Vrije Universiteit Brussel

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

End User Control of Dynamic Distributed User Interfaces

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

Jasmien De Ridder

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

VRIJE UNIVERSITEIT BRUSSEL • SCIENTIA VINCERE TENEBRAS •

Academic year 2014-2015

Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN

VAKGROEP COMPUTERWETENSCHAPPEN

End User Control of Dynamic Distributed User Interfaces

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

Jasmien De Ridder

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

Academiejaar 2014-2015

Abstract

The office environment has changed a lot over the last decades with the introduction of new technologies like smartphones, tablets, smartwatches and so on. The office environment is becoming a dynamic place, where people work cross-media and cross-devices. People tend to use different devices in their daily life. For example, John is working and browses through his emails on his laptop and saves an email about a meeting later that day. Later that day, before the meeting, he opens the same email on his mobile phone and reads it while walking towards the meeting. User interfaces or its components are distributed across multiple devices. These user interfaces are predefined by the developers or UI designers or the applications. They do not allow end users to customise the user interface or its components in any way.

The office of the future will definitely include distributed user interfaces. However, these are not the only user interfaces that have found their way to the office of the future. Augmented reality user interfaces are being extensively researched in the field of Personal Information Management, where the main focus lies on helping people to keep, re-find and organise their information. For example, augmenting a bookshelf with LEDs to indicate where certain books or folders are located. But also in the digital information space, augmented user interfaces were introduced that help a user re-find their files in the file hierarchy system. These augmented user interfaces have the same problem as distributed user interfaces. They are not customisable by end users, but are predefined by the developers of the application.

However, allowing end users to customise the augmented user interfaces to their own taste, would improve their daily work activities in office settings. In order to allow users to configure when and where the augmented reality user interfaces can be used, we developed the DUI2 framework. We will combine our framework with the Context Modelling Toolkit (CMT), a context-aware framework, in order to facilitate the configuration process. Our framework will allow users to define rules in order to state that a certain augmented reality user interface can be used/shown depending on the context. This approach allows for an easier, more natural way to interact with the framework, making it possible for end-user with no programming skills to use it. Besides the user customisation, we will allow to automatically adapt the user interface based on the current context. For example, when a user is in their office reading a paper. The paper is automatically augmented with a corresponding user interface. To illustrate our novel approaches of the customisation and automatic adaptation of augmented reality user interfaces,

we will present a use case of the framework.

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 girlfriend Aurélie and my family for their moral support during the thesis. Secondly I would like to thank my supervisor Sandra Trullemans for her guidance during the year. I would also like to thank my promoter, professor Beat Signer for his feedback during the follow-up presentations.

Lastly I would like to thank my thesis colleagues: Audrey, Ayrton and Tim.

Contents

1 Introduction

1.1 Distributed User Interfaces 1
1.2 Augmented Reality 2
1.3 Context-Awareness 4
1.4 Problem Statement 5
1.5 Thesis Contribution 6
1.6 Thesis Outline 7

2 Background

2.1 Distributed User Interfaces 9
2.1.1 Multi-Device Design Tools 15
2.2 Augmented Reality User Interfaces 17
2.3 Context-Aware User Interfaces 23
2.4 Where Is The Gap? 24
2.5 Conclusion 24

3 System and User Control

3.1 Introduction to Automatic Adaptation 28
3.1.1 Rule-based Context-aware Frameworks 29
3.2 UI4A2: UI Context Model for Automatic Adaptation 30
3.2.1 Physical vs. Digital Information Space 30
3.2.2 Type of Information Items 30
3.2.3 Location of the Augmentation 30
3.2.4 Techniques 32
3.2.5 Modalities 33
3.2.6 Devices and Technologies 33
3.2.7 Overview 34
3.3 End User Control 36
3.3.1 End User Programming 36
3.4 Requirements 39
3.4.1 Automatic Adaptation 39
3.4.2 User Customisable Adaptation 40

4 DUI2 Framework
4.1 Architecture . . . 41
4.2 MUI Framework . . . 43
4.2.1 MUI Model . . . 43
4.2.2 Object-Concept-Context (OC2) Model . . . 45
4.2.3 Mappings . . . 47
4.3 Context Modelling Toolkit . . . 51
4.3.1 CMT Architecture . . . 51
4.3.2 Idea of Templates . . . 51
4.3.3 Client Side . . . 53
4.3.4 Server Side . . . 53
4.4 DUI2 . . . 53
4.4.1 Functionality . . . 54
4.4.2 Automatic Adaptation . . . 54
4.4.3 User Adaptation . . . 55

5 Configuration and End User Tool
5.1 Use Case . . . 57
5.2 Configuration Tool . . . 58
5.3 Modelling Tool . . . 61
5.3.1 Modelling a Rule . . . 62
5.3.2 Android Implementation Details . . . 63
5.3.3 DUI2 Implementation Details . . . 64

6 Conclusions and Future Work
6.1 Contributions . . . 68
6.2 Future Work . . . 69

List of Figures

1.1 Frame of the cartoon film: *Who Framed Roger Rabbit* 3
1.2 Example of Augmented reality, where a building is augmented 3

2.1 The natural interaction mechanism of Deep Shot 10
2.2 The multi-device environment for the cross-device interactions of Conductor 12
2.3 The THAW system with near-surface interaction 12
2.4 The ReticUI user interface of ReticularSpaces; on the left the activity view and on the right the action view 13
2.5 Different ZOIL user interfaces for a physical lens on a tabletop 14
2.6 WYSIWYG approach of Gummy 16
2.7 The pile browser to digitally browse through a physical pile . . 18
2.8 Interactive Bookshelf Surface 19
2.9 MapLens 20
2.10 TagTree 21
2.11 Icon Highlights 22
2.12 WikiFolders 22

3.1 Setup of PeriTop 34
3.2 if-then rule-based approach of IFTTT recipes 37
3.3 The Reality Editor that supports editing of behaviour and interfaces of Smart Objects 38

4.1 The architecture of the DUI2 framework 42
4.2 The MUI model for user interface management 44
4.3 OC2 extension of the RSL metamodel 46
4.4 The context extension of the OC2 framework 47
4.5 personIsInLocation template example 52

5.1 Sequence diagram for registering an augmented user interface . 60
5.2 User interface of configuration tool 61
5.3 User interface of the Android application 62
5.4 Sequence diagram for creating a rule 65

List of Tables

3.1 Applications operating on the physical or digital information space 31
3.2 Overview of the applications with the corresponding categories 35

4.1 Mapping of the MUI components onto the OC2 components . 47
4.2 List of end points supported by our DUI2 framework 54
4.3 Different categories from the UI4A2 context model 55

1 Introduction

In the last decades, the office environment has changed drastically. An office used to contain a desk with a desktop computer and sometimes a few bookcases or a file cabinet. However, recent technologies introduced laptops, smartphones, tablets, tabletops and so on to the office. The office environment is becoming more dynamically, working across media, across devices and across user interfaces and interactions. Nowadays user interfaces can be distributed across different devices, like laptops and tablets.

1.1 Distributed User Interfaces

Various devices appear in the office environment and in order to deal with the interactions across them, the field of distributed user interfaces emerged. This field focuses on the distribution of the state of an application and its content across devices. Niklas Elmqvist [16] describes a distributed user interface or DUI as:

> *A distributed user interface is a user interface whose components are distributed across one or more of the dimensions:*
>
> - ***Input:*** *Managing input on a single computational device, or distributed across several different devices (so-called input redirection)*

- ***Output:*** *Graphical output tied to a single device (display), or distributed across several devices (so-called display or content redirection)*
- ***Platform:*** *The interface executes on a single computing platform, or distributed across different platforms (i.e. architectures, operating systems, networks, etc.)*
- ***Space:*** *The interface is restricted to the same physical (and geographic) space, or can be distributed geographically (i.e. co-located or remote interactive spaces)*
- ***Time:*** *Interface elements execute simultaneously (synchronously), or distributed in time (asynchronously)*

A well-known example of a distributed user interface, is the "YeloTV" application from Telenet. This application allows users to watch TV shows not only on their TV, but also on their smartphones and tablets. The user interface for watching television is distributed across different devices and platforms. Other examples allow to transfer content or application state to other devices, QR codes are already very integrated into our daily lives. For example, Bob is searching information about his next city trip to Paris, he is browsing the web, when an Android application is suggested and a QR code is given. Bob takes his smartphone and scans the QR code, the application is automatically downloaded and opened. Bob can continue planning his trip to Paris by using his smartphone.

1.2 Augmented Reality

How the office of the future will look like, has already been extensively researched . Current research focused on augmenting the office environment by using Augmented Reality techniques. Different augmented user interfaces were developed to help users in their daily work activities. All of them have the basic goal to enhance the user's perception of the real world and help them interacting with it. In 1997, Azuma [1] described a definition that today is still the most widely used. The definition goes as follows:

> *Augmented Reality systems are systems that have the following three characteristics:*
>
> - *Combines real and virtual*
> - *Interactive in real-time*
> - *Registered in 3-D*

If we however think about cartoon films, where cartoon characters are added to a film set. We can argue that this is also a kind of Augmented Reality,

Figure 1.1: Frame of the cartoon film: *Who Framed Roger Rabbit*

because a representation of the reality is being augmented. An example of a cartoon film is shown in Figure 1.1. If we compare our cartoon films with the definition of Azuma, we can see that it indeed combines virtual and real, but it is not in 3D and not in real-time.

When referring to Augmented Reality in this thesis, we will extend the definition of Azuma to include 2D objects and pre-processed augmentation (i.e. no real-time). Augmented Reality is often associated with games, because it allows to enrich the experience of playing a game. For example, board games such as Monopoly are being augmented by tracking the pawns which are markers with Computer Vision techniques. By using a the camera of a smartphone to hover over the board, the pawns get augmented and figures come to life on the smartphone screen.

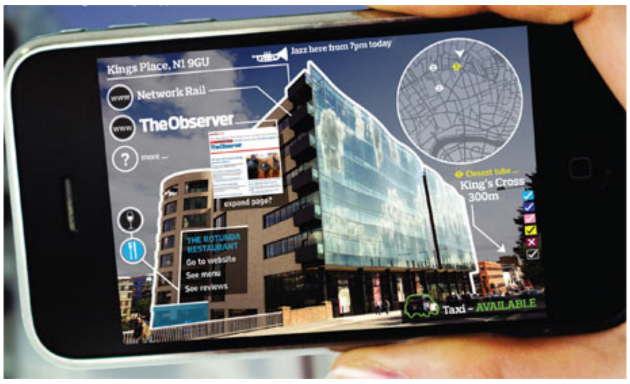


Figure 1.2: Example of Augmented reality, where a building is augmented

Besides gaming applications, more recent applications focus on overlaying real-time information on a smart phone screen. In Figure 1.2, the reality is being augmented by pointing the camera of a mobile phone towards it. For example, if a building is recognized, the corresponding information (i.e. restaurants close by, the street name and address, etc.) about the building is shown on the screen of the smartphone. In office settings, ring binders can be augmented with LEDs. When a user searches for a document, the LED of ring binder containing the document will light up and this allows users for quicker searches. The same principle of pointing a smartphone can be applied to books in office environments. But instead of pointing to buildings, the camera of the smart phone will be pointed to the books in the bookcase. The camera is able to track the spine of the books, because special markers are placed for recognition. Whenever a user needs a book, he can hover over the book spines and get additional information about the books. This technique allows the users to speed up the search process, since the user does not need to take all the books out from the shelf in order to see what they are about. Sometimes applications use a mix of augmented reality and augmented virtuality. For example, an entertainment magazine containing tags such that virtual elements are added to the magazine, only if viewed through the camera of a mobile phone. These kind of applications are called mixed reality or hybrid reality, where the physical and virtual world are merged together to create new visualisations.

1.3 Context-Awareness

Context-awareness refers to the paradigm in which applications can sense and explore the context of the users in order to provide appropriate and useful services towards users. These provided services are dynamic and are able to automatically adapt to the current needs of the users based on context changes. Some intelligence is implied in context-awareness, that enables applications to detect, reason, and predict the action to be taken in order to adapt in a dynamic environment. Most context-aware applications work without explicit user intervention. In smart homes and recommendation systems, context-aware frameworks are used to adapt the environment based on context changes. For example, when the temperature rises above 26°C, the blinds in front of the windows be automatically closed or the A/C will be turn on. In smart homes, these input signals are gathered by sensors and the context-aware system will compare the input with the available rules in order to perform corresponding actions. In the research for user interface design, context-awareness also has been explored. For example, the user interface of an iPad or tablet is able to adapt depending on how it is held by the users. By rotating and flipping the iPad, the orientation of the user interface is changed from landscape to portrait or vice a versa. The user interface itself changes/adapt according to the context of the orientation in order to show an appropriate view to the users. Different contextual information can be used as a base to adapt on, like time events, location, sensor input, etc. The adaptations of context-aware application are mostly automatically or dynamically performed by the system.

1.4 Problem Statement

The office of the future is a dynamic environment including cross-media, cross-device and cross-user interface interactions. A user interface can be distributed over various devices. Some user interfaces are adapted to one device, and others user interface components are allocated to another set of devices in order to provide more natural interactions. In such a distributed user interface setting the distribution is often done by the user interface designer. Even in large projects such as i-Land [62] which focus on the interactions of distributed user interfaces over tabletops, multiple large displays and collaborative workplaces, the developed user interfaces are not customisable by the end user. Besides these distributed user interfaces, Augmented Reality user interfaces have found their way to the Office of the Future setting. By augmenting physical documents through overhead projectors or adding LEDs in bookcases, researchers try to support users in their daily work activities.

Nevertheless, these augmented reality applications are again predefined by the developer. Although at first sight this may not be a great issue, research on descriptive Personal Information Management activities has observed that definitely in work-related activities users are very unique in their behaviour. Nevertheless, current systems do not allow users to configure their augmented workspace including distributed and Augmented Reality user interfaces. Just think about notifications that are automatically sent, when a user receives an e-mail. Some people hate it, when they are interrupted by the notification. This can, however, depend on the task at hand. It would be much more functional if a user is able to control the behaviour of the notifications. Secondly, context-aware applications are well-known in various domains such as smart-homes and recommendation systems. It is interesting to investigate the idea of context-aware user interfaces. Nevertheless, the end users should again be able to have control over these configurations. Considering the example of the notifications, in context-aware user interfaces we can include the context conditions to configure the user interface. For example, Bob is in a business meeting with his bosses and does not want to be interrupted. Here it would be preferable to configure that notifications should not be shown when in a meeting.

1.5 Thesis Contribution

In this thesis, we will investigate the opportunity to allow users to customise their user interface configurations in the Office of the Future setting, as well as automatically adapting the user interfaces depending on the current context. By doing so, we allow users to configure their own workspace that fits them best. Current systems only allow UI designers or developers to configure the user interfaces. Mostly the user interfaces are predefined and they do not provide any means for end user customisation. However, in order to increase user satisfaction and acceptance of systems, users should be given control over the behaviour of user interfaces. There needs to be a balance between the control that the system has and the control that is in the hands of the users.

In order to achieve this, we present the DUI2 framework that will allow to balance user and system control by providing two different approaches. The first approach will automatically adapt the user interface based on the current context. The second approach focuses on allowing end users to configure and customise their own user interface without needing any programming experience. In order to adapt and model the configurations for the user interfaces, we need a system that is context-aware. For this thesis, we combined our DUI2 framework with the Context Modelling Toolkit (CMT) framework. Besides the CMT framework, the Object-Concept-Context (OC2) framework will also be used as an underlying structure. The OC2 framework will allow to store and manage the user interfaces and the corresponding context conditions. In this thesis, the focus is set on distributed and augmented user interfaces. Nevertheless, the scope can be broadened to include various types of user interfaces. The DUI2 framework will be a generic framework for different types of user interfaces. Besides the DUI2 framework, we will also provide two applications that use the functionalities provided by the framework. The first tool is a configuration tool where developers will be able to register new augmented user interfaces and add them to the DUI2 framework. The second tool is the modelling tool where end users are able to construct rules for the user customisable adaptation.

1.6 Thesis Outline

In this thesis, we will first provide the necessary introduction to current research in office settings. We will explain distributed user interfaces and the tools that exist to author them. Then we will take a look at the augmented user interfaces that have found their way to the office of the future. The vision of the future and current shortcoming in user interface adaptation will be explained. In the next part, we will explained the two adaptation approaches that our framework will support in order to give a little control to the user. In Chapter 3, the automatic adaptation and user customisation will be explain in detail. Afterwards we will introduce the DUI2 framework with all of its modules and components that was implemented for the purpose of this thesis.

Finally, in Chapter 7 we will discuss the two applications that we developed for our DUI2 framework, namely the configuration tool and modelling tool. In the last chapter, we will reflect on the contributions of this thesis and we will discuss the further steps that can be taken to further improve and explore the capabilities of the framework.

2
Background

Since the increase in mobile devices, users utilise different devices to perform their daily activities. In the office setting, users can have mobile phones, laptops, desktop computers, tablets and so on, in order to complete their work-related tasks. All these different devices have different user interfaces to interact with. Switching from, for example, a laptop to a mobile phone requires some adaptation to the user interface in order to be usable. In recent research, the distribution of user interfaces across different devices, platforms and users was studied to cope with the dynamic office environment. However, distributed user interfaces are not the only type of user interfaces that are investigated in the Office of the Future setting, Augmented Reality and context-aware user interfaces are also being studied.

2.1 Distributed User Interfaces

In their daily life, users interact with different types of systems via multiple devices. The space where the interactions between a human and a computer happens, is called a user interface. It allows user to control the machine in an efficient way. Different types of user interfaces exist, all of them belong to either digital user interfaces or physical user interface. When talking about physical user interfaces, we refer to user interfaces where physical objects such as tangibles are used to control the computer in a real world setting.

Whereas digital user interface allow users to control a device by interacting with a digital user interface on the device itself.

In the field of Distributed User Interface (DUI) various research has taken place, where the focus lies on the distribution of the state of the application and its content across devices. In 2013, Frosini et al. [19] developed a framework that allows designers and developers to create applications, where the user interface components are partially or completely distributed across devices. Allowing for multiple instances of applications at the same time and for different types of devices and users. The framework imposes an environment called Distribution manager that takes care of the development of the applications and the management of the dynamic distribution by providing run-time support. The sets of devices organise themselves in order to support the dynamic distribution. For supporting the run-time distribution, no fixed server is required. A more decoupled framework was presented by Melchior et al. [45], where graphical user interfaces are distributed across four dimensions, namely multiple displays, multiple platforms, multiple operating systems and multiple users. A peer-to-peer solution is proposed, where by connecting multi-purpose proxies to one or more rendering engines, the whole or parts of the graphical user interface are rendered for any operating system, any computing platform and any display. The graphical user interface can be distributed as a whole or a subset consisting of widgets can be distributed across the same or various devices. For example, a watch face in a calendar window is distributed from that window to another window containing different functionality. The distribution and migration of user interfaces or its widgets can happen across devices, platforms, displays and users.

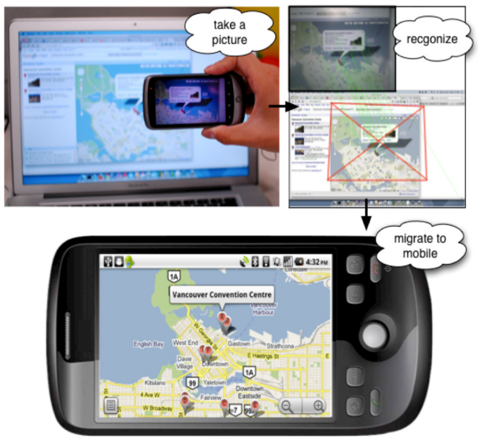


Figure 2.1: The natural interaction mechanism of Deep Shot

Some systems focus more on the interaction with distributed user interfaces, they try to make the interaction as natural as possible. An example of such a system is Deep Shot [8], which is a framework for migrating tasks across devices. When a user is performing a task, he often utilises multiple devices. For example, first he works on the PC in his office and later on he continues working on a laptop or mobile phone while taking public transportation. Currently there is no support for fluent, natural interactions between these multiple heterogeneous devices. However, Deep Shot addresses the lack of support by allowing users to take a picture of the application with their mobile phone to perform the transfer of the user's work state for a task. The Deep Shot provides two new interaction techniques, called deep shooting and deep posting. The first technique is deep shooting and is shown in Figure 2.1. A user can take a picture of the computer screen where an application is opened, in this case Google maps. Deep Shot recognizes the relevant region of the application (i.e. the map) that the user is looking at through the camera and will migrate that part of the application to the mobile phone with a recovered application state. Deep posting however, uses the same mechanism, the target screen and region needs to be identified, but it is not required to identify the application a user is looking at. Deep Shot provides two new interaction techniques, however interactions can take many forms. With an abundance of inexpensive connected devices available to a person at any time, there exist very few applications that allow users to take advantage of this large number of screens, as illustrated in Figure 2.2. In 2014, Hamilton and Wigdor [22] introduced a generic framework for cross-device interactions, named Conductor. It allows for various forms of interaction techniques across devices in order to enable easy transition between the different devices. The framework focuses on the shared interaction styles such as the use of NFC tags or bumping between mobile devices to transfer application states and content. Mechanisms for the chaining of devices (i.e. bonding with duets) in a workflow as well as for managing cross-relationships are provided.

Recent research pushes the vision of DUIs further by transferring specific functionality to the most appropriate device, the appropriate device. By doing so, these applications allow for a more natural way of interacting with the user interface. In the THAW system [39], a smart phone is handled as a tangible object for near-surface interactions with larger displays such as a computer screen, as illustrated in Figure 2.3. By handling the smartphone in combination with the larger displays, the system provides for a fluid interaction space. The THAW system only provides support in a certain range. If the smartphone is positioned too far away from the screen, additional

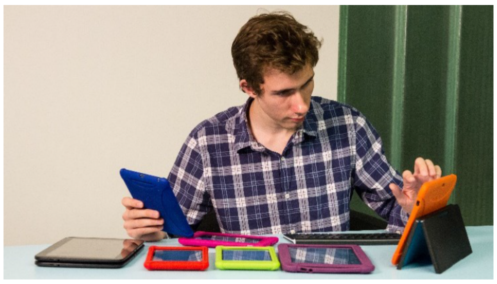


Figure 2.2: The multi-device environment for the cross-device interactions of Conductor

hardware is needed. The focus is on hovering and on-screen interaction. A server takes care of the communication with the incoming smartphones and exchanges the needed calibration and tracking information. Another approach where natural interaction is explored, is called WatchConnect [27]. The system allows a smartwatch to be used as an input device and output display. The WatchConnect toolkit allows developers to create cross-device applications and interactions. The sensor input of the smartwatch provides the trigger to, for example, change the position of images on a computer screen. The user interface can also be migrated/beamed to the larger display surface, for example, when a Skype call comes in.

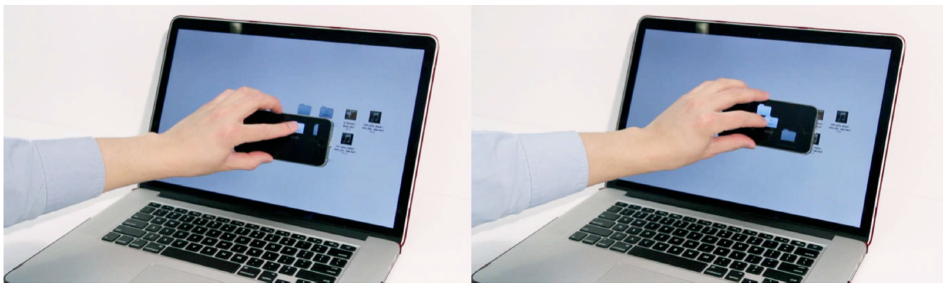


Figure 2.3: The THAW system with near-surface interaction

In smart room environments, designers of DUIs need to take into account more complexity, because multiple users, public displays and tangibles on tabletops can be present. Some fully integrated examples of smart rooms are iLand [62] and Interactive Workspaces/iRos [33]. These two smart room applications focus on communication between devices, on sharing application state through UI components and keeping track of multiple users. iLand describes an environment where information and communication technologies are integrated in room elements, such as walls, doors, and furnitures in order to support the collaboration of multiple users. By combining recent developed techniques from Augmented Reality and ubiquitous computing, the first attempt at a smart room was made, containing a electronic wall, interactive table and mobile, networked chairs with integrated interactive devices, such as displays. Interactive Workspaces/iRos developed different prototypes. The second generation prototype is iRoom, where three touch sensitive white boards and and a larger display that was designed to look like a conference table. iRos is the system infrastructure on which the devices with specific properties are tied together. Some specific room-based user interfaces were developed for the Interactive Workspaces project. Note that these large smart room projects provide a promising vision for the future of workspaces and office environments, however their focus is mainly on the collaborative aspect of workspaces using traditional desktop user interfaces.

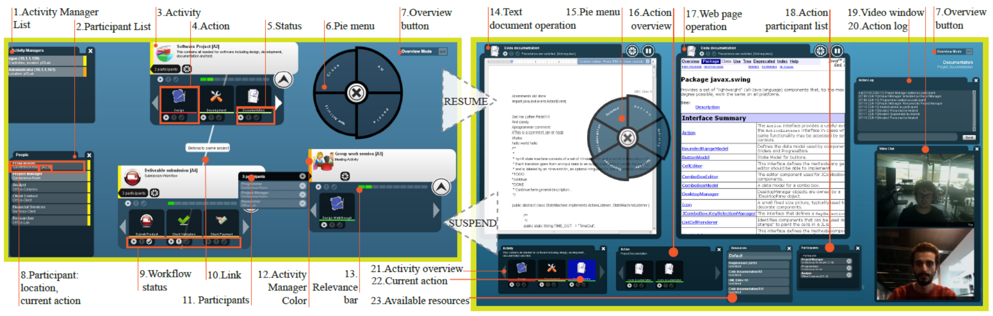


Figure 2.4: The ReticUI user interface of ReticularSpaces; on the left the activity view and on the right the action view

Bardram et al. [2] take another approach with their smart space framework, called ReticularSpaces. The framework provides a unified design to develop user interfaces for applications spanning the tasks of the user on different smart room devices and is implemented on a distributed Model-View-Controller (dMVC) architecture. ReticularSpaces provides an activity-based UI for smart spaces, called ReticUI. It is designed to unify the experience of the user. The user interface of ReticUI consists of two views, namely the Activity view and the Action View, as illustrated in Figure 2.4. Both views are designed to use on any type of display, even for large screens. The Activity view is the default view where the activity manager is shown. Users are able to access a display and mount an activity manager. The Activity view will then display the activities that belong to the mounted activity manager and are applicable to the current context. The Action view consists of a large palette where the operations that can be executed are displayed. The user interface for this smart room approach is develop beforehand and cannot be modified in any way, however depending on the display and activity manager only a custom set of activities is displayed.

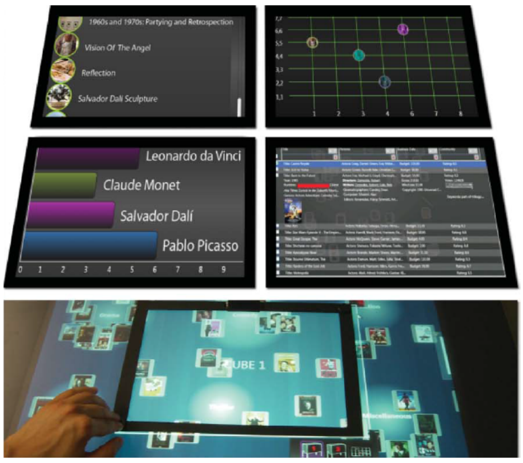


Figure 2.5: Different ZOIL user interfaces for a physical lens on a tabletop

Other systems such as glueTK [7] and ZOIL [32] provide functionality for post-WIMP interfaces in smart rooms, in order to provide more natural interactions. GlueTK is a framework that provides functionality to manage multimodal input in a whole room containing multiple devices. Each input modality has its own interactions and requirements. glueTK abstracts the complexity of these input modalities and allows for multi-device user interface design for different screen sizes. GlueTK takes a broad approach to input modalities and does not only focus on point and touch modalities. By doing so, the framework enables the design of custom applications specifically for interactive rooms, moving away from traditional desktop environments. The Office of the Future will most likely not only contain a desktop, but a range of different devices. The ZOIL framework takes a different approach to post-WIMP distributed user interfaces for interactive rooms/spaces. The framework is the most closely related framework to the presented work in this thesis. The Object-Oriented User Interface (OOUI) design paradigm is the start point of the Zoomable Object-Oriented Information Landscape (ZOIL) framework. This paradigm states that user interfaces are just “views” over data objects such as documents or just fragments of documents, emails and objects. In office settings, this approach allows more natural interaction between the user and the information space, because often users see email, documents as objects and in order to interact with them, tools are used.

Nevertheless, the ZOIL framework only supports the design of its own user interfaces, called ZOIL user interfaces, as shown in Figure 2.5. Six design principles were introduced, but the main focus is on visualisation techniques. The zoomable user interfaces are the most important technique to interact with the central data repository. A client-server architecture is implemented to provide transparent persistence and synchronisation. An object-oriented database (db4o) is used in the ZOIL data backend.

2.1.1 Multi-Device Design Tools

We have seen that it is very common for users to perform their tasks on multiple devices ranging from traditional desktops to mobile devices with various multimodal interactions. However, current user interfaces design tools target either a single device or a fixed set of devices, and they provide little support for user interfaces or their components that are distributed across multiple devices. Many different approaches have been studied for the authoring of multi-device user interfaces. For the development of multimodal Web interfaces, most authoring environments only allow to create a user interface out of predefined basic elements. Some direct manipulation techniques allow to easily create interfaces, but they do not provide any support to identify the most suitable user interface for supporting the targeted users and the tasks at hand, while taking into account the interaction space. A method and tool where developed to support the transformations of user interfaces across various interactive views. TERESA [5] allows designers to transform their description into a UI description, where the design decisions are incorporated. The format for the UI description is modality independent. The UI descriptive language format is then used to derive a modality dependent interface description from. This final description is used as a base for generating code to create the user interface. TERESA allows to define a user interface at an abstract level by using XML.

Although, MARIA and TERESA can be described as authoring tools for the underlying user interface descriptive languages, other examples that take a more natural approach exist to design cross-device user interfaces. Damask [40] and Gummy [46] are such examples which are closer to WYSIWYG (What You See Is What You Get). Damask is a prototyping tool that allows users to create multi-device web interfaces for desktops and mobile phones and has support for speech input and output. Designers can sketch their user interface for a specific device by using design patterns. These patterns are a fixed set of UI elements that are already optimised for each device. A layered approach is used to indicate which UI elements are common for all devices and which are specific to a particular device or modality. The sketch-based technique allows for easy generation of design for other devices, which can later on be refined. A study has shown that the use of patterns allows for easy creation. However some questions arise about scalability, because the tool only offers a limited set of design patterns.

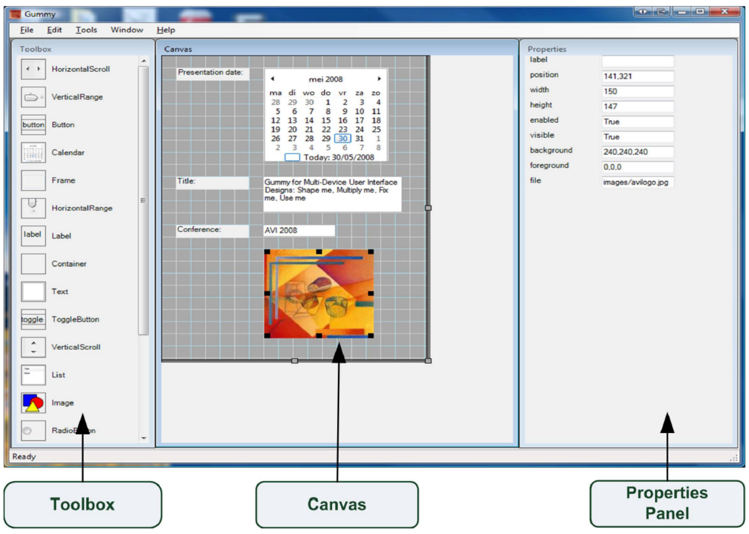


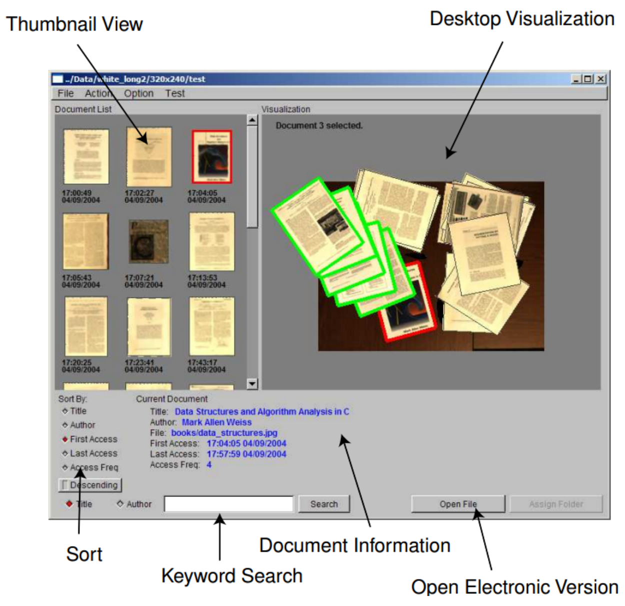
Figure 2.6: WYSIWYG approach of Gummy

Gummy is a graphical user interface tool for creating multi-platform user interfaces. A designer is able to generate an initial design for a new platform by adapting and combining features of existing user interfaces created for the same application. The tool uses the WYSIWYG approach by providing visuals, as illustrated in Figure 2.6. Gummy takes care of the consistency between multiple user interfaces of the same application and targeting new platforms requires not much additional overhead. Recent authoring tools for distributed user interfaces focus on collaborative design and crowdsourced adaptation. Quill [49] is a web user interface developer tool that enables collaboration between different stakeholders (programmer, project manager, support manager, etc.) by adopting a model-based approach for the design of cross-platform user interfaces. The model-based approach allows for incremental development for multiple technologies. Enabling common understanding of the UI specification through models to facilitate targeting various platforms. CrowdAdapt [50] however is a context-aware web design tool which uses crowdsourcing to improve the adaptations of a web suite under various viewing conditions. Developers can create adaptive layouts for different use contexts by direct manipulation of the final user interface. The main idea behind CrowdAdapt is to allow end users to adapt the interface to their specific context if not supported by the current design of the web page. This idea of end user adaptation will be explored in this thesis.

2.2 Augmented Reality User Interfaces

Already in the early nineties, researchers investigated Augmented Reality in combination with Personal Information Management in order to improve the organisation and re-finding of information items in office settings. Early research combined physical and digital media leading to hybrid surfaces, like DigitalDesk [70] where overhead projection is used to provide digital functionality to paper documents. More recent studies focus on tabletop user interfaces that are designed for linking digital and physical content. By placing a physical document on the tabletop surface, DocuDesk [17] provides a way to recognise the document and show the relevant linked documents, its digital counterpart and some other options. PeriTop [54] and ObjecTop [36] tackle the problems of occlusion created by physical objects on interactive tabletops and also allow support for the organisation of the hybrid physical-digital workspace. Thinking about knowledge workers, they often switch between different tasks. Before they can start a new task, they will first put aside the current documents they are working on. PeriTop and ObjecTop enable hybrid piling by manually putting digital objects on a stack. The hybrid piling helps the user, when they revise a task that they started earlier on. Another approach is MagicDesk [6] that tries to achieve a continuous workspace by augmenting the desk with touch regions in order to manipulate documents on the desktop. A Digital Mouse Pad enhances the mouse operations and a Multi-Functional Touch Pad allows for extra interaction possibilities with the desktop. Hybrid surfaces are used to combine digital and physical media in different kind of applications.

In the context of the PIM activities, research has been focused on providing physical augmentations in order to help users in organising and re-finding their personal documents. By tracking physical documents, digital and physical user interfaces are been developed to indicate where documents are located and they provide additional metadata in order easy the re-finding process. Seifried et al. [58] provide an augmented file cabinet linking the real and digital world. The system connects the physical and digital documents by enhancing filing cabinets and folders, while providing a mechanism for the organisation and retrieval of both, physical and digital, versions of the documents. Rather than linking the documents itself, this system focuses on linking the organisational structure of the physical and digital documents. A similar approach was proposed by Jervis [31]. He introduced a framework to trace physical documents in file folders situated in file cabinets. By the integration with OneNote, users can digitally re-find a physical document where then the LEDs of the contained file folder will highlight.



Current Document
Title: Data Structures and Algorithm Analysis in C
Author: Mark Allen Weiss
File: books/data_structures.jpg
First Access: 17:04:05 04/09/2004
Last Access: 17:57:59 04/09/2004
Access Freq: 4

Figure 2.7: The pile browser to digitally browse through a physical pile

In order to provide such a re-finding support in piles on a desk, Kim et al. [37] developed the digital pile browser where users can digitally navigate through physical piles, as illustrated in Figure 2.7. Thereby they can use the digital representations of physical documents in their workflow. Limpid Desk [29] uses a projection-based technique to help users re-find the desired document in a physical pile on a desk. Searching through all of the documents in a pile is time consuming, that is why Limpid Desk proposes an approach where users are able to scroll through the pile without doing the physical act of scrolling. The physical and digital world is combined to change the appearance of the physical documents so that they appear to look transparent. By making the documents transparent, documents stacked lower in the pile can be located without physically searching through the whole pile.

Re-finding books on a shelf has also been studied extensively in Augmented Reality. In libraries, large amounts of books are stored using a classification system. In 1905, two Belgian bibliographers invented one of the first classification systems, called Universal Decimal Classification system (UDC) [44]. In this system, each category is represented by a decimal number and the order of the decimal numbers is fixed. For huge amount of books, this system allows for easier storage and retrieval. However when thinking about bookshelves in office settings, they mostly offer less space to store books. So other techniques need to be used to coop with the storing and re-finding of books in offices. The Smart Bookshelf [12] is such an approach, it allows users to query for the presence of a book through an interface and if found, the book gets highlighted. The tracking and re-finding of the books is done by a camera and a projector.

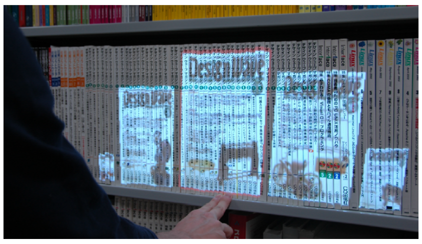


Figure 2.8: Projecting the cover of a stored book onto the bookspines allowing to search the desired book by scrolling

Interactive Bookshelf Surface [43] allows searching for a desired book on a shelf without having to take every book out to look at the cover. When the user touches the edge of the bookshelf, the cover image of the stored book is projected onto the bookspines, as illustrated Figure 2.8. By sliding his finger across the shelf the user can scroll through the books on the shelf looking for his desired book. The physical bookshelf gets augmented with digital content in the form of book cover images. Other augmented reality applications augmenting bookshelves use a mobile phone. Chen et al. [9], Hahn et al. [21] and Malhotra et al. [41] use the camera of the phone to overlay real-time information about the books on the shelves in order to re-find the desired book more efficiently. Rather then showing the digital data directly on the physical books, the data is shown to a mobile phone. Libagent [59] uses the same technique with the mobile phone, but also provides more specific context-sensitive information, which allows for better results when searching for the right book. Libagent provides context-sensitive information about missing books or books on loan.

Physical documents are found everywhere in offices. These documents themselves can be digitally augmented, like sticky-notes, land maps and paper itself. Quickies [47] are enriched sticky-notes that can be tracked and managed. A graphical user interface allows to search and browse all of the user's notes without knowing the exact location of the sticky-notes. Paper-based city or land maps are often hanging in offices. Some recent studies try to combine physical paper with digital media in order to tackle the problem of searching on maps. For example, Map Torchlight [57] which uses a mobile phone's camera together with a pico projector to highlight places on the map. As illustrated in Figure 2.9, MapLens [48] also uses a mobile phone, but overlays the camera's view with real-time information and highlights the search area. Paper itself can be augmented, MobARDoc [26] provides digital features for printed content through augmentation of a phone's camera view. People can use this system to search in printed content for keywords. Another hybrid approach to combine physical and digital media is the introduction of paper interfaces in workflows. By using the Anoto dot pattern, various applications are designed to integrate physical media in the digital space and the way around providing digital augmentations to physical media. An example is PaperPoint [61] where users are able to annotate and control PowerPoint slides by using their printouts in combination with the digital pen.

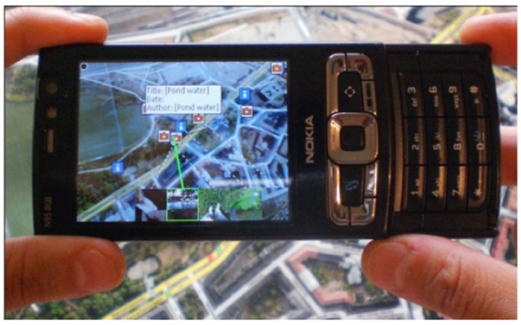


Figure 2.9: MapLens provides real-time information through the mobile phone's camera view

So far, different approaches combine physical and digital media, but in almost every office, there is also a laptop or desktop computer on which users can store digital documents. The classification problem that was described by Malone [42], also occurs when organising digital information. In the digital information space, people mimic the way they organise their physical space. For storing digital files on a computer, file hierarchies are used, because they represent the way a user stores his files inside of a physical file cabinet. Searching a digital file in a file hierarchy can be quite difficult. A user has to remember where he stored the information, namely the search path. If the file is not located in that place, users are often dependent on the desktop search engines to find the particular file. A problem with this strategy is that remembering the file's name can be even harder than remembering the location path. There is a need for systems to help us organise our digital space in order to re-find information items more easily, without remembering the exact name or location path. Personal Information Management research introduced various augmented reality applications that try to solve this problem.

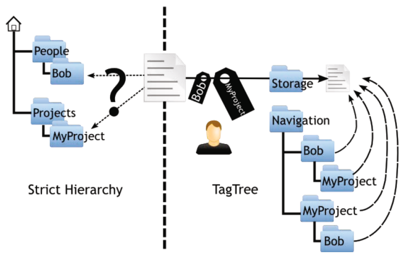


Figure 2.10: The concept of TagTree when storing the file "Bob's ideas about MyProject.txt"

One of the proposed applications to improve re-finding information without remembering the exact location is called TagTree [69]. Storing and retrieving files and folders is done by using tagging and automatically maintained navigational hierarchies. An example, we want to store a file, named `Bob's ideas about MyProject.txt`. The file is stored into a central folder, but the user adds tags to the file, like `Bob` and `MyProject`. Hence, the single target file can be found in any one of four possible paths: `Bob`, `MyProject`, `Bob/MyProject` or `MyProject/Bob`. This approach is similar to how the human brain works, because it allows users to find the right item by using associations. The concept of TagTree when storing the file can be viewed in Figure 2.10. iMecho [11], an associative memory-based desktop search system, also exploits these associations that the brain creates, but it adds contexts to enhance the traditional desktop search. Another example for speeding up the process of finding the right file when using a file system is *Icon Highlights* [18]. This approach uses algorithms to predict which items a user is going to access. Depending on the likelihood of being accessed inside the current folder, icons will be highlighted to draw the user's attention, as illustrated in Figure 2.11. Besides highlighting the icon, they also provide *Hover Menus*, which show shortcuts to commonly accessed items inside folders in order to reduce the number of steps in the location path. A third technique is *Search Directed Navigation*, which guides users through the file hierarchy based on a filename query.

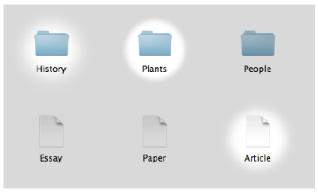
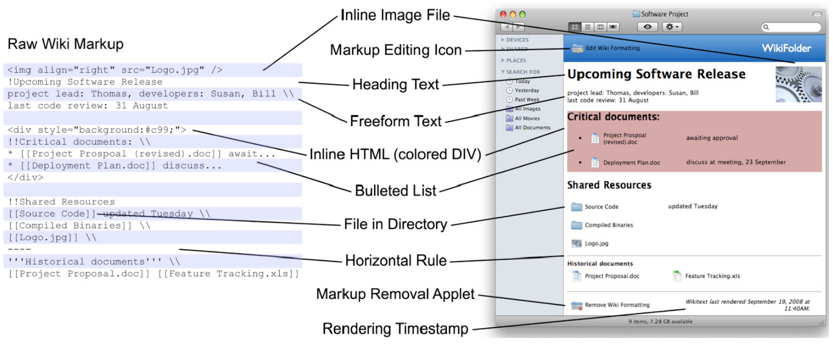


Figure 2.11: Highlighting the icon that are most likely to be accessed

In 2009, the view of a folder inside of a file hierarchy was augmented with additional meta-data. WikiFolders [68] is a system for annotating the digital folders by combining the strengths of hierarchical file systems and wikis. WikiFolders modifies the way folders are represented within the existing Mac OS X file browser. The purpose of the system is to enable users to organise and document their digital items as if the folder itself were a wiki page, as shown in Figure 2.12.


Raw Wiki Markup

```
<img align="right" src="Logo.jpg" />
!Upcoming Software Release
project lead: Thomas, developers: Susan, Bill \\
last code review: 31 August

<div style="background:#c99;">
!!Critical documents: \\
* [[Project Prospoal (revised).doc]] await...
* [[Deployment Plan.doc]] discuss...
</div>

!!Shared Resources
[[Source Code]] updated Tuesday \\
[[Compiled Binaries]] \\
[[Logo.jpg]] \\
----
'''Historical documents''' \\
[[Project Proposal.doc]] [[Feature Tracking.xls]]
```

Figure 2.12: The view WikiFolders presents in the standard file browser with the corresponding markup on the left hand side

In the Personal Project Planner or Planz, the file system is augmented with document-like overlays. Planz allows users to not only organise files, but also other forms of information including email messages, web references and informal notes. The file hierarchy is used as backbone for this application. Headings and subheadings correspond to folders and subfolders of the file system.

2.3 Context-Aware User Interfaces

Some user interfaces adapt their content or layout depending on the context around them. Well-known examples of user interfaces that are aware of the context are user interfaces for mobile devices such as smartphones and tablets. Most user interfaces allow to change their orientation (i.e. from landscape to portrait or vice versa), when the device is rotated or flipped. The user interface takes the input from the context into account, in this case the orientation. However, a lot of smartphone applications also use the geo-location of a user as context input, to suggest restaurants or bars nearby or information on buildings as an interactive tourist guide. Google maps offers a Traffic widget for Android smartphones that allows users to set up different locations. Whenever you are on the predefined locations, traffic information is updated in real-time and shown on the screen of your smartphone. Nevertheless, when you are at the location, you do not always want the notifications about the traffic coming up. Currently there is no way to turn off the notification or to state that they should only be shown under certain conditions.

In the research field of Context-Awareness, applications like CybreMinder [13] were proposed to allow context-aware support for reminders. The prototype tool provides a solution for the lack of using context in order to specify when a reminder should be presented to the user. Current reminder tools only allow to send reminders at a specified time. No other contextual situations are taken into account. However, contexts such as location could be very helpful as described in the Google Traffic widget. The reminders can be time-based, location-based and complex reminders. The time-based reminders are the normal reminders that are widely used. When the time matches the current time, the reminder is shown to the user. Location-based reminders are send when a user is in a certain location. For example, Alice wants to be reminded to take her umbrella to work. The umbrella is kept next to the door of her house. The reminder will be triggered when Alice is in a certain proximity of the door. Complex reminders allow for the use of richer context, such as reminding Bob to fill the gas tank of his car when the gas tank contains less than 5 gallons of fuel. This approach uses the Context Toolkit [55] developed by Dey et al. to develop the context-aware application. The toolkit allows to separate context sensing or acquisition from context use and interprets the context.

2.4 Where Is The Gap?

In the first part of this chapter, we have seen that the office environment is becoming very dynamic. User interfaces are distributed across different devices and spaces in order to easily interact with them. When distributing a user interface over multiple devices, different approaches of adaptation were used. Some approaches adapt the user interface based on other devices, others only adapt allocated components of the user interface to other devices. All of these approaches aim to support more natural ways of interaction. One thing, we also noted is that the distribution of the user interface is managed by the developer of the application, he is responsible of how and with what end users will interact. One of the authoring tools introduced the concept of end user control. Nonetheless, most user interfaces are not customisable by end users.

In the second part, we have seen that distributed user interfaces are not the only user interfaces that have found their way into the office environment and into our daily activities. Augmented reality user interfaces have been extensively studied in the field of personal information management and context-aware user interfaces were introduced to automatically adapt based on the context. However, these user interfaces are again predefined by the developer or UI designer. End users have no means to customise the augmented user interfaces themselves, depending on their preferences. In context-awareness, they studied user and system control. Studies showed that there needs to be a balance in the control. Giving all the control to the system in order to automatically adapt based on the current environments is not desired, because users are not aware of what and how things happen. Current research only focuses on automatically adapting, we will try to bridge the gap by providing a balance between end user and system control for user interface adaptation.

2.5 Conclusion

To summarise, we have seen that different types of user interfaces exist. Distributed user interfaces focus on the office environment and the interactions with multiple devices. The design of distributed user interfaces is organised by UI designers or developers of the applications. In Augmented Reality, the user interfaces were developed to help users in their daily activities, for example when they need to re-find a document on their computer. Different kinds of augmented reality user interfaces were developed, having specific properties and needs (i.e. the required devices, modalities used, etc.). Context-aware user interfaces take the context into account in order to automatically adapt the user interface accordingly. However, the set of user interfaces for a specific application is mostly fixed or limited. The user interfaces are also predefined by the developer of the applications.

3
System and User Control

As observed in the field of context-awareness and recommender systems, user acceptance and satisfaction increases, when end users have some control over the behaviour of context-aware systems. The user control is described as the level of intervention that is required by a user in order to operate the system [23]. Recent research in context-awareness focuses on the contextual information and automatically adapting the system depending on this information. The input from the user gets limited, because the control is transferred to the system, rather than the user. However van der Heijden [67] argues in his paper that the transfer of control to the system creates a personal discomfort when using the system. A correct balance between user and system control needs to be achieved without requiring extra mental effort. In our framework, we will try to bring this balance of control to the research field for adaptation of user interfaces. We will focus on augmented user interfaces and try to achieve a balance in control by supporting two approaches, namely automatic adaptation and end user control. After a brief introduction of automatic adaptation, we will discuss our two approaches in detail.

3.1 Introduction to Automatic Adaptation

Adaptation is the process where interactive systems adapt/change their behaviour for individual users depending on the information about the user itself and its environment. This information about the user and its environment, is called the context. Context is associated with the field of Context-awareness, which refers to the idea that computers can sense the environment and react accordingly. The term was introduced by Schilit [56] in the field of ubiquitous computing, where the main idea is that devices are appearing everywhere and anywhere in our daily life. Context-aware systems adapt according to the contextual elements in their environment. In smart-homes, for example, the lights are automatically turned on when a person walks past a detector during the night. The term *context* refers to much more than just the location of a user. Schilit described three important aspects of context, namely where you are, who you are with and what resources are nearby. We will give some examples of the three aspects below and put them into a category.

- Computing context: nearby resources such as printers, displays and workstations.
- User context: location, user profile, people nearby, and even social situation
- Physical context: lighting, noise levels and temperature

In 2000, Chen and Kotz [10] argued that time should be a fourth aspect of context. Because time is often used as context for many applications and can not easily be fitted into the three categories described above. In the 'time context' category, they consider time of day, week, month and year, but also season of the year. When talking about context in this thesis, we will include the aspects of both Schilit and Chen.

In smart homes, the input produced by sensors is used to adapt the environment. Physical context, such as the temperature are taken into account to trigger corresponding actions. For example, if the temperature is above 26ÂřC, the system triggers the action to close the blinds of the windows. Other systems, take user context, such as location into account in order to suggest events nearby, for example. Computing context can also be taken as input. When using a smartphone to browse the internet, the content of web pages is adapted depending on the fact that we are using a smartphone to view it. In W3Touch [51], the content is adapted to touch-based mobile phones.

3.1.1 Rule-based Context-aware Frameworks

For decades, context-aware frameworks are being developed to ease the development of applications in various domains, like recommendation systems and smart homes. Some context-aware frameworks offer support for context modelling and reasoning in order to adapt based on rules. Different approaches have been studied, like the SOCAM [20] framework. This framework is based on ontologies, where developers define an ontology for possible situations and their descriptions in the OWL language. The rule engine reasons over the ontology at runtime in order to detect abstracted situations from context data. Nevertheless, dynamic modelling of context is challenging, because changing the ontology at runtime can introduce conflicts and inconsistencies. The JCAF [3] framework written in Java takes an object-oriented approach based on a service-oriented architecture. Service components notify other services or client applications about detected contextual situations. Higher abstractions can be defined by composing context services. Via a distributed peer-to-peer setting the new high level abstract situations can be added or removed at runtime. JCAF, however, forwards the reasoning of the context rules to the application layer. When users want to dynamically change the rules' behaviour, client applications need to be redeployed. Context rules are also fragmented across multiple applications, leading to inconsistencies. The JCOOLS [53] framework overcomes these problems by integrating JCAF with a Drools rule engine[1]. Client applications can create new facts and events in the framework by following a XML schema and these can then be used in Drools rules.

Another context-aware framework is the Context Toolkit by Dey et al. [55], which is the most widely used and known in the Human-Computer Interaction (HCI) field. A component-based software design consisting of widgets is used. These widgets are responsible for the mediation between the environment and the client application. Client applications are able to listen to widgets and they can reason over the context rules in their knowledge base in order to determine the action to be taken. The context reasoning is pushed to the application, like in JCAF. In an additional middleware *Situations*, Dey and Newberger [14] provide control over context rules where users can change the parameters of a rule. In most context-aware frameworks reasoning is performed over a context model, the system can automatically be adapted corresponding with the rules of the context-aware frameworks.

[1]`http://www.drools.org`

3.2 UI4A2: UI Context Model for Automatic Adaptation

For the purpose of this thesis, we describe a context model for the Augmented Reality user interfaces that were discussed in the previous chapter. Since there is no specific context model defined for user interfaces. After an extensive study of the literature of user interfaces in the office of the future environment, we came up with the UI4A2 context model, described in detail below. This context model focuses on office settings. However this can be easily extended to include different types of settings and user interfaces. By studying different kinds of augmented user interfaces, different parameters were distinguished.

3.2.1 Physical vs. Digital Information Space

When taking a look around us, we can see information everywhere, like papers, books, pictures, post-its, folders and so on. But also digital information items (e.g. digital documents and emails) are included in the personal information space of a person. There is a clear distinction merging between the physical and digital information space. Translating this divide to the augmentations that were introduced in the previous chapter, we can see that a lot of the applications augment the physical information space, like books, paper documents, post-its, folders, etc. A rather small amount of the applications augments the digital information space, namely digital documents, emails, etc. In Table 3.1, an overview is given of the applications and to which information space they belong.

3.2.2 Type of Information Items

The type of information that is being augmented is the second question that we asked ourselves. Information items can take many forms, as we described before. Information items can be books, papers, post-its, pictures, etc. But information items can also include electronic documents, digital folders, emails, web pages and so on. Note that it is possible for an augmentation application to extend more then one type of information items.

3.2.3 Location of the Augmentation

Depending on the object or information item that is getting augmented, we can make a distinction based on the location of the augmentation with respect

	Physical	**Digital**
WikiFolders [68]		x
The Smart Bookshelf [12]	x	
Planz [34]		x
PeriTop [54]	x	
ObjecTop [36]	x	
Limpid Desk [30]	x	
Magic Desk [6]	x	
Docudesk [17]	x	
Interactive Bookshelf Surface [43]	x	
Viewfinder [9]	x	
Library service [21]	x	
Libagent [59]	x	
Library management system [41]	x	
Smart Filing System [58]	x	x
TagTree [69]		x
iMecho [11]		x
Icon Highlights [18]		x
Quickies [47]	x	
Map Torchlight [57]	x	
MapLens [48]	x	
MobARDoc [26]	x	

Table 3.1: Applications operating on the physical or digital information space

to the object. The location of the augmentation can be overlaid, around or virtual. When using DocuDesk [17], a physical document is placed on a desk, the document gets recognised by a camera and the augmentation consisting of links is shown *around* the document. In applications like Limpid Desk [29], the augmentation is projected on top of the stacked documents or books that are placed on a desk, so the augmentation is *overlaid*. A last possible location for the augmentations is virtual, by which we mean that the location of the augmentation is not directly on the document or around it, but rather on another screen or device, such as the screen of a smartphone. This is the case with Viewfinder [9], where the books on a shelf are augmented with real-time information that is shown on a smartphone.

3.2.4 Techniques

The domain of Augmented Reality provides different techniques for augmenting the real world environment. The techniques can be divided into six types:

- **Compositing:** where virtual information is overlaid in a scene, video or image, often to create the illusion that all the elements are part of the same scene or image
- **Head-up display:** the display of instrument readings projected on to the windscreen or visor, without requiring users to look away from their usual viewpoints[2]
- **Direct projection:** wearable augmented reality interfaces
- **Magic lens metaphor:** real-time augmentation of mobile device's camera view
- **Magic mirror metaphor:** the concept is similar to magic lens, except for the different orientation of the camera (e.g. pointed to the user)
- **Magic eyeglass metaphor:** See-through head mounted display where virtual images are mixed with the real world (e.g. Google glasses)

The magic lens and composition techniques are mostly used by the applications from the previous chapter. For example, Viewfinder uses the view of the mobile phone's camera to overlay the spines of books on a shelf with relevant information about the books in the view.

[2]`http://www.oxforddictionaries.com/definition/english/head-up-display?q=head-up+display`

3.2.5 Modalities

Users can interact with a system in different ways. A system provides an interface which relies on input and output channels that enable users to interact with it. Each single independent channel is called a modality [35]. Systems only based on one modality are called unimodal, whereas systems that are based on multiple modalities are called multimodal. The most common modalities that a computer uses to communicate and interact with users are vision, audition and tactition (vibrations and other movement). Computers can be equipped with different types of input devices and sensory input to allow interactions from users. Such modalities can be a keyboard, pointing device, touchscreen, computer vision, speech recognition, motion and orientation.

3.2.6 Devices and Technologies

A last categorisation that we will introduce is the different devices that are being used by the applications. Each application has its own setup to be able to coop with the tracking, organising and re-finding of information items. Note that applications in the physical information space use very different technologies and devices, then applications for the digital information space. In digital information space, most of the applications augment the file hierarchy structure on a desktop. Depending on which operating systems being chosen, either OS X finder or Windows Explorer were used to inspire the applications. The applications were either extending the file browser or creating a stand-alone application using the file structure as a guideline.
An augmentation in physical space can use one or more devices. We will sum up the most frequently used devices by the applications that were introduced in the previous chapter. Some of the devices include multiple features that are used, as illustrated below.

Frequently used devices:

- **projector**
- **camera:** smartphone, tablet, stand-alone camera
- **touchscreen:** smartphone, tablet, tabletop
- **mobile devices:** smartphone, tablet
- **desktop**

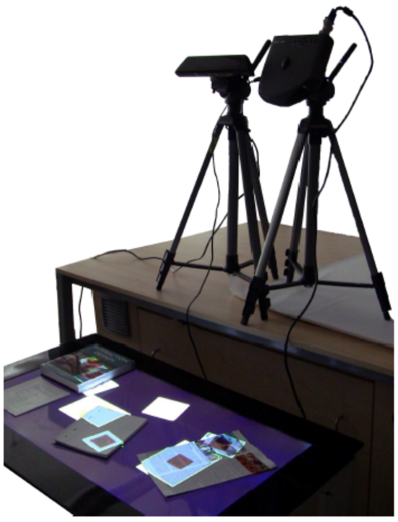


Figure 3.1: The setup of the PeriTop system: camera, projector and tabletop

The viewfinder application introduced by Chen [9] uses a mobile phone to browse through the bookshelves searching for a book. The real-time view of the camera is overlaid with relevant information about the books in the camera frame. A mobile phone is clearly a mobile device. Applications using multi-touch tables, like MagicDesk [6] and ObjecTop [36] are placed under the touchscreen category. Note that some of the applications can be placed in more than one of the categories. As illustrated in Figure 3.1, PeriTop [54] uses a pico projector and depth camera to augment the tabletop system. By doing so, PeriTop combines three categories, namely the projector, touchscreen and camera categories. Computer-based augmentations include the applications augmenting the file hierarchy structure on a desktop, like TagTree [69]. But also Quickies [47] which are augmented sticky-notes that can be viewed and searched in a desktop application with a graphical user interface.

3.2.7 Overview

In this section we will provide an overview of the different augmentation applications in correspondence with the categories that were defined in the sections above. The overview of the different categories and how the augmented user interfaces correspond to the categories, is given in Table 3.2.

	Physical space	Digital space	Information items	Augmentation location	Technique	Modalities	Devices
WikiFolders [68]		✓	folders	overlaid	Compositing	pointing device	computer (Windows)
The Smart BookShelf [12]	✓		books	overlaid	Compositing	Computer Vision	smartphone
Planz [34]		✓	digital files: emails, pdf, etc.	overlaid	Compositing	pointing device	computer
PeriTop [54]	✓		physical documents and books	overlaid/around	Compositing	touch display, camera	pico-projector, camera, touchscreen
ObjecTop [36]	✓		documents	around	Compositing	touch display	touchscreen
Limpid Desk [30]	✓		piles of files	overlaid	Compositing	Computer Vision	projector, camera
Magic Desk [6]	✓		documents	around	Compositing	pointing device, keyboard	computer, projector, camera
Docudesk [17]	✓		physical documents	around	Compositing	Computer Vision	camera, display
Interactive Bookshelf Surface [43]	✓		books	overlaid	Compositing	Computer Vision	projector, camera
Viewfinder [9]	✓		books	virtual	Magic Lens	Computer Vision	smartphone
Library service [21]	✓		books	virtual	Magic Lens	Computer Vision	smartphone
Libagent [59]	✓		books	virtual	Magic Lens	Computer Vision	smartphone
Library management system [41]	✓		books	overlaid	Compositing	Computer Vision	computer, smartphone
Smart Filing System [58]	✓	✓	folders and files	virtual	Compositing	RFID, digital pen	computer, digital pen, special folders
TagTree [69]		✓	folders	overlaid	Compositing	pointing device, keyboard	computer (Windows)
iMecho [11]		✓	folders and files	overlaid	Compositing	pointing device	computer (windows)
Icon Highlights [18]		✓	folders	overlaid	Compositing	keyboard	computer (Mac OSX)
Quickies [47]	✓		post-its	virtual	Compositing	RFID	computer, RFID, digital pen
Map Torchlight [57]	✓		land map	overlaid	Compositing	Computer Vision	pico-projector, smartphone
MapLens [48]	✓		land map	virtual	Magic Lens	Computer Vision	smartphone
MobARDoc [26]	✓		words and sentences	overlaid	Magic Lens	Computer Vision	smartphone

Table 3.2: Overview of the applications with the corresponding categories

3.3 End User Control

Recent research only focuses on the automatic adaptation based on the context. However, users might want to have some control over these adaptations. In our framework we will provide an alternative option, where we will allow for a transfer of control towards the user. Just think back to the example of notifications when new email message arrive. Some users would however like to configure/program that when they are in a meeting no notifications are shown, because they would interrupt the meeting. With the end user approach, we explore the possibility to bring this concept to augmented user interface and eventually user interfaces in general. By allowing users to define their own rules for augmented user interfaces, the user interface adaptation is user-specific. We will take a look at end user programming and specifically at end user modelling tools that allow for easy modelling without extra mental efforts.

3.3.1 End User Programming

Many toolkits and infrastructures were developed to support the development of ubiquitous computing, however only few have focused on allowing end users to build applications. In order to empower the end users who have little or no knowledge about programming, different types of applications were developed. We will discuss rule-based tools and augmented reality editors, because of the similarities with concepts in this thesis.

Rule-based tools

A first rule-based approach that was investigated is Jigsaw Editor [28]. It allows novice end users to reconfigure home devices in a domestic environment by using a jigsaw puzzle-based approach. By combining puzzle pieces on a tablet, users are able to "snap" sensors or devices together to meet their personal needs without writing any code. A second approach is CAMP [65], where they provide a magnetic poetry interface to enable end users to create and customise applications for their homes. The application uses a natural language approach of arranging fridge magnets-like words, because it offers flexibility to the users. By arranging the words, for example, a user places the words: *take picture every 5 seconds* on the board, the context is captured. iCAP [15] introduces a rules-based approach for rapid prototyping and testing of applications without writing programming code. It supports simple trigger-action programming where an action is triggered if a condition is satisfied. This if-then rules are easy to grasp by end users and do not require any programming background, making it very suitable for end user programming.

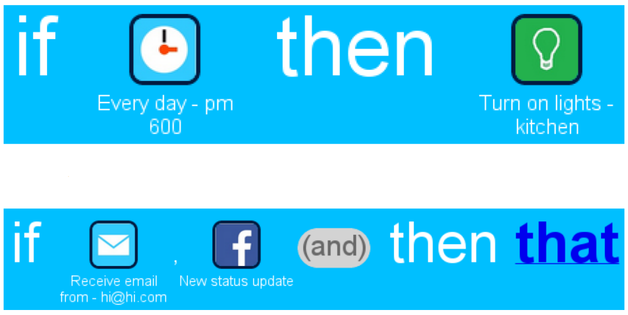


Figure 3.2: if-then rule-based approach of IFTTT recipe

End user programming often takes the form of rules, more specifically "if trigger, then action" rules. Such an approach is also called trigger-action programming. A study in 2014 by Ur et al. [66] provides evidence that users with little knowledge about programming can easily engage in trigger-action programming. During the study users had to control their products and apps by using IFTTT[3]. As illustrated in Figure 3.2, an IFTTT recipe allows an event (e.g. time event, receiving an email, etc.) to trigger an action (e.g. turn lights on, open facebook, etc.). Simple visual elements are used to easily grasp the recipes.

Lee et al. [38] created a mobile and tangible programming tool to allow users to create context-aware applications by defining rules in smart environments. Their application, called GALLAG Strip, enables visual programming by physical demonstration of envisioned interactions with the sensors and objects that will be encountered in the finished application. Working with the smartphone allows for a natural interaction and most people already use a smart phone in their daily life.

Augmented Reality Editors

Editing behaviour of objects is also studied in the domain of augmented reality. Until recently, augmented reality editors and browsers such as Layar[4] and Wikitude[5] only allow designers with the ability to generate or view extra information on top of magazines, newspapers or city landscapes. Anyone

[3]https://ifttt.com/wtf
[4]https://www.layar.com/
[5]https://www.wikitude.com/

with a standard smart phone is able to use these applications that typically enrich the physical world with information. However, researchers have started to explore AR in combination with Ubiquitous Computing. Smarter Objects [25] is a recent work that follows this direction. It explores new interactions with everyday objects by direct mapping of virtual interfaces. For example, when a user points a mobile device with a camera at a physical object, then the object gets recognised by the application and it provides a graphical user interface to program the behaviour of and interactions with the object.

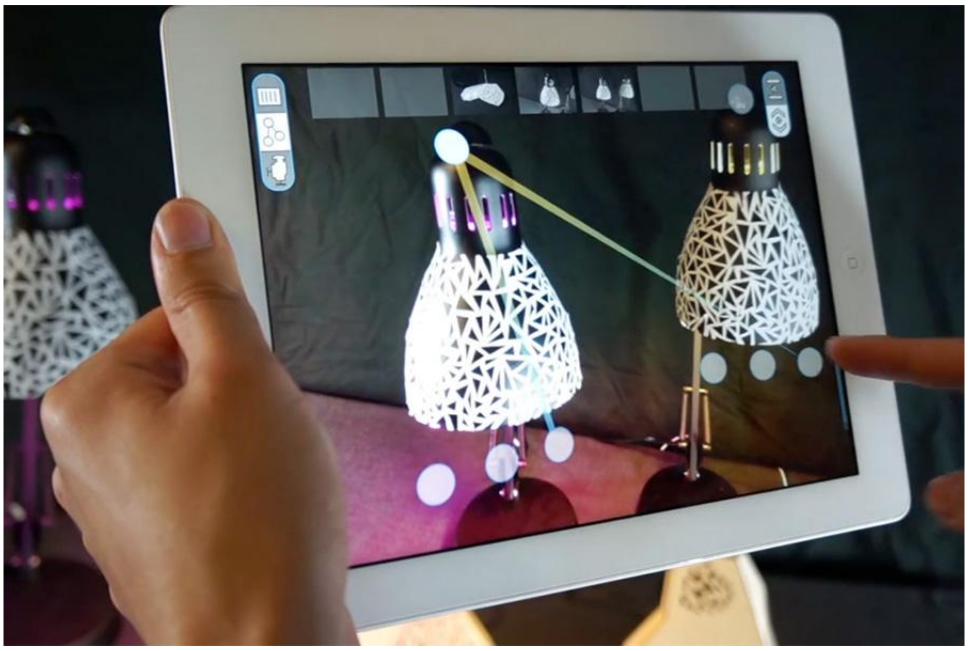


Figure 3.3: The Reality Editor that supports editing of behaviour and interfaces of "Smart Objects"

Further research was conducted by Heun et al. [24], where a framework was introduced that allows editing the behaviour and interfaces of the "Smarter Objects". The Reality Editor uses augmented reality techniques to map graphical elements on top of tangible interfaces found on physical objects, such as lamps, as illustrated in Figure 3.3. The reality editor allows for a fluid, natural way to interact with objects by using everyday objects in order to edit the behaviour and interfaces.

3.4 Requirements

In the sections above, we described the two approaches that our system will provide. On the one hand, we want the automatic adaptation corresponding with the user preferences and on the other hand, we want a natural, easily understandable way to allow end users to adapt the augmented user interface that should be shown, depending on the context. Each approach has its own requirements that need to be fulfilled and the main requirements are described below.

3.4.1 Automatic Adaptation

There are three main focuses for our category-based automatic adaptation, namely intelligibility, user preferences and easy adaptation configuration. We will discuss them in more detail.

Intelligibility Users need to be able to gain insights in how systems work. In the context-awareness research field, the term intelligibility was introduced by Belotti and Edwards [4] to state that users must be able to understand why a system invokes certain actions or why not. Our framework needs to provide some form of "intelligibility", the user needs to be able to understand why a certain augmented user interface is shown or get feedback why it could not be shown. Although, it is important that users are aware of why certain things happen, too much intelligibility is not recommended.

User preferences The automatic adaptation is based on the preferences of the user. These preferences are based on the categories that were introduced in this chapter. Users need to be able to state that they want or do not want certain properties of an augmented user interfaces under certain conditions. So based on the preferences of the user (e.g. no speech modality), an automatic adaptation needs to be applied, when the current context matches the conditions of the adaptation.

Adaptation configuration The user interface decides on which categories will be taken into consideration for the automatic adaptation. Depending on the categories of the user interface, it can be adapted if the conditions match with the current context.

3.4.2 User Customisable Adaptation

Our second approach will allow users to model their own rules in a simple way without further knowledge, but expressing enough detail. The details of these requirements are discussed below.

IF-THEN rules Allowing end users to program or model applications, requires for a natural, easy to understand approach. Trigger-action programming lets users customise their devices by using simple if-then rules. These rules are easily graspable by end users who have little to no knowledge about programming. Our framework will use simple if-then rules to allow end users to configure their augmented user interfaces.

Granularity of context components Our framework will allow users to model which contextual components need to be invoked for a certain augmented user interface to be shown. Different granularities or types of context components need to be provided in order to allow users to express themselves. For example, a user want to state that certain people need to be available in the office for a certain action to be performed. Our framework needs to be able to let the user express this by allowing for persons and locations as context components. Other components can be provided, but in the section about the Context Modelling Toolkit, we will discuss these components in detail.

User guidance Applications are becoming more advanced and complex. The importance of usability in the development has risen in order to minimise the memory load on the user. Our framework will provide guidance to the user in order to lower the cognitive memory load. By hiding or disabling parts that are not applicable at the moment. By doing so, we will guide the user "by the hand". We will not only by limit his possible options, but we will also provide feedback.

4 DUI2 Framework

For end users to be able to configure their user interfaces and to allow adaptation of user interfaces, the DUI2 framework is introduced in this chapter. The framework tries to provides a balance between user and system control. The two approaches, namely automatic adaptation and user customisable adaptation, discussed in the previous chapter are supported. The architecture and the different modules of the DUI2 framework are presented in detail below.

4.1 Architecture

The DUI2 framework is a framework that is implemented in Java and it consists of four modules, respectively the Core, Util Component, the DUI2MUI and the CMT Client module, as shown in Figure 4.1. The parts in blue are the components that were implemented for the purpose of this thesis. The other components are existing components that the DUI2 framework utilises. The Core module of the DUI2 framework takes care of the communication with the other modules and combines the functionality of the different modules to implement the DUI2 functionalities. The Util Component provides some helper functions that can be used in the Core module. The DUI2MUI module is responsible for the communication with the MUI component. The module offers functions to the core module, which internally calls the REST

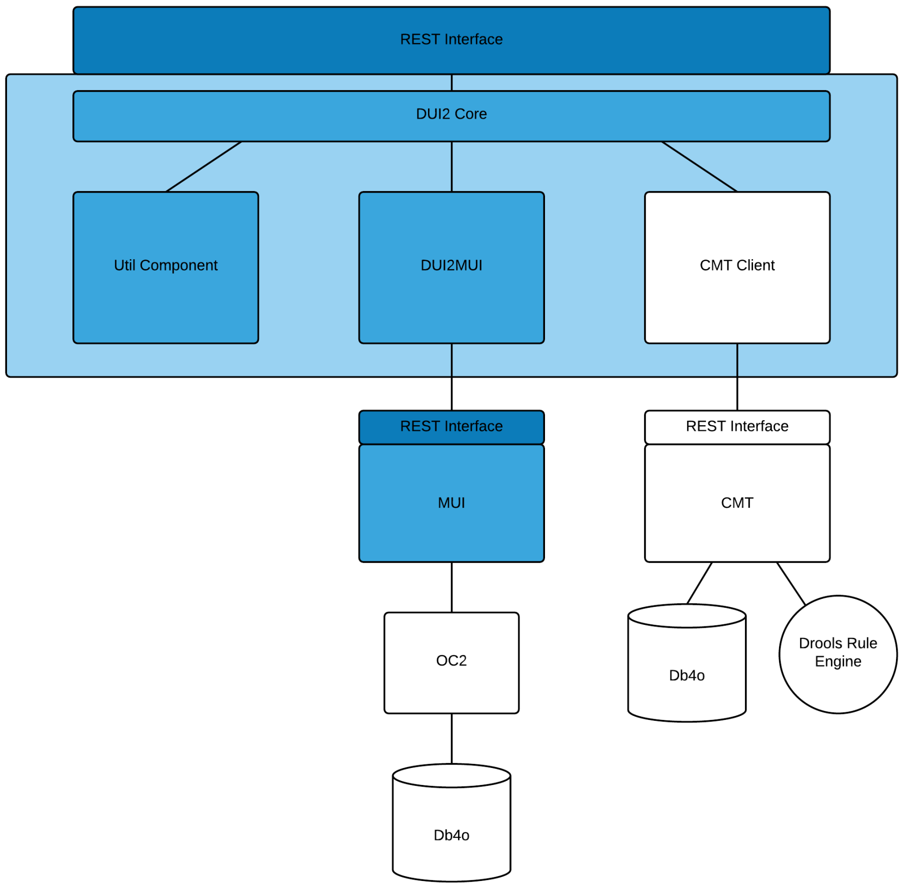


Figure 4.1: The architecture of the DUI2 framework

end points of the MUI component. The GET, POST and PUT request to the MUI framework are implemented with the Unirest library[1], written in Java. The MUI component was provided in function of this framework and it takes care of the management of the user interfaces by mapping its components to components of the OC2 PIM framework [64]. The PIM framework was used, because it provides the functionality to link digital as well as physical objects. The CMT Client module allows us to communicate with the CMT server in order to construct rules, get the current context, get event or fact types and so on. The communication with the CMT server is done via a RESTful interface and the data exchange format that is used is JSON. The CMT server consists of a db4o object-oriented database and a Drools 6 rule engine and supports context reasoning.

[1]`http://unirest.io/`

4.2 MUI Framework

The MUI component is the component that is responsible for the management of the user interfaces. By mapping its components internally to the OC2 mmodel, we are able to use the OC2 model as a metamodel in order to store and manage user interfaces. Our MUI component reuses existing OC2 components by keeping mappings that were created for this thesis. We will first discuss the model of MUI that we use for the management of user interfaces, then we will introduce the OC2 model and its underlying structure. And afterwards, we will talk about the mapping itself, and how our MUI component keeps track of its components.

4.2.1 MUI Model

Our MUI component will take care of the management of user interfaces. In order to do so, a model was created using the OM datamodel [52]. The OM data model distinguishes between collections of objects and associations. A collection is represented as a white rectangle, whereas the type of objects included in the collection is represented by the shaded rectangle around it. A collection may be subtyped by other collections with a partition constraint or a disjoint constraint. The associations are shown as oval shapes and they connect elements of collections. For each collection that participates in an association cardinality constraints can be set. The model consists of `Objects`, `User Interfaces`, `AugmentUI`, `UserInterfaceLinks`, `Conditions` and `UIProperties`. These are linked via associations such as `HasUITarget`, `HasUIProperties` and `HasConditions`.

For our MUI component, we want to keep track of `User Interfaces`, these user interfaces can be added to the MUI framework and are stored in a UI collection. A collection of `Augment UI` keeps track of the User Interfaces that are also augmented user interfaces. In our use case, all of the User Interfaces will be added in the Augment UI collection. An instance of a User Interface can have `UIProperties`, these UIProperties are the configuration details of a User Interface, which can be an Augment UI. The UIProperties can contain information, such as the name of the User Interface, the type of augmentation, location of the augmentation, information items that are being augmented by the UI, the devices that are required in order to show/execute the User Interface and the modalities that are being used. The UIProperties correspond to the UI4A2 context model that was described in Chapter 3.2.

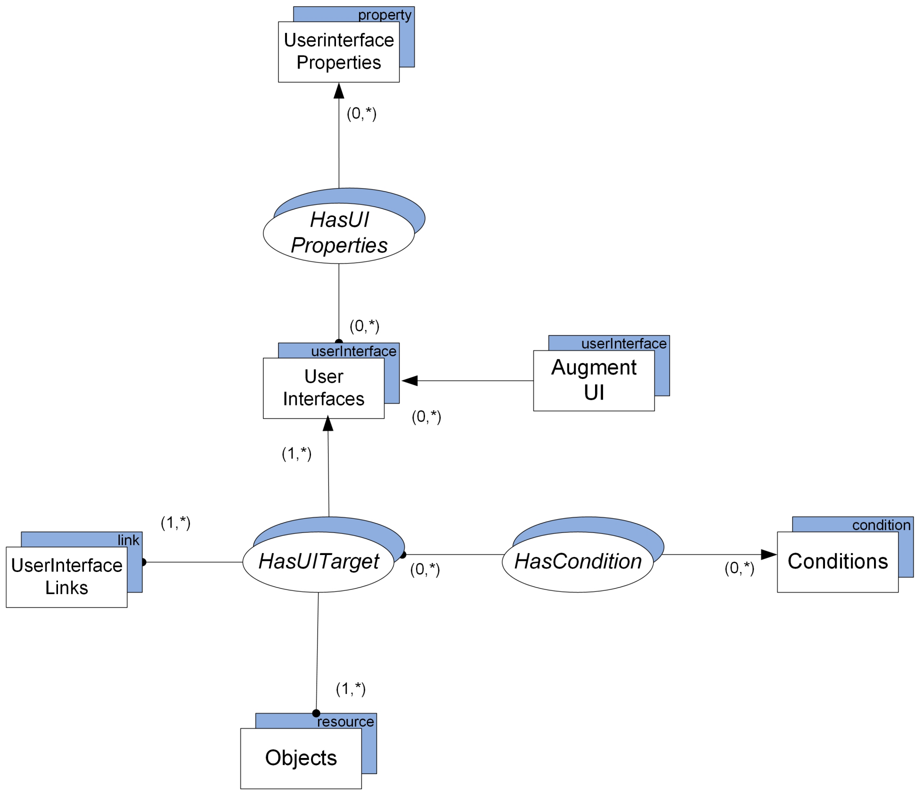


Figure 4.2: The MUI model for user interface management

Multiple UIProperties can be added to an instance of User Interface. Now we can store User Interfaces, Augment UIs and UIProperties that belong to a certain User Interface. In order for our system to be useful, we need to be able to connect an Object to a User Interface. For example, the paper *"DocuDesk: An interactive surface for creating and rehydrating many-to-many linkages among paper and digital documents"* is always augmented with the *"AugDocu"* augmented user interface. `UserInterfaceLinks` are used to link Objects to user interfaces, the target of the Object is an instance of User Interface. That is why we need the HasUITarget association, this association can only be used, if the Condition associated is set on true. Because an Object can only have one target User Interface at a time. Various Conditions can be set on when a User Interface is shown on an Object, then the HasUITarget is used if all the Conditions are set to true. The Conditions can be, for example, the required devices that are needed in order to show the User Interface on an Object. If a device is not available, this condition is set to false and the association HasUITarget will not be active. Now we can link Objects with a User Interface, if the Conditions of the User Interface are set to true. Then the target User Interface can be shown on the corresponding Object. Remember the *"AugDocu"* augmented user interface used for the paper *"DocuDesk"*, in order for this association to be used, the Conditions need to be true. We have just one Condition stating that a camera needs to be available. If the current context matches this condition, so a camera is available, then the "AugDocu" augmented user interface is shown on the Object, which is a paper in this case. The model of our MUI module can be seen in Figure 4.2.

4.2.2 Object-Concept-Context (OC2) Model

Our MUI component uses the OC2 model defined by Trullemans and Signer [63] as a metamodel. The OC2 model is an extension of the Resource-Selector-Link (RSL) metamodel. The RSL meta model was designed by Signer and Norrie [60] for dealing with data, structure and navigation in hypermedia systems. For example, in the context of research on interactive paper, PaperPoint [61] was developed. PaperPoint allows people that are giving a PowerPoint presentation to navigate, annotate and draw on the digital slides. By using a digital pen on a printed version of the PowerPoint slides, the according actions are shown in real-time on the digital slides.

The OC2 model describes a conceptual framework that consists of three levels which correspond with the human memory. The three levels include the *object, concept* and *contextual* levels, where each level has its own specific elements. At the object level are *objects*, which represent any information item that can be observed in digital or physical space. The OC2 framework makes a distinction between whether an information item is a physical object (i.e. a book) or digital object (i.e. an email). This extra information allows the user to re-find the needed information item faster. At the concept level, resources can be *concepts*. A concept is a general idea formed inside the brain, in an attempt to abstract the real world. Thus, concepts are words or sentences representing the user's mental model of an observation of objects. In the digital space, the labels that are given to folders in the digital file system can be seen as concepts. These labels represents a certain abstraction of the elements that are contained in the folder. The labels of the folders can be different for different users, depending on their mental model of the label. At last, the contextual level consists of *contexts* elements, which describe a composition of contextual factors. For example, a context 'Meeting' consists of contextual factors, such as the date, place, attendees, agenda topics and so on.

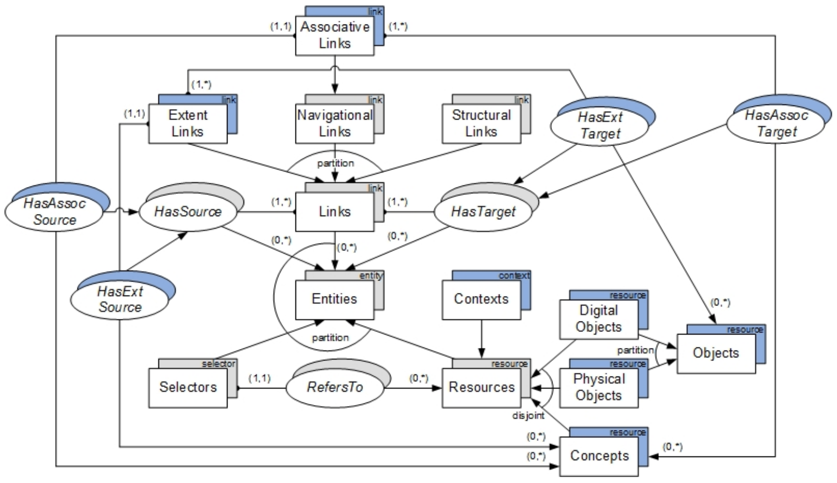


Figure 4.3: The OC2 metamodel extension of the RSL metamodel

The RSL model consists of different types of links, navigational, structural links, etc. In the OC2 model, on the concept layer, two new types of links are introduced, namely *associative links* and *extent links.* Concepts can have associative links to other concepts, but can also have extent links to objects of the underlying object layer. The human brain creates semantic associations, these associations are represented in the OC2 framework by bidirectional associative links. An extent link represents a categorical relationship between a concept and objects, these objects might be participating in one or more extent links.

The conceptual OC2 framework has been translated to the OC2 metamodel, which can be seen in Figure 4.3. The core link component of the RSL model was extended by the OC2 metamodel. The RSL components of the model are indicated with a grey color, whereas the extensions of the OC2 metamodel are highlighted in blue. By using the RSL metamodel as underlying structure, information items can be linked together, however context was not included. The OC2 metamodel added a context layer to the model. In Figure 4.4, context was added, giving the possibility to link two information items in a given context. For example, John writes a paper about Personal Information Management. He has a paper entitled *"DocuDesk: An Interactive Surface for Creating and Rehydrating Many-to-Many Linkages among Paper and Digital Documents"* with a weight of **0,9** (the weight is a number between zero and one) in the context of writing the paper. With this weight, the system lets the user know that this particular paper is relevant for writing a paper on Personal Information Management.

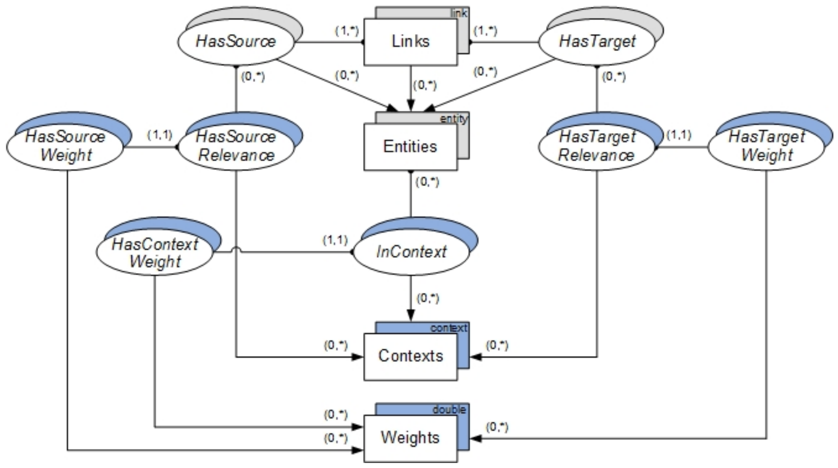


Figure 4.4: The context extension of the OC2 framework

If the user has another paper that describes Augmented Reality, the weight can be set to **0.3**, which indicates that this paper is less relevant in the context of Personal Information Management.

4.2.3 Mappings

All of the components that are shown in our MUI model need to be mapped to OC2 components in order to use it as a metamodel. Below we will describe the mapping we introduce to do the mapping between the MUI and OC2 components. Internally, the MUI components are implemented by making calls to the corresponding OC2 components.In the Table 4.1, the different MUI components and corresponding OC2 component are represented.

MUI Component	**OC2 Component**
Object	Object
User Interfaces	Resources
Augment UI	Resources
Conditions	Resources
UserInterfaceLinks	UtilityLinks
HasUITarget	UtilityLinks
HasCondition	UtilityLinks
HasUIProperties	UtilityLinks
UserInterface Properties	Properties

Table 4.1: Mapping of the MUI components onto the OC2 components

Object The Object component of MUI is exactly mapped onto the Object component of OC2. This means that Objects in MUI and OC2 are the same objects. These objects include physical objects, digital objects or concepts, as seen in the MUI model. An Object is in the OC2 framework equivalent with a Resource. By doing the following call, an Object is retrieved from the OC2 framework:

```
Object object = IServerOC2.get().getObject(Long.parseLong(anObj.getId()));
```

User Interfaces A user interface in our MUI component is created by calling the getNewUI function. This function calls the OC2 framework where a Resource is created. In our MUI component, we keep track of a collection of Resources with a special "UI" tag in order to make it possible to retrieve the collection of all user interfaces. In Listing 4.1, the code is shown to create a new User Interface in our MUI framework.

Listing 4.1: The function that creates a new user interface

```
public Resource getNewUI(String name){
    Resource ui =null;
    try {
        ui = IServerOC2.get().createResource(name, admin);

        Collection uicol = IServerOC2.get().getCollection(Schema.UI);
        uicol.add(ui);
        IServerOC2.get().updateCollection(uicol);
    } catch (CardinalityConstraintException ex) {
        Logger.getLogger(ReViTaCore.class.getName()).log(Level.SEVERE, null,
            ex);
    }
    return ui;
}
```

Augment UI The Augment UI collection allows us to stored all the user interfaces that belong to the category of augmented user interfaces. In essence, an Augment UI is a specific type of User Interface. This means that an Augment UI also corresponds to a Resource in OC2. For creating a new Augment UI, we first have to create a new user interface that is added to the collection with the "UI" tag. When stating that the user interface is also an augmented user interface, we will add the Resource to a new collection with the "AUGMENTEDUI" tag. This will allow us afterwards to retrieve all the augmented user interfaces, in the same way as the user interfaces except for a different tag.

Conditions A Condition or collection of conditions can be added to an augmented user interface and Object. A Condition is defined over an Object that has a certain User Interface. Conditions are internally mapped in the same way as the User Interfaces. In OC2, they are stored as Resources. A collection of Resources is kept in the MUI framework with the tag "CONDITIONS", where all the conditions known to the system are stored. With the 'intersects' principle, they can be retrieved.

UserInterfaceLinks UserInterFaceLinks are mapped to UtilityLinks in OC2. When we want to retrieve the user interface that is linked to an Object, the MUI component makes a call to OC2 and first gets the Object ID back from OC2. Afterwards the links for which the Object ID is the source are retrieved. By intersecting this collection of links with the links stored in our "UILINKS" collection, we have the User Interface by getting the targets of the Links. In Listing 4.2, the implementation code is given to retrieve the user interfaces.

Listing 4.2: The function that will retrieve all the user interfaces that are stored in the OC2 framework

```java
public HashSet<Resource> getUis(Object theObject){

    HashSet<Resource> result = new HashSet<Resource>();
    Object object = IServerOC2.get().getObject(Long.parseLong(theObject.
        getId()));
    if(object !=null){
        HashSet<Link> linksObject = object.getLinksISource();
        HashSet intersects = IServerOC2.get().getCollection(Schema.UILINKS).
            intersect(linksObject);
        for(java.lang.Object li : intersects){
            if(li instanceof UtilityLink){
                UtilityLink link = (UtilityLink) li;
                List<Entity> targets = link.getTargets();
                for(Entity ent : targets){
                    if(ent instanceof Resource){
                        Resource res = (Resource) ent;
                        result.add(res);
                    }
                }
            }
        }
    }
    return result;
}
```

HasUITarget The HasUITarget association links an Object, User Interface and Condition. Essentially, these are three Resources that can be retrieved from the OC2 database. In order to create a link between these three, the UserInterfaceLinks of the Object are retrieved and if the target of these links matches the User Interface, a UtilityLink is created and also added to the "CONDITIONLINKS" collection. A special property with a state, active or nonActive, is created and added to the UtilityLink.

HasCondition In our MUI component, we already stated that we keep track of a "CONDITIONS" collection, however we also keep track of the links between a Condition, Object and User Interface. We keep these links in a collection called "CONDITIONLINKS". In OC2, these links are just UtilityLinks.

HasUIProperties When an Augment UI is added to an Object, OC2 will get both Resources from the database. A UtilityLink between these two Resources is created in OC2 and also added to a collection with the name "AUGMENTEDUILINKS" in our MUI component.

UserInterface Properties Our UserInterface Properties are mapped to the Properties collection of the OC2 model. Whenever a Property is added to a User Interface, the Property is created with the key "configProp", to state that these are the configuration properties of the User Interface. Afterwards the Property is added to the Resource collection of OC2 and is also added to the "UIPROPERTIES" collection of our MUI framework. The Listing 4.3 shows how the User Interface Properties corresponding to a certain User Interface, which is essentially a Resource, are retrieved via the intersection of all the properties of OC2 and the UIPROPERTIES collection from the MUI framework.

Listing 4.3: The function for getting all the user interface properties corresponding to a user interface

```
public HashSet<Parameter> getUIProperties(Resource ui){
        Resource uiDb = IServerOC2.get().getResource(Long.parseLong(ui.getId
            ()));
        HashSet<Parameter> result = new HashSet<Parameter>();
        if(uiDb!=null){
            List<Parameter> params = uiDb.getProperties();
            Collection uiProps = IServerOC2.get().getCollection(Schema.
                UIPROPERTIES);
            if(uiProps!=null){
                List intersects = uiProps.intersect(params);
                for(java.lang.Object obj : intersects){
                    if(obj instanceof Parameter){
                        result.add((Parameter) obj);
                    }
    }
   }
  }
    return result;
}
```

4.3 Context Modelling Toolkit

The Context Modelling Toolkit (CMT) provides a multi-layered context modelling approach that distinguishes between different types of users, namely end users, expert users and programmers. Depending on the expertise of the user, the CMT application will allow the user to be in control of different parts of the application. The CMT application is a context-aware system that provides a user-specific balance between control and automation, it is implemented as a rule-based client-server architecture. For the purpose of this thesis, we will only use the server side, which is responsible for the context evaluation and the CMT client which provides functionality to create client applications for context modelling.

4.3.1 CMT Architecture

The architecture of CMT is a rule-based client-server architecture, where the client side takes care of the context modelling and the actual reasoning over the context is done at the server side. The framework is implemented in Java and uses a db4o database and for the reasoning, Drools 6 is used as rule engine. Both Java Remote-Method Invocation (RMI) and REST communication protocols are supported, but they require different data-exchange formats. The CMT-server includes the REST and RMI components that translate their input to the corresponding CMT data model entities and generate the required output format when data is sent to the client. We will use the REST communication protocol to interact with the CMT-server, the data-exchange is done via JSON. CMT provides a CMT Client to facilitate the development of client applications. The CMT Client supports common client functionalities and has four major components that will be explained below. The server side does not provide any context modelling functionality, but only serves as a platform for context reasoning.

4.3.2 Idea of Templates

In order to allow users to construct expressive rules for modelling the context, CMT foresees the possibility to reuse situations, when defining new situations and context rules. New situations are created by constructing rules of the form: *"if situations then new situation"*. However, regarding the rule management, this approach introduces a conflict. The actions of the new situation as well as these of the included situations will be triggered, due to the combination of situation and context rules. Therefore, a mechanism needs to be foreseen that manages the invocation of undesired actions from the included situations.

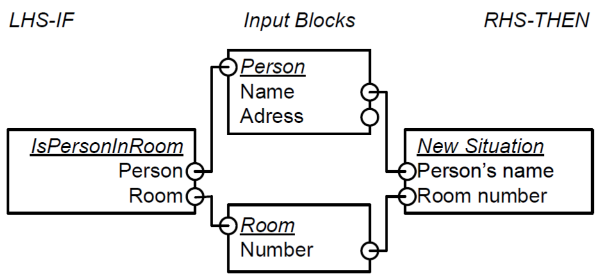


Figure 4.5: personIsInLocation template example

By introducing the concept of *templates*, CMT allows for easy declaration of situation and context rules. A template can be seen as a skeleton for defining rules. A rule consists of a left-hand side (LHS) and a right-hand side (RHS), respectively the IF-part and the THEN-part. The LHS can include multiple situations or logical functions which return a boolean. Situations and functions can take parameters of a certain primitive or class type. A template defines a skeleton including the signature of the required situations or functions, but leaves the actual parameters open to fill in later in time. Via *input blocks*, a template defines what class types are necessary to pass to the template's LHS. These input blocks are then linked to the corresponding parameters. The RHS can include actions or the new situation's name and its defined parameters. These parameters also originate from parameters of input blocks. For example, the `personInLocation` template, shown in Figure 4.5, has a LHS function which takes as parameters a Person and Room. These objects are entered in the input blocks. The RHS defines a new situation and has as parameters the person's name and the room number. By adding the object instances `Person:Bob` and `Room:10G731` in the input blocks, a new situation can be created. For the purpose of this thesis, we will use a simplified version of the template. A new rule can be defined by filling the LHS with initialised input blocks and the RHS will contain the corresponding action that needs to be executed. In order for the action to be executed, all of the input blocks need to return true.

4.3.3 Client Side

The client side of the Context Modelling Toolkit consists of a CMT Client module that provides functionality towards the users in order to model context. The CMT Client module contain four components. The `Listeners` allow a client application to listen for notifications from the server, for example, when a new rule is inserted. The `Transformer` provides functionality to convert plain Java object to CMT data model entities. The `Rule Compiler` compiles a DRL rule from filled in template instances. The `Com Unit` is responsible for the communication with the CMT server and can be used with both Java RMI and a RESTfull interface. The functionality that the CMT Client offers is registering class types of facts, events and actions, adding facts, situations, functions and templates, compiling a rule and listening for notifications via a publish-subscribe pattern.

4.3.4 Server Side

The server side of the CMT framework is responsible for registering different class types and making them available towards the clients. Once the class types are registered, instances of them can be added to the server, where they are sent to the db4o database and the Drools rule engine. Functions and templates can also be added to the server. Besides adding context elements, the server is responsible for adding rules to the Drools knowledge base via Drools Unit. However, programmers might add rules via Drools itself or via any other provided Drools user interface, such as GUVNOR. This allows programmer to add rules without the use of templates. The CMT server also foresees various listeners for client applications. Client can, for example, listen to notifications about added rules. The Observer component provides the necessary mechanism to notify the clients. For the communication via the REST interface, a publish-subscribe mechanism is implemented where clients are able to subscribe to various event types via a websocket connection.

4.4 DUI2

In our DUI2 component, we provide a RESTful interface to allow client applications to use the implemented functionality. The component consist of four subparts, namely the Core, a Util Comp, DUI2MUI and a CMT Client module, as explained before. The DUI2 framework supports two approaches of adaptation, namely automatic adaptation and user adaptation.

4.4.1 Functionality

Our DUI2 component provides functionality towards client applications via a RESTful interface. The interface includes different end points, which are listed in the Table 4.2 below.

End points	**Explanation**
.../DUI2/aug/activities	the input blocks for creating rules are given
.../DUI2/aug/activities/{name}	get input block with a certain name
.../DUI2/aug/sendActivities	the IF-part of a rule can be send to DUI2, in order to filter the actions
.../DUI2/aug/rule	add a rule (i.e. IF activities THEN do action)
.../DUI2/aug/config	add an Augmented User Interface to the application

Table 4.2: List of end points supported by our DUI2 framework

4.4.2 Automatic Adaptation

The first approach that the DUI2 framework supports is the automatic adaptation of a user interface depending on the current context. The defined UI4A2 context model that was introduced in the previous chapter is used to automatically adapt the user interface. The different categories are listed in Table 4.3. Depending on the current context, for example, the room, devices and people present, a certain augmented user interface is automatically shown when the properties of the user interface match the properties of the current environment. At the moment, the variables of user interface properties for an augmented user interface are stored as `Strings`, which can be extended in future work. So strings are compared in order to match a user interface in the automatic adaptations rules.

Categories
Physical or digital space
Information items
Augmentation location
Technique
Modalities
Devices

Table 4.3: Different categories from the UI4A2 context model

4.4.3 User Adaptation

Our DUI2 framework also allows for user controlled adaptation by creating templates. The end point `.../DUI2/aug/rule` of DUI2 allows applications to send filled in template instances to our framework via the RESTful interface, these are then sent to the CMT Client to be compiled to a DRL rule. The Rule Compiler of the CMT Client is responsible for the transformation of a filled in template instance to a compiled DRL rule, the function compileDrlRuleActivity which takes a template as input parameter is responsible for the compilation. After compiling the DRL rule, it is added to the CMT server, where it is added to the knowledge base of the Drools rule engine. Rules are created by using the idea of templates. A template consists of a LHS (i.e. the IF-part) and a RHS (i.e. the THEN-part). The LHS can contain one or more input blocks, which can be chosen from the input blocks that are available through the `.../DUI2/aug/activities` endpoint. This endpoint gives us back a HashSet with the avaible FactType elements to be chosen from. The RHS can only contain one action. In our case the action will always be *show {UI}*. Note that the user interface to be shown needs to be selected. The template for the augmented user interface will look the following: "if *situations* then show *UI*".

In our framework, we allow the LHS of the template to influence the RHS. Just showing all available user interfaces is very tedious. In our framework, we will guide the user towards the most compatible/preferred user interface. In Listing 4.4, the implementation code of the recursive function is shown that will filter the user interfaces depending on the input blocks inserted in the LHS of the template.

Listing 4.4: Recursive filter function for the compatible user interfaces

```
private HashSet<String> findUIMatches(HashSet<String> props, HashSet<String>
     uis){
    String prop = props.iterator().next();
    props.remove(prop);
    HashSet<String> uisMatches = new HashSet();
    if(!uis.isEmpty()){
        for(String ui : uis){
            boolean res = Communication.getAUIHasProperty(ui, "configProp",
                prop);
            if(res){
                uisMatches.add(ui);
            }
        }
    }else{
        uisMatches.addAll(Communication.getAUIsWithPropertyWithNameAndValue(
            "configProp", prop));
    }
    if(!props.isEmpty()){
        findUIMatches(props, uisMatches);
    }else{
        return uiMatches;
    }
    return null;
}
```

The findUIMatches takes as input the properties from the selected LHS input blocks and user interfaces, which will be initially `null`. The first time the function gets called, the augmented user interfaces that match the first property to be checked will be stored in the variable uiMatches. If the properties to be checked only contain one property, the list from uiMatches is returned. If multiple properties need to be checked the function is recursively called with the rest of the properties to be checked and the uiMatches from the first property. In the set of uiMatches the augmented user interfaces with both properties will be selected. The properties of an augmented user interface can be found by searching for the "configProp" tag. When an augmented user interface is registered, the configuration properties are inserted as strings into the "configProp" tag. This happens until the properties are empty. The findUIMatches function allows us to filter the user interface depending on the input blocks of the LHS. In this way, we can provide the users with guidance towards a compatible user interface, as our requirement "user guidance" from the previous chapter implied.

5 Configuration and End User Tool

The DUI2 framework provides the functionality to support both automatic and end user adaptation. In order for developers and end users to interact with our framework, we developed two applications, a configuration and modelling tool. The configuration tool takes care of the registration of the augmented user interfaces which is done by the developers and the modelling tool provides end users with an easy way to configure their augmented user interfaces.

5.1 Use Case

In our use case for the DUI2 framework, we focus on some scenarios that can happen in real life settings. In our first scenario, Sandra is working in her office on her CHI paper and she has augmented her paper by putting some post-its besides it to make some notes. The post-its are considered a physical augmentation of the reality. Later that day, she attends the CISA meeting in the living together with her colleagues. During the meeting, the layout and content of her CHI paper is discussed. Sandra has the paper with her in the CISA meeting, however she forgot to bring the notes on her post-its. The meeting goes on and they discuss the layout and contents of the paper. When Sandra remembers that she wrote something on her post-its that she needs to discuss about. An augmented user interface showing the digital version of these post-its would be useful to recall what she wrote.

In our second scenario, the automatic adaptation is illustrated. Researcher Tom is reading papers in his office. When reading papers, he likes to use an augmented user interface that shows him the relevant and linked papers. However, during his reading of the paper, he takes a book. The book gets detected and a corresponding augmented user interface for books is being shown, taking into account the properties of the current location and the book.

5.2 Configuration Tool

For our use case, we need four augmented user interfaces, which are listed below with their properties:

- **AugPost-its**
  - *Type of augmentation:* physical
  - *Location of augmentation:* around
  - *Information items:* paper
  - *Devices:* digital pen
  - *Modalities:* visual
- **Digi-Notes**
  - *Type of augmentation:* digital
  - *Location of augmentation:* around
  - *Information items:* paper
  - *Devices:* camera, projector
  - *Modalities:* visual
- **AugPaper**
  - *Type of augmentation:* digital
  - *Location of augmentation:* overlaid
  - *Information items:* paper
  - *Devices:* camera, projector
  - *Modalities:* touch, visual

- **AugBooks**
  - *Type of augmentation:* digital
  - *Location of augmentation:* around
  - *Information items:* books
  - *Devices:* camera, projector
  - *Modalities:* visual

The first application that we provide is the configuration tool, which will allow developers to register a new augmented user interface to the DUI2 framework. By sending, the configuration details to the DUI2 framework, an augmented user interface is added to the CMT server as an action and the user interface is also added to the MUI framework as a new user interface. The calls to the different components are handled by our DUI2 framework. The interaction between our configuration tool which is implemented as an HTML form, happens by calling the REST interface via the spring RestTemplate library[1]. The configuration tool can send the configuration details for registering the augmented user interfaces via the `DUI2/aug/config` endpoint that DUI2 provides. The DUI2 component forwards the request first to CMT in order to add the user interface as a fact and secondly to MUI in order to add the new user interface to the management component. The sequence diagram for the registration of a new augmented user interface is illustrated in Figure 5.1.

The configuration tool sends a request via the "config" endpoint to the DUI2 framework. The DUI2 framework will use the sent data of the augmented user interface in order to add the augmented user interface as a fact to the CMT server, this is done via the addFact. In the CMT server, the fact is compiled at runtime and added to the database and to the knowledge base of the Drools rule engine. Afterwards the augmented user interface is sent to the MUI framework by using the addUI request. The MUI framework will create a new user interface stating that it is an augmented user interface by keeping it in the "AUI" collection and doing the necessary calls to the OC2 framework. Now the user interface is added. However the properties of the user interface are not yet added to the UI. A new property is created with the "configProp" tag in order to state that these are the configuration properties of the augmented user interface, because many other properties can be linked to the user interface. By calling the addUIProperty endpoint

[1]`http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html`

with the user interface id and the property id, the properties are added to the augmented user interface.

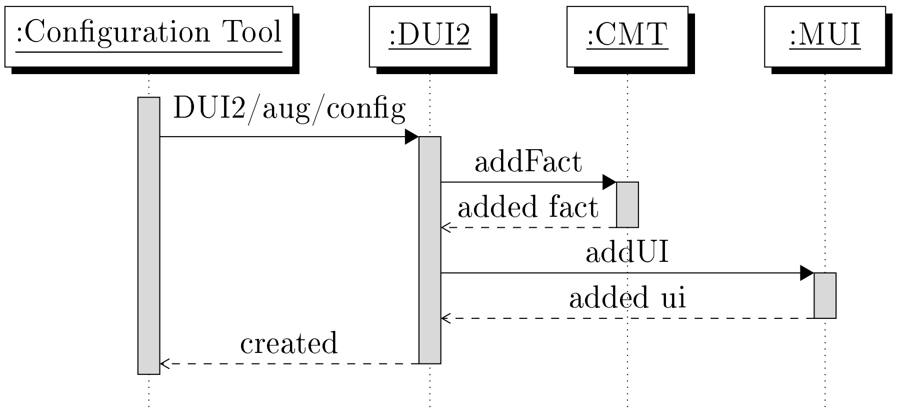
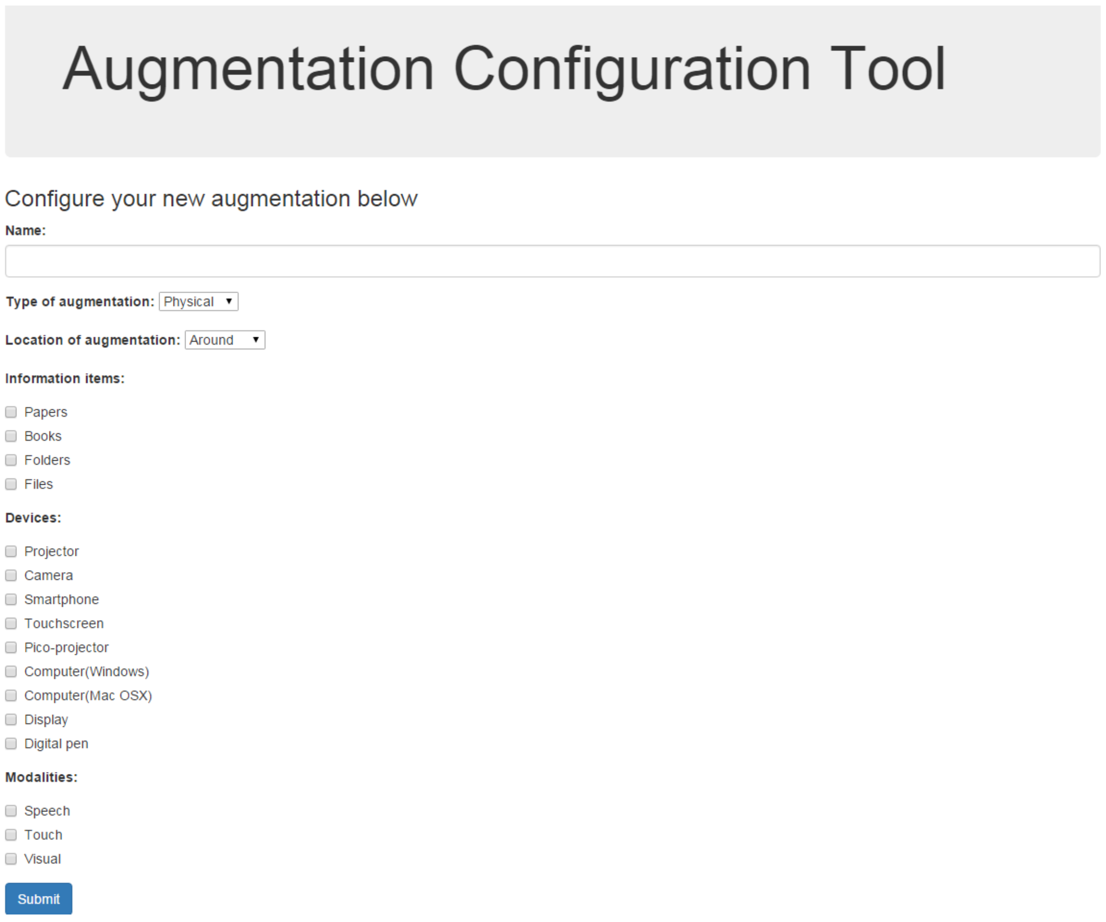


Figure 5.1: Sequence diagram for registering an augmented user interface

The configuration tool is a website with an HTML form where developers can register new augmented user interfaces. Remembering our required augmented user interfaces from the use case. we will describe the process to register an augmented user interface by taking one of them as example. John has developed a new augmented user interface that augments paper by projecting digital post-its around it with notes of the user. When entering the application via the dedicated web page, John is presented with a form where he first has to enter the name of the augmented user interface. Then he is able to give the configuration details by selecting the appropriate values in the drop-down menus and checkboxes. John selects that the augmentation is digital, because the post-its are digital representations. The location of the augmented user interface can be around, overlaid or virtual, in this case John selects "around". In the devices, he must check the checkboxes of the devices that are required in order to show the augmented user interface. The same is repeated for the modalities.

After entering all the configuration details, John presses the 'Submit' button and the augmented user interface is sent to the DUI2 framework in order to be stored. The user interface of the configuration tool is shown in Figure 5.2.



Augmentation Configuration Tool

Configure your new augmentation below

Name:

Type of augmentation: Physical

Location of augmentation: Around

Information items:

- [ ] Papers
- [ ] Books
- [ ] Folders
- [ ] Files

Devices:

- [ ] Projector
- [ ] Camera
- [ ] Smartphone
- [ ] Touchscreen
- [ ] Pico-projector
- [ ] Computer(Windows)
- [ ] Computer(Mac OSX)
- [ ] Display
- [ ] Digital pen

Modalities:

- [ ] Speech
- [ ] Touch
- [ ] Visual

Figure 5.2: User interface of the configuration tool

5.3 Modelling Tool

Besides the configuration tool, we also introduce a second application, namely the modelling tool, which is an Android application. This tool will allow end users to easily model rules for the adaptation of augmented user interfaces depending on contextual elements. We implemented an Android application where users can model context, making it possible to model and create new rules for the adaptation. In our requirements, we stated that there needs to be granularity of context components. As illustrated in Figure 5.3, we provide the granularity by allowing a user to choose between different types of context conditions, like persons, locations, objects, etc. Coming back to our use case, Sandra is working on her paper. When she goes to the CISA meeting, she only takes the paper with her without the post-its. However, with our modelling tool Sandra is able to construct a new rule that states that when Sandra is in a meeting and the paper is present, the digital post-its user interface needs to be shown.

5.3.1 Modelling a Rule

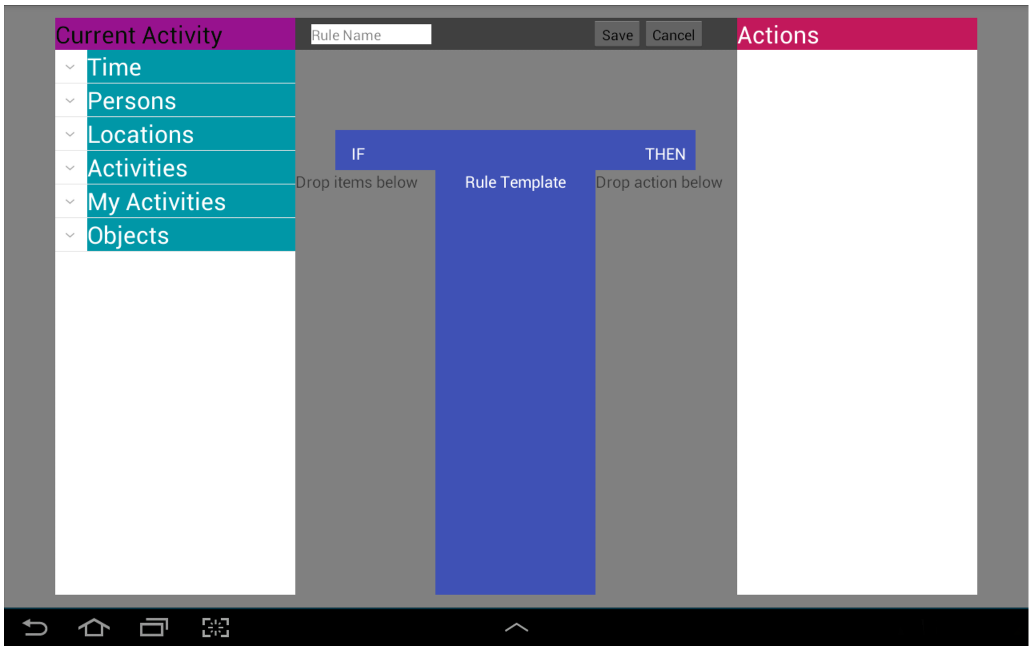


Figure 5.3: User interface of the Android application

Sandra opens the Android application and she is presented with the screen shown in Figure 5.3. The layout of the screen is kept simple, since end users with little or no programming skills need to be able to model the rules using this application. In the left-hand side of the user interface, a list of context elements is shown that can be selected and dragged. Sandra is able to choose between different types of context elements. In the middle part of the UI, the "workspace" of the user is shown. Since, we are using the IF-THEN template approach, we provided an appropriate user interface. Context elements can be dragged to the grey area on the left of the skeleton for our rule. If the wrong elements are selected, the user is able to clear the screen by pushing the 'cancel' button or can simply remove the wrong context elements by pressing the 'X' button. Depending on the context elements that are shown, the list of actions will be filled up with compatible actions. In the use case, Sandra drags the elements (i.e. Sandra, Beat, Reinout, Achmed, living, meeting and CHI paper) to the left grey dropzone. Whenever, she added one of the elements, the list of actions got an update showing the compatible user interfaces corresponding with the context elements. By doing so, we guide the user to a compatible user interface, as stated in the requirements. After all the context elements are inserted Sandra can select the user interface, she wants to be shown. In order to save the rule, we give it a name by entering it in the corresponding name field. Afterwards we can push the 'save' button in order to save the newly created rule.

5.3.2 Android Implementation Details

- **Drag and drop:** On the left and right side of the applications, two listviews are shown. By dragging and dropping, a new rule is created. One major difficulty encountered was the drag and drop functionality. In android, when a user drags an element the drag-event is detected by the listeners. In this case their are two listeners, since we are able to drag items to the IF-part and to THEN-part. However, we only want elements from the left side in the left dropzone and from the right-side in the right dropzone. In order to distinguish between the events, the information in the drag-event needed to be explored. The only useful information inside of the drag-event is the classname of the starting position in which the item was selected. Since both lists are listviews, the only way to work around the issue was to change one of the listviews to a gridview in order to distinguish. In the Listing below, the corresponding code for the drag and drop for the left-hand side is presented. Whenever the selected context element is dragged across the dropzone, the color of the dropzone changes from grey to orange, to indicate that the drag operation is valid.

- **Save** Whenever the 'save' button is pushed to create a new rule, the application uses the spring RestTemplate[2] library to send a POST request to the DUI2 framework, which takes care of it.

[2]http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html

```
dropzone.setOnDragListener(new OnDragListener() {
    public boolean onDrag(View v, DragEvent event) {
        if(event.getLocalState().getClass().toString().equals("class␣
            android.widget.GridView")){
            switch (event.getAction()) {
                case DragEvent.ACTION_DRAG_ENTERED:
                    v.setBackgroundColor(Color.rgb(237, 165, 0));
                    break;

                case DragEvent.ACTION_DRAG_EXITED:
                    v.setBackgroundColor(Color.rgb(128, 128, 128));
                    break;

                case DragEvent.ACTION_DRAG_STARTED:
                    return processDragStarted(event);

                case DragEvent.ACTION_DROP:
                    v.setBackgroundColor(Color.rgb(0, 151, 167));
                    return processDrop(event);
            }
            return false;
        }
        return false;
    }
});
```

- **Asynchronous task** Via the AsyncTask object, we performed a background operation to retrieve the context elements. The GET request `getActivities` is executed in the background. When the background process is done, the listview will be updated with the results from the GET request.

5.3.3 DUI2 Implementation Details

The modelling tool allows users to compose new rules and add them to the DUI2 framework, which will add it to the CMT server. In order for a user to compose a new rule, he has to select context elements from the left-hand side. These context elements are inserted asynchronously in the gridview by calling the `getActivities` endpoint, which returns all possible elements. The user is able to select and drag the context conditions to the specified dropzone. Whenever a context condition is dropped, the list of dropped conditions is send to the DUI2 framework with a POST request to the `sendActivities` endpoint. In the Core component from the DUI2 framework, the function getCompatibleUIs from the Util Component is called with the dropped list as argument. This function will call the MUI component via the DUI2MUI component of our DUI2 framework, in order to check if the context conditions match the properties of the augmented user interfaces. Only compatible augmented user interface are returned and are sent to the Android application. These compatible user interfaces are inserted into the listview on the right side of the screen. A corresponding action can be selected and send to the DUI2 framework via the `selectAction` end point. After all the required information is entered for the creation of a rule, it is sent to the DUI2 framework. The framework will convert the template rule to a DRL rule in the CMT Client module. The created DRL-rule is then added to the CMT server. In Figure 5.4, the sequence diagram for the modelling of a new rule is shown. The `sendActivities` call can be called multiple times, if multiple context elements are added.

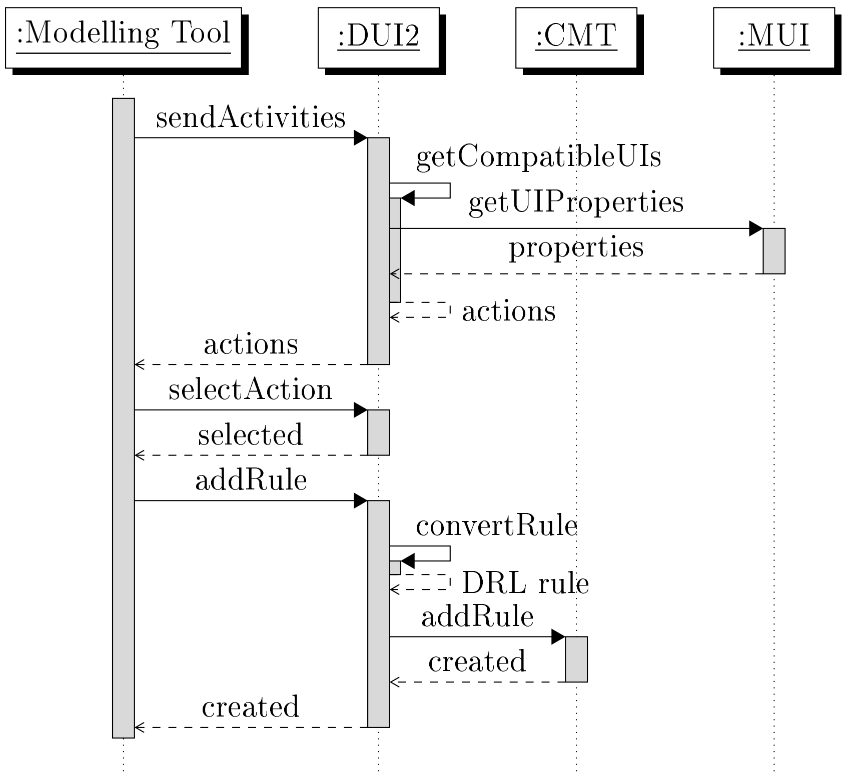


Figure 5.4: Sequence diagram for creating a rule

6 Conclusions and Future Work

In this thesis, the goal was to develop a framework that will allow users to adapt their augmented user interfaces. We provided the DUI2 framework which supports two approaches for the adaptation, namely automatic adaptation and user adaptation. The automatic adaptation will take the current context into account and depending on the rule knowledge base of our framework and matched properties of our UI4A2 context model, the corresponding augmented user interface will be shown. On the other hand, the user adaptation will allow end users to model their own rules for adapting augmented user interfaces. The framework uses the idea of templates in order to facilitate the creation of new rules. Simple rules can be defined which contain context elements and a corresponding action. In our case the corresponding action is limited to only perform a 'show user interface' operation. Users are able to select the user interface that needs to be shown. We combined our framework with the Context Modelling Toolkit, which takes care of the context-awareness and the reasoning over context elements and context rules. In order to perform the automatic adaptation, we extensively researched the field of Augmented User Interface in office settings. We defined a context model, named UI4A2, which provided us with a way to reason about the current environment.

Besides the CMT framework, we also used the OC2 framework as an underlying structure to be able to manage our user interfaces. The OC2 framework provided us with the functionality to link entities. Nevertheless, in order to store our user interface, we created the MUI framework that provides a RESTFul interface to communicate with. The MUI framework consist of a model (i.e. MUI model) that is mapped to the underlying OC2 model. In essence, the OC2 model is used as a metamodel for our MUI framework. The whole DUI2 framework consists of our MUI framework, CMT framework, the connections between the frameworks and the internal core functionality that is provided towards the users. The DUI2 framework allows third party applications to easily communicate with our framework via the RESTful interface. By doing so, developers are able to build their own end user applications.

For our use case, we implemented two client applications, namely the configuration and modelling tool. The configuration tool provides the functionality to register new augmented user interface to the developers of the augmented user interfaces. The second application is the modelling tool, which provides the functionality to allow end users with little or no programming knowledge to model their own context rules. For example, creating a rule that when Bob and Alice are in the office discussing about the "Docudesk" paper, the augmented user interface "AugDocu" should be shown. This is a simple rule that allows the end users some control over the system without having all the control, because we also provide the option to automatically adapt the user interface depending on the current context. The DUI2 framework's main goal is to provide a framework where the balance of control is just right between the user and system.

In order for the users to be able to easily compose new rules, we created an Android application that will allow users with no or little programming skills to construct new rules. By making use of templates, simple rules can be defined which contain context elements and a corresponding action (i.e. show user interface).

6.1 Contributions

The major contributions of this thesis are the DUI2 framework, the MUI framework and the two client applications that communicate with the DUI2 framework. In the DUI2 framework, we have created support for two approaches, namely automatic and end user adaptation. In order for our framework to support these approaches, we integrate the Context Modelling

Toolkit in our framework. The CMT framework allows us to reason over context and context rules, but also support the creation of new facts, rules, etc. After studying the different kind of Augmented Reality applications in office settings, we introduced a context model for the implementation of the automatic adaptation. The UI4A2 model allows the system to reason over the current context and to automatically apply a rule when the current context matches. The end user adaptation happens by allowing users to model new rule for adaptation. Via a simple rule-template, users can easily create new rules without any programming skills. A second contribution is the MUI framework that was introduced to manage the user interfaces. The MUI model takes care of the management of our user interfaces. In order to store them, we used the OC2 framework as an underlying metamodel for our MUI model. BY doing so, we are able to map all of the components of our MUI model to the OC2 model. By using the OC2 framework, we are able to link user interfaces and objects to each other. A last major contribution are the two client applications that were developed to interact with the DUI2 framework. Note that third parties can develop their own applications by using the RESTful interface that is provided. The first application that we introduced is the configuration application that takes care of the registration of new augmented user interfaces. The application is mostly directed at developers that have created new user interfaces. The second application is the modelling tool that allows end users to model and create their own rules. No programming skills are required, the Android application uses simple drag and drop movements to model the rules.

6.2 Future Work

Our framework is still in the early stages and can be further improved. One of the improvements is to broaden the scope of the framework by allowing different kinds of user interfaces to be adapted. Our framework is easily extendable to allow this broader scope. The configuration and modelling tool that were developed to show the interactions with the framework, can be extended or replaced by third party applications, since our RESTful interface allows the development of external applications. The user interface of the configuration tool can be made more colorful and intuitive, the configuration details that are inserted can be further extended. For the automatic adaptation, only the properties of the current location are being checked against the adaption rules in the knowledge base. In the modelling tool, we allow users to select different types of context elements, these elements can be further extended, depending on the context of use of the framework. After some improvements, an evaluation of the framework can be performed for checking the advantages or disadvantages users experience when using the framework. Further studies can focus more on the balance between user and system control.

Bibliography

[1] Ronald T. Azuma. A Survey of Augmented Reality. *Presence*, 6(4):355–385, August 1997.

[2] Jakob Bardram, Sofiane Gueddana, Steven Houben, and Soren 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, 2012.

[3] Jakob E. Bardram. The Java Context Awareness Framework (JCAF) – a Service Infrastructure and Programming Framework for Context-Aware Applications. In *Proceedings of PERVASIVE 2005, 3th international conference on Pervasive Computing*, pages 98 – 115, Munich, Germany, May 2005.

[4] Victoria Belotti and Keith Edwards. Intelligibility and Accountability: Human Considerations in Context-Aware Systems. *Human-Computer Interaction*, 16(2 –4):193 – 212, February 2001.

[5] Silvia Berti, Francesco Correani, Fabio Paternó, and Carmen Santoro. The TERESA XML Language for the Description of Interactive Systems at Multiple Abstraction Levels. In *Proceedings of AVI 2004, International Conference on Advanced Visual Interfaces*, pages 103 – 110, Gallipoli, Italy, May 2004.

[6] Xiaojun Bi, Tovi Grossman, Justin Matejka, and George Fitzmaurice. Magic Desk: Bringing Multi-Touch Surfaces into Desktop Work. In *Proceedings of CHI 2011, Conference on Human Factors in Computing Systems*, pages 2511–2520, Vancouver, Canada, May 2011.

[7] Florian van 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, 2013.

[8] 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, 2011.

[9] David Chen, Sam Tsai, Cheng-Hsin Hsu, Jatinder Pal Singh, and Bernd Girod. Mobile Augmented Reality for Books on a Shelf. In *Proceedings of ICME 2011, IEEE International Conference on Multimedia and Expo*, pages 1–6, Barcelona, Spain, July 2011.

[10] Guanling Chen and David Kotz. A Survey of Context-Aware Mobile Computing Research. Technical report, Dartmouth College, Hannover, USA, 2000.

[11] Jidong Chen, Hang Guo, Wentao Wu, and Wei Wang. iMecho: an Associative Memory Based Desktop Search System. In *Proceedings of CIKM 2009 , 18th Conference on Information and Knowledge Management*, pages 731–740, Hong Kong, November 2009.

[12] Danny Crasto, Amit Kale, and Christopher Jaynes. The Smart Bookshelf: A Study of Camera Projector Scene Augmentation of an Everyday Environment. In *Proceedings of WACV/MOTIONS 2005, 7th IEEE Workshops on Application of Computer Vision/ IEEE Workshop on Motion and Video Computing*, volume 1, pages 218–225, Breckenridge, USA, January 2005.

[13] Anind K. Dey and Gregory D. Abowd. CybreMinder: A Context-Aware System for Supporting Reminders. In *Proceedings of HUC 2000, 2nd International Symposium on Handheld and Ubiquitous Computing*, pages 172 – 186, Bristol, UK, September 2000.

[14] Anind K. Dey and Alan Newberger. Support for Context-Aware Intelligibility and Control. In *Proceedings of CHI 2009, Conference on Human Factors in Computing Systems*, pages 859 – 868, Boston, USA, April 2009.

[15] Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. iCAP: Interactive Prototyping of Context-Aware Applications. In *Proceedings of PERVASIVE 2006, 4th International Conference on Pervasive Computing*, pages 254 – 271, Dublin, Ireland, May 2006.

[16] Niklas Elmqvist. Distributed User Interfaces: State of the Art. In *Proceedings of DUI 2011, 1st Workshop on Distributed User Interfaces*, pages 1–12, Vancouver, Canada, May 2011.

[17] Katherine M. Everitt, Meredith R. Morris, Bernheim A. J. Brush, and Andrew D. Wilson. Docudesk: An Interactive Surface for Creating and Rehydrating Many-to-Many Linkages Among Paper and Digital Documents. In *Proceedings of ITS 2008, Third IEEE International Workshop on Tabletops and Interactive Surfaces*, pages 25–28, Amsterdam, The Netherlands, October 2008.

[18] Stephen Fitchett, Andy Cockburn, and Carl Gutwin. Improving Navigation-Based File Retrieval. In *Proceedings of CHI 2013, Conference on Human Factors in Computing Systems*, pages 2329–2338, Paris, France, April 2013.

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

[20] Tao Gu, Xiao H. Wang, Hung K. Pung, and Da Q. Zhang. An Ontology-based Context Model in Intelligent Environments. In *Proceedings of CNDS 2004, Communication Networks and Distributed Systems Modelling and Simulation Conference*, pages 270–275, San Diego, USA, January 2004.

[21] Jim Hahn. Mobile Augmented Reality Applications for Library Services. *New Library World*, 113(9/10):429–438, 2012.

[22] 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, 2014.

[23] Bob Hardian, Jadwiga Indulska, and Karen Henricksen. Balancing Autonomy and User Control in Context-Aware Systems - a Survey. In *Proceedings of PERCOM 2006, 4th annual IEEE international conference on Pervasive Computing and Communications*, pages 51 – 56, Pisa, Italy, March 2006.

[24] Valentin Heun, James Hobin, and Pattie Maes. Reality Editor: Programming Smarter Objects. In *Proceedings of UbiComp 2013, International Conference on Ubiquitous Computing*, pages 307 – 310, Zurich, Switzerland, September 2013.

[25] Valentin Heun, Shunichi Kasahara, and Pattie Maes. Smarter Objects: Using AR Technology to Program Physical Objects and Their Interactions. In *Proceedings of CHI 2013, Conference on Human Factors in Computing Systems*, pages 961–966, Paris, France, May 2013.

[26] Tom Holland and Aaron Quigley. MobARDoc: Mobile Augmented Printed Documents. In *Proceedings of UbiComp 2011, 13th International Conference on Ubiquitous Computing*, volume 10, pages 26–29, Bejing, China, September 2010.

[27] Steven Houben and Nicolai Marquardt. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of CHI 2015, Conference on Human Factors in Computing Systems*, pages 1247–1256, Seoul, Republic of Korea, 2015.

[28] Jan Humble, Andy Crabtree, Terry Hemmings, Karl-Petter Åkesson, Boriana Koleva, Tom Rodden, and Pär Hansson. "Playing with the Bits" User-configuration of Ubiquitous Domestic Environments. In *Proceedings of UbiComp 2003, 5th International Conference on Ubiquitous Computing*, pages 256 – 263, Seattle, USA, October 2003.

[29] Daisuke Iwai and Kosuke Sato. Limpid Desk: See-through Access to Disorderly Desktop in Projection-based Mixed Reality. In *Proceedings of VRST 2006, Symposium on Virtual Reality Software and Technology*, pages 112–115, Cyprus, November 2006.

[30] Daisuke Iwai and Kosuke Sato. Document Search Support by Making Physical Documents Transparent in Projection-Based Mixed Reality. *Virtual reality*, 15(2-3):147–160, June 2011.

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

[32] Hans-Christian Jetter, Michael Zollner, 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.

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

[34] William Jones, Dawei Hou, Bhuricha Deen Sethanandha, Sheng Bi, and Jim Gemmell. Planz to Put Our Digital Information in Its Place. In *Proceedings of CHI 2010, Conference on Human Factors in Computing Systems*, pages 2803–2812, Atlanta, USA, April 2010.

[35] Fakhreddine Karray, Milad Alemzadeh, Jamil A. Saleh, and Mo N. Arab. Human-Computer Interaction: Overview on State of the Art. *International Journal on Smart Sensing and Intelligent Systems*, 1(1), March 2008.

[36] Mohammadreza Khalilbeigi, Jürgen Steimle, Jan Riemann, Niloofar Dezfuli, Max Mühlhäuser, and James D Hollan. ObjecTop: Occlusion Awareness of Physical Objects on Interactive Tabletops. In *Proceedings of ITS 2013, ACM International Conference on Interactive Tabletops and Surfaces*, pages 255–264, St Andrews, UK, October 2013.

[37] Jiwon Kim, Steven M. Seitz, and Maneesh Agrawala. Video-Based Document Tracking: Unifying Your Physical and Electronic Desktops. In *Proceedings of UIST 2004, 17th Symposium on User Interface Software and Technology*, pages 99–107, 2004.

[38] Jisoo Lee, Luis Garduno, Erin Walker, and Winslow Burleson. A Tangible Programming Tool for Creation of Context-Aware Applications. In *Proceedings of UbiComp 2013, International Conference on Ubiquitous Computing*, pages 391 – 400, Zurich, Switzerland, September 2013.

[39] Sang-won Leigh, Philipp Schoessler, Felix Heibeck, Pattie Maes, and Hiroshi Ishii. THAW: Tangible Interaction with See-Through Augmentation for Smartphones on Computer Screens. In *Proceedings of TEI 2015, International Conference on Tangible, Embedded, and Embodied Interaction*, pages 89–96, Stanford, USA, 2015.

[40] James Lin and James A. Landay. Employing Patterns and Layers for Early-Stage Design and Prototyping of Cross-Device User Interfaces. In *Proceedings of CHI 2008, Conference on Human Factors in Computing Systems*, pages 11313 – 1322, Florence, Itlay, April 2008.

[41] Nishant Malhotra, Aayushi Singh, J. DivyaKrishna, Kanika Saini, and Neeraj Gupta. Context-aware Library Management System Using Augmented Reality. *International Journal of Electronic and Electrical Engineering*, 7(9):923 – 929, 2014.

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

[43] Kazuhiro Matsushita, Daisuke Iwai, and Kosuke Sato. Interactive Bookshelf Surface for in Situ Book Searching and Storing Support. In *Proceedings of AH 2011, 2nd Augmented Human International Conference*, number 2, pages 1 – 8, Tokio, Japan, March 2011.

[44] Ian C. McIlwaine. The Universal Decimal Classification: Some Factors Concerning Its Origins, Development, and Influence. *Journal of the American Society for Information Science*, 48(4):331–339, 1997.

[45] Jeremie Melchior, Donatien Grolaux, Jean Vanderdonckt, and Peter Van Roy. 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, 2009.

[46] Jan Meskens, Jo Vermeulen, Kris Luyten, and Karin Coninx. Gummy for Multi-Platform User Interface Designs: Shape me, Multiply me, Fix me, Use me. In *Proceedings of AVI 2008, International Conference on Advanced Visual Interfaces*, pages 233 – 240, Naples, Italy, May 2008.

[47] Pranav Mistry and Pattie Maes. Augmenting Sticky Notes as an I/O Interface. In *Proceedings of CHI 2009, Conference on Human Factors in Computing Systems*, pages 547–556, Boston, USA, April 2009.

[48] Ann Morrison, Alessandro Mulloni, Saija Lemmelä, Antti Oulasvirta, Giulio Jacucci, Peter Peltonen, Dieter Schmalstieg, and Holger Regenbrecht. Collaborative use of mobile augmented reality with paper maps. *Computers & Graphics*, 35(4):789–799, 2011.

[49] Vivian G. Motti, Dave Raggett, Sascha Van Cauwelaert, and Jean Vanderdonckt. Simplifying The Development of Cross-Platform Web User Interfaces by Collaborative Model-Based Design. In *Proceedings of SIGDOC 2013, 31st International Conference on Design of Communication*, pages 55 – 64, Greenville, USA, September 2013.

[50] Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. CrowdAdapt: Enabling Crowdsourced Web Page Adaptation for Individual Viewing Conditions and Preferences. In *Proceedings of EICS 2013, 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 23 – 32, London, UK, June 2013.

[51] Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. W3Touch: Metrics-Based Web Page Adaptation for Touch. In *Proceedings of CHI 2013, Conference on Human Factors in Computing Systems*, pages 2311 – 2320, Paris, France, April 2013.

[52] Moira C. Norrie. An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In *Proceedings of ER 1993, International Conference on the Entity-Relationship Approach*, pages 390–401, Arlington, USA, December 1994.

[53] Jongmoon Park, Hong-C. Lee, and Myung-J. Lee. JCOOLS: A Toolkit for Generating Context-Aware Applications with JCAF and DROOLS. *Journal of Systems Architecture*, 59(9):759 – 766, 2013.

[54] Jan Riemann, Mohammadreza Khalilbeigi, and Max Mühlhäuser. PeriTop: Extending Back-projected Tabletops with Top-projected Peripheral Displays. In *Proceedings of ITS 2013, 31th International Conference on Interactive Tabletops and Surfaces*, pages 349 – 352, St Andrews, Scotland, October 2013.

[55] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of CHI 1999, Conference on Human Factors in Computing Systems*, pages 434 – 441, Pittsburgh, USA, May 1999.

[56] Bill Schilit, Norman Adams, and Roy Want. Context-Aware Computing Applications. In *Proceedings of WMCSA 1994, First Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE, 1994.

[57] Johannes Schöning, Michael Rohs, Sven Kratz, Markus Löchtefeld, and Antonio Krüger. Map Torchlight: a Mobile Augmented Reality Camera Projector Unit. In *Proceedings of CHI 2009, Conference on Human Factors in Computing Systems*, pages 3841–3846, Boston, USA, April 2009.

[58] Thomas Seifried, Matthew Jervis, Michael Haller, Masood Masoodian, and Nicolas Villar. Integration of Virtual and Real Document Organization. In *Proceedings of TEI 2008, 2nd International Conference on Tangible and Embedded Interaction*, pages 81 – 88, Cambridge, UK, February 2008.

[59] Adrian Shatte, Jason Holdsworth, and Ickjai Lee. Mobile Augmented Reality Based Context-Aware Library Management System. *Expert Systems with Applications*, 41(5):2174–2185, April 2014.

[60] Beat Signer and Moira C. Norrie. As We May Link: A General Metamodel for Hypermedia Systems. In *Proceedings of ER 2007, 26th International Conference on Conceptual Modeling*, pages 359–374, Auckland, New Zealand, November 2007.

[61] Beat Signer and Moira C. Norrie. PaperPoint: A Paper-Based Presentation and Interactive Paper Prototyping Tool. In *Proceedings of TEI 2007, First International Conference on Tangible and Embedded Interaction*, pages 57–64, Baton Rouge, USA, February 2007.

[62] Norbert A. Streitz, Jörg Geissler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-LAND: An interactive Landscape for Creativity and Innovation. In *Proceedings of CHI 1999, Conference on Human Factors in Computing Systems*, pages 120–127, Pittsburgh, USA, 1999.

[63] 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 JCDL 2014, 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 87–96. IEEE, 2014.

[64] 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, 33rd International Conference on the Entity-Relationship Approach*, pages 313–320, Altanta, USA, October 2014.

[65] Khai N. Truong, Elaine M. Huang, and Gregory D. Abowd. CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home. In *Proceedings of UbiComp 2004, 6th International Conference on Ubiquitous Computing*, pages 143 – 160, Nottingham, UK, September 2004.

[66] Blase Ur, Elyse McManus, Melwyn P. Y. Ho, and Michael L. Littman. Practical Trigger-Action Programming in the Smart Home. In *Proceedings of CHI 2014, Conference on Human Factors in Computing System*, pages 803 – 812, Toronto, Canada, May 2014.

[67] van der Heijden, Hans. Ubiquitous Computing, User Control, and User Performance: Conceptual Model and Preliminary Experimental Design. In *Proceedings of RSEEM 2003, 10th Research Symposium on Emerging*

Electronic Markets, pages 107 – 112, Bremen, Deutschland, September 2003.

[68] Stephen Voida and Saul Greenberg. WikiFolders: Augmenting the Display of Folders to Better Convey the Meaning of Files. In *Proceedings of CHI 2009, Conference on Human Factors in Computing Systems*, pages 1679–1682, Boston, USA, April 2009.

[69] Karl Voit, Keith Andrews, and Wolfgang Slany. Tagging Might not be Slower Than Filing in Folders. In *Proceedings of CHI 2012, Conference on Human Factors in Computing Systems*, pages 2063 – 2068, Austin, Texas, May 2011.

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