	○	○	○	○	○	○	efficient	20
2 - inefficient	○	○	○	○	○	○	○	efficient	20
1 - clear	○	○	○	○	○	○	○	confusing	21
2 - clear	○	○	○	○	○	○	○	confusing	21
1 - impractical	○	○	○	○	○	○	○	practical	22
2 - impractical	○	○	○	○	○	○	○	practical	22
1 - organized	○	○	○	○	○	○	○	cluttered	23
2 - organized	○	○	○	○	○	○	○	cluttered	23
1 - attractive	○	○	○	○	○	○	○	unattractive	24
2 - attractive	○	○	○	○	○	○	○	unattractive	24
1 - friendly	○	○	○	○	○	○	○	unfriendly	25
2 - friendly	○	○	○	○	○	○	○	unfriendly	25
1 - conservative	○	○	○	○	○	○	○	innovative	26
2 - conservative	○	○	○	○	○	○	○	innovative	26

Bibliography

[1] Christopher Ahlberg and Ben Shneiderman. Visual Information Seeking Using the FilmFinder. In *Proceedings of CHI 1994, Conference on Human Factors in Computing Systems*, pages 433–434, Boston, USA, April 1994.

[2] Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. Dynamic Queries for Information Exploration: An Implementation and Evaluation. In *Proceedings of CHI 1992, Conference on Human Factors in Computing Systems*, pages 619–626, Monterey, USA, May 1992.

[3] Jakob Bardram, Sofiane Gueddana, Steven Houben, and Søren Nielsen. ReticularSpaces: Activity-Based Computing Support for Physically Distributed and Collaborative Smart Spaces. In *Proceedings of CHI 2012, Conference on Human Factors in Computing Systems*, pages 2845–2854, Austin, USA, May 2012.

[4] Jakob Bardram, Steven Houben, Søren Nielsen, and Sofiane Gueddana. 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*, pages 269–274, Copenhagen, Denmark, June 2012.

[5] Jacob Biehl and Brian Bailey. ARIS: An Interface for Application Relocation in an Interactive Space. In *Proceedings of GI 2004, Conference on Graphics Interface*, pages 107–116, London, Canada, May 2004.

[6] Richard Boardman and Angela Sasse. "Stuff Goes into the Computer and Doesn't Come Out": A Cross-tool Study of Personal Information Management. In *Proceedings of CHI 2004, Conference on Human Factors in Computing Systems*, pages 583–590, Vienna, Austria, April 2004.

[7] Andrew Bragdon, Robert DeLine, Ken Hinckley, and Meredith Ringel Morris. Code Space: Touch + Air Gesture Hybrid Interactions for Supporting Developer Meetings. In *Proceedings of ITS 2011, International*

Conference on Interactive Tabletops and Surfaces, pages 212–221, Kobe, Japan, November 2011.

[8] Nick Braisby and Angus Gellatly. *Cognitive Psychology*. Oxford University Press, 2012.

[9] Vannevar Bush and Jingtao Wang. As We May Think. *Atlantic Monthly*, 176(1):101–108, 1945.

[10] Tsung-Hsiang Chang and Yang Li. 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*, pages 2163–2172, Vancouver, Canada, May 2011. ACM.

[11] Michael Coen. Design Principles for Intelligent Environments. In *Proceedings of AAAI/IAAI 1998, Conference on Artificial Intelligence and Innovative Applications of Artificial Intelligence*, pages 547–554, Madison, USA, July 1998.

[12] Paul Dietz and Darren Leigh. DiamondTouch: A Multi-User Touch Technology. In *Proceedings of UIST 2001, Symposium on User Interface Software and Technology*, pages 219–226, Orlando, USA, November 2001.

[13] Alex Endert, Patrick Fiaux, and Chris North. Semantic Interaction for Visual Text Analytics. In *Proceedings of CHI 2012, Conference on Human Factors in Computing Systems*, pages 473–482, Austin, USA, May 2012.

[14] Katherine Everitt, Meredith Morris, Bernheim Brush, and Andrew Wilson. Docudesk: An Interactive Surface for Creating and Rehydrating Many-to-Many Linkages among Paper and Digital Documents. In *Proceedings of Tabletop 2008, International Workshop on Tabletops and Interactive Surfaces*, pages 25–28, Amsterdam, Netherlands, October 2008.

[15] Katherine Everitt, Chia Shen, Kathy Ryall, and Clifton Forlines. MultiSpace: Enabling Electronic Document Micro-mobility in Table-Centric, Multi-Device Environments. In *Proceedings of Tabletop 2006, International Workshop on Horizontal Interactive Human-Computer Systems*, pages 27–34, Adelaide, Australia, Januari 2006.

[16] Kim Fairchild, Steven Poltrock, and George Furnas. SemNet: Three Dimensional Graphic Representation of Large Knowledge Bases. *Cognitive Science and Its Applications for Human-computer Interaction*, pages 201–233, 1988.

[17] Scott Fertig, Eric Freeman, and David Gelernter. Lifestreams: An Alternative to the Desktop Metaphor. In *Proceedings of CHI 1996, Conference on Human Factors in Computing Systems*, pages 410–411, Vancouver, Canada, April 1996.

[18] Martin Fischer, Maureen Stone, Kathleen Liston, John Kunz, and Vibha Singhal. Multi-Stakeholder collaboration : The CIFE iRoom. In *Proceedings of CIB 2002, Conference on Distributing Knowledge in Building*, pages 12–14, Aarhus, Denmark, June 2002.

[19] Blaz Fortuna, Marko Grobelnik, and Dunja Mladenic. Visualization of Text Document Corpus. *Informatica*, 29(4):497–502, 2005.

[20] Luca Frosini, Marco Manca, and Fabio Paternò. A Framework for the Development of Distributed Interactive Applications. In *Proceedings of EICS 2013, Symposium on Engineering Interactive Computing Systems*, pages 249–254, London, United Kingdom, June 2013.

[21] George Furnas. Generalized Fisheye Views. In *Proceedings of CHI 1986, Conference on Human Factors in Computing Systems*, pages 16–23, Boston, USA, April 1986.

[22] Elisa Giaccardi. Cross-media Interaction for the Virtual Museum. *New Heritage: New Media and Cultural Heritage*, pages 112–131, 2008.

[23] Peter Hamilton and Daniel Wigdor. Conductor: Enabling and Understanding Cross-Device Interaction. In *Proceedings of CHI 2014, Conference on Human Factors in Computing Systems*, pages 2773–2782, Toronto, Canada, April-May 2014.

[24] Jeffrey Heer and Danah Boyd. Vizster: Visualizing Online Social Networks. In *Proceedings of InfoVis 2005, Symposium on Information Visualization*, pages 32–39, Minneapolis, USA, October 2005.

[25] Tomoharu Iwata, Takeshi Yamada, and Naonori Ueda. Probabilistic Latent Semantic Visualization: Topic Model for Visualizing Documents. In *Proceedings of KDD 2008, International Conference on Knowledge Discovery and Data Mining*, pages 363–371, Las Vegas, USA, August 2008.

[26] Matthew Jervis and Masood Masoodian. SOPHYA: A System for Digital Management of Ordered Physical Document Collections. In *Proceedings of TEI 2010, Conference on Tangible, Embedded, and Embodied Interaction*, pages 33–40, Cambridge, USA, Januari 2010.

[27] Hans-Christian Jetter, Michael Zöllner, Jens Gerken, and Harald Reiterer. Design and Implementation of Post-WIMP Distributed User Interfaces with ZOIL. *International Journal of Human-Computer Interaction*, 28(11):737–747, 2012.

[28] Ninad Jog and Ben Shneiderman. Starfield Visualization with Interactive Smooth Zooming. In *Proceedings of IFIP 1995, Working Conference on Visual Database Systems*, pages 3–14, Lausanne, Switzerland, March 1995.

[29] Brad Johanson, Armando Fox, and Terry Winograd. The Interactive Workspaces Project : Experiences with Ubiquitous Computing Rooms. *Pervasive Computing*, 1(2):67–74, 2002.

[30] Brad Johanson, Greg Hutchins, Terry Winograd, and Maureen Stone. PointRight : Experience with Flexible Input Redirection in Interactive Workspaces. In *Proceedings of UIST 2002, Symposium on User Interface Software and Technology*, pages 227–234, Paris, France, October 2002.

[31] Jeff Johnson, Teresa Roberts, William Verplank, David Canfield Smith, Charles Irby, Marian Beard, and Kevin Mackey. The Xerox Star: A Retrospective. *Computer*, 22(9):11–29, 1989.

[32] William Jones. Finders, Keepers? The Present and Future Perfect in Support of Personal Information Management. *First Monday*, 9(3), 2004.

[33] William Jones. *Keeping Found Things Found: The Study and Practice of Personal Information Management.* Morgan Kaufmann, 2010.

[34] Jiwon Kim, Steven Seitz, and Maneesh Agrawala. Video-Based Document Tracking: Unifying your Physical and Electronic Desktops. *Proceedings of UIST 2004, Symposium on User Interface Software and Technology*, pages 99–107, October 2004.

[35] Scott Klemmer, Mark Newman, Ryan Farrell, Mark Bilezikjian, and James Landay. The Designers' Outpost: A Tangible Interface for Collaborative Web Site Design. In *Proceedings of UIST 2001, Symposium*

on User Interface Software and Technology, pages 1–10, Orlando, USA, November 2001.

[36] Xia Lin. Visualization for the Document Space. In *Proceedings of Visualization 1992*, pages 274–281, Boston, USA, October 1992.

[37] Paul Luff and Christian Heath. Mobility in Collaboration. In *Proceedings of CSCW 1998, Conference on Computer Supported Cooperative Work*, pages 305–314, Seattle, USA, 1998.

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

[39] Nicolai Marquardt, Ken Hinckley, and Saul Greenberg. Cross-Device Interaction via Micro-mobility and F-formations. In *Proceedings of UIST 2012, Conference on User Interface Software and Technology*, pages 13–22, Cambridge, USA, October 2012.

[40] Jérémie Melchior and Donatien Grolaux. A Toolkit For Peer-to-Peer Distributed User Interfaces: Concepts, Implementation, and Applications. In *Proceedings of EICS 2009, Symposium on Engineering Interactive Computing Systems*, pages 69–78, Pittsburgh, USA, July 2009.

[41] Moira Norrie. An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In *Proceedings of ER 1993, Conference on the Entity-Relationship Approach*, pages 390–401, Arlington, USA, December 1993.

[42] Peter Pirolli and Stuart Card. The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis. In *Proceedings in International Conference on Intelligence Analysis*, pages 2–4, McLean, USA, May 2005.

[43] Roman Rädle, Hans-christian Jetter, Nicolai Marquardt, Harald Reiterer, and Yvonne Rogers. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. In *Proceedings of ITS 2014, International Conference on Interactive Tabletops and Surfaces*, pages 45–54, Dresden, Germany, November 2014.

[44] David Redmond-Pyle and Alan Moore. *Graphical User Interface Design and Evaluation (guide): A Practical Process*. Prentice Hall, 1995.

[45] Abigail Sellen and Richard Harper. *The Myth of the Paperless Office*. MIT press, 2002.

[46] Chia Shen, Katherine Everitt, and Kathleen Ryall. UbiTable: Impromptu Face-to-Face Collaboration on Horizontal Interactive Surfaces. In *Proceedings of UbiComp 2003, International Conference on Ubiquitous Computing*, pages 281–288, Seattle, USA, October 2003.

[47] Ben Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, 1983.

[48] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of VL 1996, Symposium on Visual Languages*, pages 336–343, Boulder, USA, September 1996.

[49] Beat Signer and Moira Norrie. As We May Link: A General Metamodel for Hypermedia Systems. In *Proceedings of ER 2007, International Conference on Conceptual Modeling*, pages 359–374, Auckland, New Zealand, November 2007.

[50] David Smith, Charles Irby, Ralph Kimball, and Eric Harslem. The Star User Interface: An Overview. In *Proceedings of AFIPS 1982, National Computer Conference*, pages 515–528, Houston, USA, June 1982.

[51] David Smith, Frank Ludolph, Charles Irby, and Jeff Chairman-Johnson. The Desktop Metaphor As an Approach To User Interface Design. In *Proceedings of the Annual Conference on The Range of Computing: Mid-80's Perspective*, pages 548–549, Denver, USA, October 1985.

[52] Jürgen Steimle, Oliver Brdiczka, and Max Muhlhauser. CoScribe: Integrating Paper and Digital Documents for Collaborative Knowledge Work. *IEEE Transactions on Learning Technologies*, 2(3):174–188, 2009.

[53] Jürgen Steimle, Mohammadreza Khalilbeigi, Max Mühlhäuser, and James D Hollan. Physical and Digital Media Usage Patterns on Interactive Tabletop Surfaces. In *Proceedings of ITS 2010, International Conference on Interactive Tabletops and Surfaces*, pages 167–176, Saarbrücken, Germany, November 2010.

[54] Norbert Streitz, Jiirg Geibler, Torsten Holmer, Christian Muller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-LAND: An Interactive Landscape for Creativity and Innovation. In *Proceeding of CHI 1999, Conference on Human Factors in Computing Systems*, pages 120–127, Pittsburgh, USA, May 1999.

[55] Norbert Streitz, Thorsten Prante, Christian Müller-Tomfelde, Peter Tandler, and Carsten Magerkurth. Roomware: The Second Generation. In *Proceedings of CHI 2002, Conference on Human Factors in Computing Systems*, pages 506–507, Minneapolis, USA, April 2002.

[56] Peter Tandler, Thorsten Prante, Christian Müller-tomfelde, Norbert Streitz, and Ralf Steinmetz. ConnecTables : Dynamic Coupling of Displays for the Flexible Creation of Shared Workspaces. In *Proceedings of UIST 2001, Symposium on User Interface Software and Technology*, pages 11–20, Orlando, USA, November 2001.

[57] Sandra Trullemans. Personal Cross-Media Information Management. Master's thesis, Vrije Universiteit Brussel Belgium, 2013.

[58] Sandra Trullemans and Beat Signer. From User Needs to Opportunities in Personal Information Management: A Case Study on Organisational Strategies in Cross-Media Information Spaces. In *Proceedings of DL 2014, International Conference on Digital Libraries*, pages 87–96, London, UK, September 2014.

[59] Sandra Trullemans and Beat Signer. Towards a Conceptual Framework and Metamodel for Context-Aware Personal Cross-Media Information Management Systems. In *Proceedings of ER 2014, International Conference on Conceptual Modelling*, pages 313–320, Atlanta, USA, October 2014.

[60] Florian van de Camp and Rainer Stiefelhagen. GlueTK: A Framework for Multi-Modal, Multi-Display Human-Machine-Interaction. In *Proceedings of IUI 2013, International Conference on Intelligent User Interfaces*, pages 329–338, Santa Monica, USA, March 2013.

[61] Pierre Wellner. Interacting with Paper on the DigitalDesk. *Communications of the ACM*, 36(7):87–96, 1993.

[62] Christopher Williamson and Ben Shneiderman. The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System. In *Proceedings of IR 1992, Conference on Research and Development in Information Retrieval*, pages 338–346, Copenhagen, Denmark, June 1992.

[63] James Wise, James Thomas, Kelly Pennock, David Lantrip, Marc Pottier, Anne Schur, and Vern Crow. Visualizing the Non-Visual: Spatial Analysis and Interaction with Information from Text Documents. In

Proceedings of INFOVIS 1995, Symposium on Information Visualization, pages 51–58, Atlanta, USA, October 1995.