Vrije Universiteit Brussel

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

Big Brother is Watching Your Documents: A Framework to Manage Tracked Physical Data

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

Ayrton Vercruysse

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

Academic year 2014-2015

Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN
VAKGROEP COMPUTERWETENSCHAPPEN

Big Brother is Watching Your Documents: A Framework to Manage Tracked Physical Data

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

Ayrton Vercruysse

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

Academiejaar 2014-2015

Abstract

Nowadays people are still having a hard time to keep track of their documents. With the upcoming of the digital age, people thought that organising their personal information would become easier. Paper would be replaced by digital documents and re-finding in the digital space would be a lot easier than in the physical world. Unfortunately, the physical world never disappeared and the digital world just got added to the pile, which made the problem only bigger. To top things, re-finding in the digital world is not as easy as it was thought. These problems opened up an entire field of research on how people could keep track of their personal information, called Personal Information Management (PIM). Even though the study field of PIM has been around for a while, still not all the problems have been tackled. In this thesis, a part of a framework called the KOR (Keeping-Organising-Re-finding) framework, is introduced. The focus will lie on the keeping, or tracking of documents, more specifically on physical documents. In contrary to existing PIM frameworks, our framework will take the organisational structure that contains the document into account. This opens up a wide variety of possibilities to interact with the tracked documents.

To create a framework that is able to track organisational structures, the problem was divided in three sub problems, where each and every one of them was focussed on extensibility. First of all, a framework was needed to keep the information about the documents. To implement this framework a choice has been made to use the OC2 system as an underlying system to store document information, but some additional information had to be kept for the physical documents. The framework requires an extension of OC2 by means of a mapping on the existing system. This extension together with a part of the OC2 system has then been opened up to the world by implementing a REST interface on for it. Adding the REST interface meant that the system was usable for users to create their own tracking system using the framework. Next to the extension of OC2, we have developed a tracking component. The goal of this component is to make use of data received by tracking technologies and store the right information in the underlying OC2 extended component. Also this REST interface is widely usable for users to create their own tracking mechanisms.

The tracking mechanisms developed in this project were as a proof of concept to illustrate how the framework could be used. Nevertheless, also here extensibility was an important requirement. The tracking component is built out of two components being the identification, which boils the document down to a unique recognising entity, and a way to compare these entities. Both the recognising side and the comparing side are coupled as

loosely as possible which enables the use of one comparing component with multiple recognising components. The loose coupling between the recognising and comparing components gives a user a lot of freedom to invent their own recognising tools. However, nothing stops them from inventing their own comparing algorithms.

Declaration of Originality

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

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

Acknowledgements

First of all I would like to thank my promoter Prof. Dr. Beat Signer for giving me the possibility to write my thesis in the WISE lab, his feedback during presentations and his proofreading of this document. Equally important is my thesis supervisor Ph.D. student Sandra Trullemans who provided constant guiding during the creation of this thesis. Next to this, she gave me and her other thesis students the opportunity to use the WISE lab as a place we could work, discuss and create test set-ups.

Because the WISE lab was a nice place to get some work done, I would like to thank my thesis colleagues, Jasmien De Ridder, Audrey Sanctorum and Tim Reynaert. Working in a group is always more fun than alone, and as they were working in the same field of research, they were perfect rubber ducks. Because it does not always need to be work and no fun, it was easy to go for some distractions with them. This distraction could be as simple as going for lunch together.

I would also like to thank Infogroep and its members. Infogroep was a nice place to escape to for half an hour when the inspiration at the lab ran dry. It helps to get some distraction from the thesis, even though, every time the Infogroep room was entered the first question that was popped was: "How is the thesis going?"

For helping to really clear my head I would like to thank my dogs. Going for a walk with them in the countryside really helps to clear your mind. Even though, when walking them alone in the nice and quiet surrounding, your mind starts to wonder off and the first thing it hits is what is occupying it most, your thesis. This resulted that not only my head could be cleared, but also a lot of ideas for my thesis popped in my head during some of these walks.

The next person I want to thank is my seven months old nephew of which I am godfather. He is probably the only person during the last year who did not ask me "How is your thesis going?" or "How is it going at school?". Not that I am not proud of my work, or do not want to talk about it, but these are probably the questions I heard most during the last year, so one person that did not ask about it was a relief. In addition to this, when being occupied by a seven months old child, in contrary than taking your dogs for a walk, your mind does not have time to wonder off, and so he was the best way to clear my head.

Last, but certainly not least, I would like to thank my parents. They gave me the possibility to do these studies at the first place and during my studies they were a constant support. Both during exams and during the writing of this thesis.

Contents

1

Introduction

Times are changing, people say the physical age is coming to an end and the digital age will take its place. This is mainly true for personal information. Where documents used to be written, phone numbers kept in a phone book and communication done by sending a letter through mail, now documents are often digitally stored on the computer, contacts are kept on the smartphone or a digital phonebook on the computer and letters are digitally sent using e-mail. The physical age has not yet come to an end though. In office environments there are still a lot of documents printed and stored. Invoices often come through mail and not everyone fills in their taxes on the Web. This results in an in-between situation, where more and more will be done digitally, but still a lot is done on physical documents. The main goal of the digital age would be that keeping track of information is easier. It is easier to search through, easier to browse in and easier to order. This idea seems to be slightly wrong. People have huge difficulties in keeping track of their information in the digital space. Having parts of their documents still in a physical way and no way to interact between digital and physical documents only makes things harder.

1.1 Why Trees Still Need to Fear

People think the digital age is coming and physical documents will in the near future disappear form our lives. Sellen and Harper [50] wrote in 2002 a book called "The Myth of the Paperless Office" in which they stated that paper will never leave our lives. Sellen and Harper state that the advantages of paper are still relevant enough for people to keep using it. Some of these advantages are that paper is affordable. Users can print a lot of paper before they have spent as much money as a laptop or tablet costs. For some people writing on paper feels more permanent and trusted. They are afraid of running out of power on their laptop or think their document will be deleted without them wanting it to be deleted. Paper is also said to be better to think about. Writing has a better reminding function than typing. This has been scientifically proven because another part of the brain is used when writing than when typing. These different parts of the brain each stimulate different parts of our neurological system [42]. Writing should also help to see the progress that has been made. This is why people, when designing something, for example, when a programmer designs a database model, the programmer will often start doing this on a piece of paper or on a whiteboard. Things can be adjusted on paper and so the trail of thought can be seen. On the other hand, when something has been typed on a computer, changes that have been made erase the old version.

For some people the idea of privacy is taken into account. They feel that a handwritten document offers more privacy than a document on the computer because computers can be hacked. It is already common knowledge that it is safer to write a password down on a piece of paper and store in a drawer rather than writing it down in a document on the computer. Paper also tends to be easier to group, easier to navigate through and it has a good reminding function. It is easier to remember that a document needs to be processed when it is placed upon a desk than when it is hidden in an e-mail folder. Some people also prefer to read on paper rather than reading on a screen. It makes it easier to make annotations, it has a better feel and they remember better what they have read in paper documents than in digital documents.

These findings have been espoused by Whittaker and Hirschberg [56] in 2002. They did research in an office environment to check how far the digital age has come. They found no compelling evidence that paper had no role in modern office work. Whattaker and Hirschberg even stated that younger employees were just as likely to rely on paper than older employees which might indicate that paper documents will not leave the office together with older employees. A remarkable finding of Whittaker an Hirschberg is that

users (at least the ones in their study) think paper information is valuable in the long term. This in contrary to Kidd [32], who stated that, for knowledge workers, paper is a critical method for acquiring information, but not for long time storage.

1.2 Keeping Order in the Chaos

To keep order in the chaos, both digital and physical, there is need to keep users their information ordered. This task is a research field in Computer Science called Personal Information Management which focusses on keeping the personal information of the user clear for the user to use. It has fields focussing on the digital documents of the user, fields focussing on the physical documents of users and fields that want to be able to relate both physical and digital documents with each other when they are related.

Figure 1.1: A messy desk

Concerning the physical part of document tracking, the goal is to bring order into the chaos users can create in their working environment as illustrated in Figure 1.1. The goal is to help users keep track of their physical documents, but to do so be no bigger burden for the users than when they would not use the PIM system and try to re-find their documents in an old-fashioned way. For the digital part of PIM, the users need help to organise and re-find their documents which are stored digitally. In this day and age this gets harder than it was a few decades ago. In past times documents could only be stored on the computer or on floppy disks. In this day and age you have the computer itself, external hard drives, CD roms, USB sticks,

the Internet, the cloud or e-mail. Keeping track of all these documents is a big challenge. Certainly when documents which are stored at entirely different places need to be linked with each other. For example, an e-mail with information about a document needs a link between this document and the e-mail.

When focussing on physical documents, the tracking did not get very far yet. There are systems that mark a document with tags or barcodes to keep track of their location, but this still does not really relate to the way people tend to use documents. This keeps track of individual documents but not of organisational structures and relations between these documents. In 2002 Malone [40] stated that users store their physical documents in organisational structures. He considered the filing and piling organisational structures. In 2014 Trullemans and Signer [52] considered that also mixing could be seen as an organisational structure. Further explanation about these structures will be given in the related work in Chapter 2. Knowing that users group their documents in organisational structures and there is no relation between these organisational structures and the digital representation of documents opens up the question how good the digital representation of physical documents is at the moment.

1.3 Problem Statement

Every tool currently available for storing a digital representation of physical documents in the digital environment does not take the organisational structures into account. One of the only tools that use the links between the physical documents is SOPHYA developed by Jervis in 2010 [22]. In his approach every document is an object and objects can be linked to each other. This opens up the possibility to display some relations between documents but the notion of the organisational structures are not taken into account. The use of organisational structures can open up a wide variety of ways to interact with the stored information about the documents. The fact that certain organisational structures can have certain re-finding cues [32] can be seen as example of this usage. A more elaborate explanation about this cues will follow in this thesis.

A second problem with the current ways of keeping track of physical documents is that they are very strict to use. The document needs to be provided by special hardware in special cases. This hardware is connected with software that can keep track of these documents, but only in a restricted way. As every user has their own environment, their own hardware and their own way of storing documents, the user should have a great amount of

freedom to track these documents. This is why only a framework that keeps track of the information about the documents will be provided and it will be up to the user to use this framework in a correct way to store information about their documents. This enables the user to use the framework in a very personal setting.

Besides storing the information about relations between documents, like organisational structures and interactions with the documents, it is interesting to know in what context the document is being used. This kind of information can help re-finding documents or explore documents. If a user wants to re-find documents related to a certain document, (i.e. having the same context) or the user wants to start working on a project (which is a context), they can use the knowledge of the context in which a document has been used to re-find it. Also some exploring, grouping or augmenting can be done using the contexts. A document is not restricted to one context of course. A document can be used in more than one context, even though it will relate more to one context than to another. This means that for each document not only the relevant contexts need to be stored, but also how important these contexts are in relation to each other.

The framework will be so open that not only the way people communicate with it is versatile, also the framework itself is versatile. It presents a wide variety of knowledge about the document that is being kept, but there is also the possibility for the user to store extra information in the framework if this would be interesting for them.

1.4 Thesis Contribution

In this thesis, the KOR framework is presented. The KOR framework is based on various requirements which, all together, tackle the above-mentioned problems. The goals of this framework can be summarised as follows: an extensible framework that is capable of storing extra information about documents and takes the way documents are stored into account.

In this thesis the main focus lies on the tracking part of the KOR framework. Documents need to be tracked and information about these documents need to be kept up to date whenever some changes are made to the document.

To make the framework extensible the user should have the possibility to add extra functionality to the system, or an easy way to extend the system should be provided. This means that the system needs to be accessible and open for the users to interact.

Even though, the goal is to make an extensible framework, adding additional information about documents in the database should be possible for

the users without using extension. This way users can store information that in their specific case is interesting. This extra information on its turn can be used again to create applications making use of data stored in the system.

One of the main reasons that the systems which are available today do not suffice, is that they do not take the way a document is stored into account. A document can be stored in a folder or a pile. This information can be used in order to re-find documents.

The tracking component needs to store the correct information in the system, but this tracking component needs to get his information from somewhere. As discussed before it is the goal that user provides this information. This can be done by implementing specific tracking techniques. Some tracking techniques have been implemented as a guide for users to inspire them to create their own tracking techniques.

1.5 Thesis Outline

We start by investigating related work. Thereby, we study how people organise their desks, how they can keep track of physicals documents at the moment and how this can work together in an office environment. Once the existing work has been discovered, the requirements for the framework can be written down. These requirements are based on the problem statement and discuss every requirement that needs to be met to conquer the problems stated in the problem statement. Once it is known what needs to be done it is time to see how this can be done. In Chapter 4 the KOR framework is discussed. First the general architecture is explained and discussed. Later on, it is explained how the framework has been implemented. This is done in two major parts. First the part to keep the information in the system is explained. After this, the tracking component that uses this system is discussed. This chapter discusses how interactions with documents are stored in the system. It explains what information is being stored in the system when a document is created, accessed or placed inside another organisational structure. Additionally, the chapter on the KOR framework will also explain some extra functionality that has been implemented to keep track of extra information. The most important thing implemented to keep this extra information are lifelines.

Once the framework is implemented, it is time to check what this framework can do. To do so, first some tracking techniques are implemented. These will only be implemented as a proof of concept to show how the framework can be used. tracking techniques consist out of two parts, recognisers which determine a URI given a document and a comparator which compares these

URIs. In total five identifiers have been implemented, one using OCR, one using SIFT, one using the colour intensity of the document, one using RFID tags and a last one using TUIO tags. For every of these identifiers, a comparator needs to be implemented. But as both RFID and TUIO tags return just an id, they can use the same identifier. This means that there are four comparators, for OCR, SIFT, colour intensity and a string comparator which compares the ids returned by RFID and TUIO tags. Once the tracking techniques have been implemented some use cases to demonstrate their working have been implemented. These use cases are a pile, some paper trays, a ring binder and a file cabinet drawer.

In the last chapter the delivered work is discussed and some possible future work is presented.

"Yeah, but the lost diadem," said Michael Corner, rolling his eyes, "is lost, Luna. That's sort of the point."

J.K. Rowling, Harry Potter and the Deathly Hallows

2 Tracking Your Documents

In the introductory chapter we stated why people still make use of physical documents. In this chapter we be explain how people keep track of these documents. This includes how these documents get stored, re-found or, in the worst case, lost. Documents can be stored in a number of ways, and at various places. Even more, some documents can belong together without being stored together, for example, the digital and physical version of a document, or a document in a mailbox and a document in a folder on the computer. We need to create links between various ways to store documents, so people can keep track where a document is stored. To find the optimal way to store this information, first it is necessary to know how people currently store documents and what technologies are present to help keeping track of these documents.

2.1 How do we Organise?

When people organise their affairs, their main goal is to be able to find back what has been carefully stored. Being able to find back what has been stored is a matter of remembering where things have been put. This leads to looking how the human memory works. Miller [45] stated that the human memory is best at remembering spatial positions. He stated that, as a teacher, it was far easier for him to link the name of a student to his place in the

classroom than to the student's face. This would have been based on one simple principle in mnemotechnics, a branch of applied psychology. This principle states that concrete visual images are easy to remember, especially if they are remarkable or even a little bizarre. This principle has been known to men for a long time. According to some classical authors, this principle has been invented by Simonides, a Greek poet who lived about 500 B.C. There is a legend telling that Simonides who was at a feast was just called away at the moment the room collapsed, and buried all other guests. The bodies were mutilated beyond identification, but Simonides realised that he could recall the place that each person took at the banquet table. This way he could identify the other guests by knowing their spatial location during the feast. Shortly after this tragedy, Simonides invented a mnemonic system inspired by this tragedy.

An additional advantage of spatially represented information is that it is easier to reason about it. A lot of things that not really have a logical spatial representation gets a spatial representation just to help people think and reason about it. Some examples can be found in tree structures in Computer Science or flow charts used by system engineers.

Taking this knowledge to an office environment, one can assume that people have a fair idea where they left items. If piles are being created by type, for example a pile of research papers about PIM and a pile of research papers about Context-aware programming. Then, when a new paper needs to be sorted, by some sort of muscle memory, the user will know on which pile he needs to put this paper to be sorted. Also when willing to re-find things one can say: I have read this in a book somewhere. And by selecting a number of books from the bookshelf, based on the spatial location of this book the user can re-find an episode in a book. Even when flipping through them a person will rather remember it was at the beginning of the book or near the end, or it was at the top of a page, or at the bottom rather than recalling the chapter or content.

The first real research done on how people organise their desk was by Malone [40][34] in 1983. Malone interviewed a number of users about how they organise their desk. After the interview, the users were asked to find some documents in their offices. These documents were chosen by their colleagues based on how easy or hard they think it would be for the user to find them.

Malone concluded that there are two major units of office arrangement, filing and piling. Filing and piling are two ways to collect groups of elements into larger units. Files are explicitly titled and arranged in some order (e.g. chronological, contextual or alphabetical). Piles on the other hand con-

tain documents which are not necessarily titled and they are not arranged in any particular order. The pile itself is not named and often not arranged in any particular order on the desk. As stated before, their spatial location is often important in re-finding them.

Table 2.1 gives an overview of Malone's findings. A file contains titled and ordered elements. The file itself does not need to be titled nor ordered. A pile contains items (documents) which may or may not be titled or ordered. The pile itself has no title and has no order.

	Elements titled	Elements ordered	Groups titled	Groups ordered
Files	Yes	Yes	?	?
Piles	?	No	No	?

Table 2.1: Piles and files

In 2002 Sellen and Harper [50] introduced the terms *warm*, *hot* and *cold* files. They used the terms warm and hot for documents which are used very often, like documents which are brought to meetings, have writings on them or have been flicked through. Cold files are files about finished projects which are not commonly used anymore. Thinking about Malone's findings, a link can be seen between warm and hot documents and piles and between cold documents and files. Documents which are not used often anymore are classified, put away and are made easy to re-find when they should be needed in the future. Warm and hot documents are kept close to the user because they could need them at any time.

As mentioned before, Malone stated that there are two major units of office arrangement being files and piles. In 2004, Trullemans and Signer [52] introduced a third major office arrangement, called *mixing*. According to Trullemans and Signer*filing* and *piling* do not suffice to describe the way people organise their documents in the office workspace. They stated that far to often a combination of filing and piling is being used, which they described as mixing.

Mixing was described as: to insert some kind of order in otherwise unordered documents. Some items just can not be classified as files or piles. Examples of these are ring binders with binder dividers or labelled letter trays, as illustrated in Figure 2.1. In between these labels these items can function as a pile with no specific order. But the presence of the labels makes sure there is some kind of reference to re-find items, as there would be in folders.

Figure 2.1: Structures which are neither files nor piles

After introducing the use of mixing Trullemans and Signer did research how often each of these organisational units were used. The conclusion was that mixing is one of the most used office arrangements. Most people do use it moderately in their office arrangement. Besides the use of the these organisational structures, research was done on how helpful each of them were in re-finding objects. Here mixing was mostly categorized as very easy to re-find objects, while files were easy to moderate and piles were more categorised between easy, moderate and hard to re-find objects in.

2.2 Personal Information Management

The research described in the previous section was necessary to be able to create the field of Personal Information Management (PIM). PIM was first seen as an application helping people to keep track of their personal information like e-mails, digital documents, calendar items, bookmarks or addresses. Examples of these systems are Personal Information Dashboard [2], BumpTop [1] or Haystack [29] [30]. An overview of this can be seen in Figure 2.2. The problem of keeping all this information organised in an easy to use application became so big that the term PIM was not only used for a certain kind of applications but has grown to an entire field of study [26].

2.2.1 From Tools to a Research Field

The field of PIM is interested in how you can organise your personal information in a way that makes it easy to re-find this information when needed. This can be physical documents as well as digital information. Before the term PIM was introduced, some research had been done in changing how the interaction with computers were done. In 1993 Mander et al. [41] invented the 'pile' metaphor on the computer. Acting upon the knowledge that folders

in the office setting are being used for more permanent and formal archiving and piles are used for storing documents temporarily, knowing they are going to be needed again soon, Mander took the idea of the pile and translated it to a digital equivalent. The idea followed the earlier explained theory that it is often easier for people to find something they used recently in a pile than in a folder.

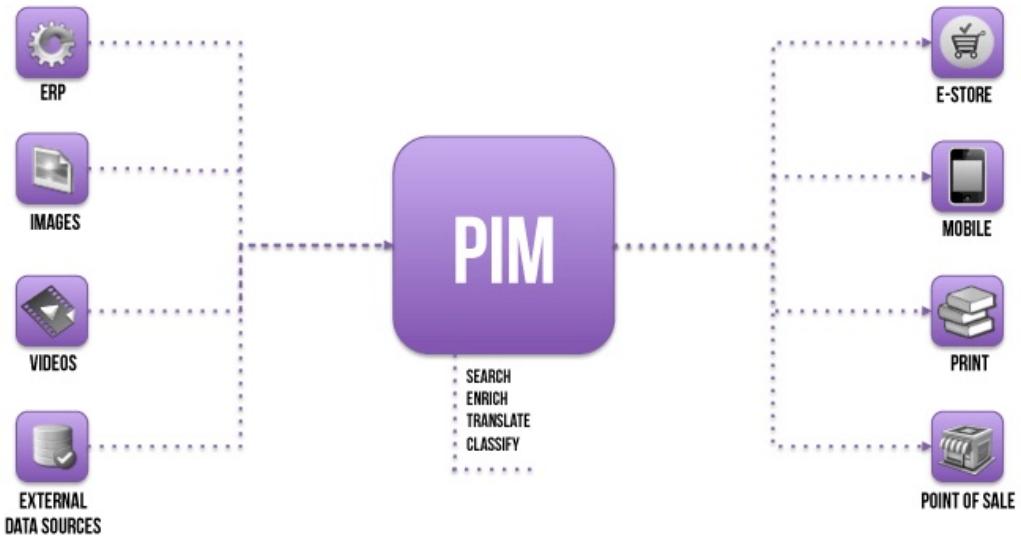


Figure 2.2: PIM overview

In 1995 Barreau [6] presented a conceptual framework that conveys the complex, high-level nature of PIM. She divided PIM into four sub-activities: (1) acquisition of items to form a collection, (2) organisation of items, (3) maintenance of the collection and (4) retrieval of items for reuse.

After the emergence of a lot of PIM tools, even more research has been done about what tools are the best to use in certain circumstances. For e-mail there was research done by Whittaker and Sidner [57] addressing e-mail overload and solutions.

2.3 PIM in Action

Of course, the research in the PIM field has resulted in some interesting PIM applications. One of the best known PIM applications is MyLifeBits [13, 14, 15, 16], of which a screenshot is shown in Figure 2.3. MyLifeBits has been developed at Microsoft by Jim Gemmell and his team. The original goal of MyLifeBits was to fulfil the Memex vision of Vannevar Bush, described in

1945 [8]. This goal can be seen as storing all of one's digital media. This can be documents, images, sounds and videos. The creators built MyLifeBits on four principles: (1) collections and search must replace hierarchy for organisation (2) many visualisations should be supported (3) annotations are critical to non-text media and must be made easy, and (4) authoring should be via transclusion.

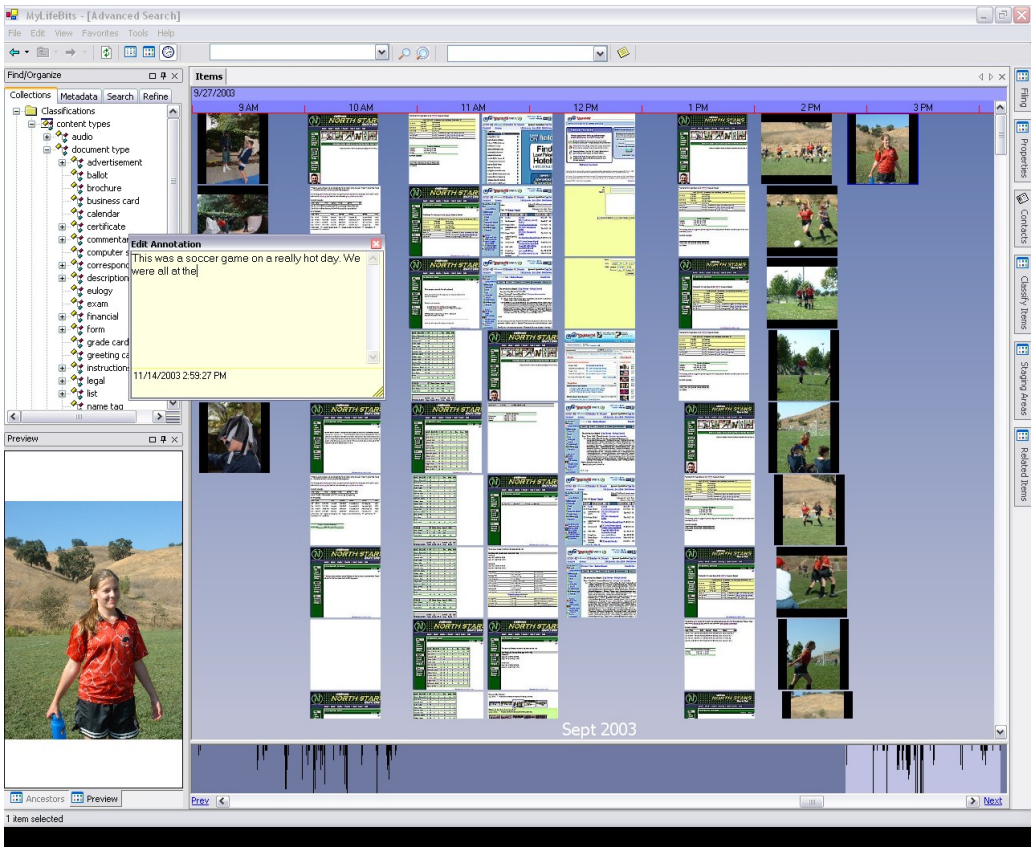


Figure 2.3: Screenshot of MyLifeBits

2.4 How Can You Track Your Documents

Keeping track of your physical documents in an office environment has been a part of the PIM research for quite some time. Keeping track of physical documents can be done in a various number of ways. Most of these approaches need some psychical enhancements to the documents or a pre-made database of all documents. Recently some research has been done about tracking documents without any enhancements to the documents themselves, as will be explained later in this chapter.

2.4.1 Tracking Documents with Enhancements

Tracking documents can be done in a lot different ways. A few aspects need be taken into consideration and based on what seems the most important for the user, some decisions need to be taken. The most important aspects are the ease of use and the correctness and error handling. Next to these, there are some less important aspects like speed, cost or reliability.

In this section we will discuss how to track documents using extra hardware. The main advantage of adding hardware to documents, is that they have a very low chance on errors. When a barcode or an RFID code is scanned one can be almost sure this code contains no errors and surely represents the document that was requested.

Radio-Frequency Identification

Radio-frequency identification (RFID) is the use of electromagnetic fields to transfer data. RFID is based upon the idea of identification based upon radio signals. This was first used in World War II where planes emitted a radio frequency. Based on the frequency a plane emitted, one could know if this was an allied or enemy plane. In the early sixties, two employees of Philips discovered how computer chips could be read from a distance. Based upon this finding, the RFID chip as it is known today was developed. The modern RFID chips were made to contain some data which could be used later on.

In practise this data will most often be an identifier which can be linked to a digital object. RFID tags are widely used in loads of object and beings [10]. This can go from clothes in stores to dogs or more recently even humans which use the RFID tag implemented into their hands to open locks or start cars. There are some known problems with RFID of which the most important ones are privacy and security [35] [12] [4] [27].

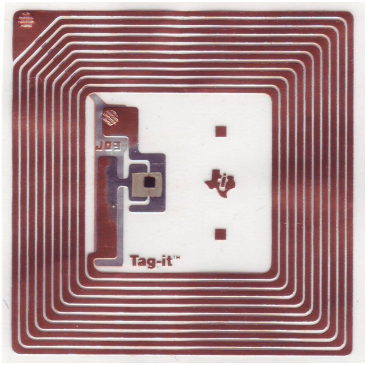


Figure 2.4: An RFID tag

In the context of document tracking based on RFID tags, each object that needs to be tracked will be equipped with an RFID tag as illustrated in Figure 2.4. This tag will link to a document on the computer. Whenever the tag of a certain document has been scanned, the computer knows which file or documents has been seen. This causes an overhead due to the fact that every RFID tag that gets placed upon a document needs to be digitally linked.

Figure 2.5: A paper tray with an RFID reader

The advantage of RFID is that there exist readers in various sizes and weights, and RFIDs in forms that can be placed upon almost everything. For the office environment, companies like Motorola and Sata Vicinity have developed entire platforms for document tracking based RFID technology. In an office environment RFID readers can be placed in ceilings scanning all documents leaving a room, under desks scanning all documents placed upon a desk, in walls or under cupboards. Even a paper tray can be equipped with an RFID reader, as shown in Figure 2.5.

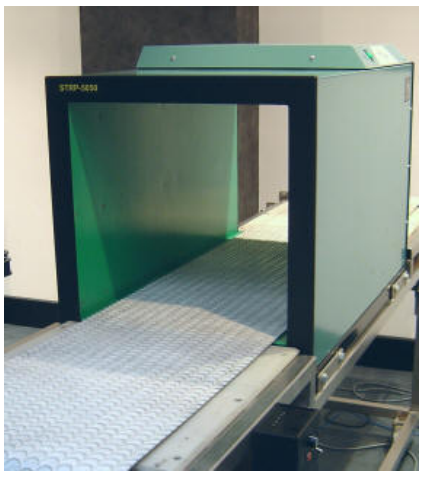


Figure 2.6: A tunnel for reading RFID tags

To keep documents in storage, documents will be stored in boxes, so entire boxes filled with RFID tagged documents need to be read. Special machines creating a tunnel reading all tags in a box have been developed to make this possible. An example of this can be seen in Figure 2.6.

Fitting an entire office and all documents within the office with RFID tags opens the opportunity to know which reader has seen which tag at which what moment. If a document needs to be found, one can look up the document on the computer and this will return the last RFID scanner that has scanned this particular code. When every employee of the firm wears a badge which is also equipped with an RFID code, there is the possibility to track which person passed that scanner with the documents in case.

Figure 2.7: A mobile RFID scanner

When a document is needed and the PC returns it was last seen entering the archive which is filled with rows of filing cabinets foreseen with multiple shelfs housing files, we are still looking for a needle in a haystack. To conquer this problem, some manufacturers created wearable RFID scanners as illustrated in Figure 2.7

The mobile scanners have the same usability as a Geiger counter. The device makes a beeping sound and based on the range between the tag and the reader the sound changes. This way the person looking for a specific document can hoover over the cabinets and based on the sound the device makes the user knows whether they get closer to the documents they are looking for.

The combination of knowing in which room the document is placed and the use of the mobile scanner makes it fairly easy to re-find a document. Currently there are already some system making use of RFID to re-find documents. This can be in conventional ways [3] or more exotic like printing the tags using e-ink technology [18]. The Smart Shelf system designed by

Decker can locate documents depending on the position of the document above an array of RFID antennas [9]. Hinske built a system that makes use of a moving arm under the table that changes the location of the RFID reader [19] [20]

Barcodes

The advantage of barcodes is that they are easier to apply than RFID tags and they are also cheaper. While RFID tags need to be applied and put on every sheet of paper that one wants to track, for barcodes one can install an extra printer API which automatically adds a barcode to every printed document. This takes away some of the overhead. However when previously printed documents need to be tracked, a barcode needs to put on them. Also the same system is needed when not only documents need to be tracked but also folders. These folders need to get barcodes put on them.

Figure 2.8: Barcode

Just like with RFID, every time a new barcode is used, a link between the barcode and the document needs to be established. To do so, a special piece of software is needed where all this information can be kept. When the user would employ a printer adding this barcode the software could automatically create a link between the physical and the digital document.

Figure 2.9: QR code

Recently, the use of the old-fashioned barcode, shown in Figure 2.8 is no longer popular. They have been replaced by 2-dimensional codes like the QR code which can be seen in Figure 2.9. The advantage of a QR code over a barcode is that the QR code can carry over a hundred times the amount of information a conventional barcode would carry. Because the QR code is two-dimensional it can be read from every side, which gives the extra advantage that the reader and code do not need to be aligned before the code can be read.

The biggest disadvantage of using barcodes is that they need line of sight for a few seconds to be scanned. This means that documents need to be held still under the barcode scanner for a few seconds, depending how good the scanner is. As the code needs to be visually scanned, every documents needs to be seen. This means that a pile with a large number of documents needs to pass the scanner all by its own, while RFID technology can read multiple files with are placed upon each other.

Fuducial Markers

Fiducial markers or fiducials are objects placed in a room that can be tracked by computer. These fiducial markers are used as markers in the room and can be used for reference or measurements. Fiducial markers exist in loads of forms and can be made unique for identification. An example of a fiducial marker is shown in Figure 2.10.

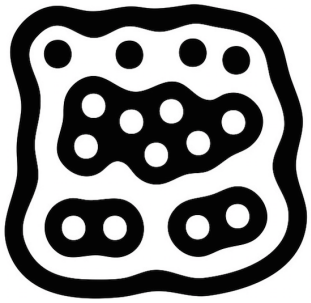


Figure 2.10: Fiducial marker

Fiducial markers are already widely used apart from Computer Science. There are applications used in sniper guns where fiducial markers are used to determine the distance to a target. Another example can be found in printers where ink cartridges are equipped with fiducial markers so the cyan, magenta, yellow and black printing plates can be aligned correctly.

Figure 2.11: An ArUco marker

One of the best known libraries for Fiducial markers used in Computer Science is ArUco. An example of an ArUco marker can be seen in Figure 2.11. ArUco markers are 7 by 7 squares where the borders are black. Inside the borders the 5 by 5 square represents a unique code. This means the code consists of 5 words of 5 bits each. This code is a slight modification of the Hamming Code. Only 2 of the 5 bits contain information, the other 3 are used for error detection. This results that only 1024 (2^{10}) unique codes can be made.

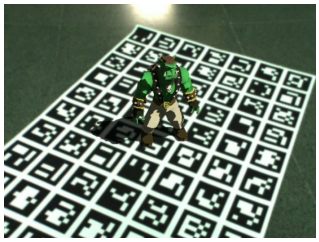


Figure 2.12: The ArUco ogre

The best-known examples of the use of ArUco is the ArUco Ogre that can be seen in Figure 2.12. This is a digital Ogre that, based on the fiducial markers that were filmed, can virtually run around a real room. The use of the fiducial makers lets the computer know how the ogre can run in the room.

Bluetooth

Bluetooth is well-known to be used as a communication channel between devices. But next to transferring data, Bluetooth can also be used for the indoor localisation of devices. This can be done by using Received Signal Strength (RSS) of other Bluetooth devices relative to the ones we want to locate.

Johnson and Seeling [25] used the RSS technique to create a working prototype of indoor locating devices using Bluetooth. To be able to locate the devices the fixed markers (beacons) make use of their discoverability. Each beacon his name is in a well known format containing certain information about the location. It starts with the **:geo:** tag and is followed by the longitude and latitude of the device. This way, the device only need to discover each other to know all information needed, and it is not needed to pair the devices to be able to locate the device.

This localisation has some flaws. Devices can be localised with an approximate accuracy of 1.83m with a standard deviation of 1.41m.

Bargh [5] stated that the use of RSS, or Bluetooth RSS, specified by RSS Indicator (RSSI) and Link Quality(LQ) is not reliable enough, because of the heterogeneity of Bluetooth hardware in devices. Also the fact that with the method Johnson uses your devices always need to be discoverable, which tend to be known as unsafe he created a new way to use Bluetooth for localisation. Bargh his location determination method is based on a fingerprint-based localization solution that relies on the Response Rate (RR) of Bluetooth inquiries.

There are some practical devices available on the market that make use of this system [46]. The best-known example is iBeacon. iBeacon is an indoor positioning system described by Apple Inc. These devices are used to determine the range of a phone in comparison to a token. This way, the nearest token can be determined and based on this information actions can be taken. It can be that it is determined in what room the phone is present and based on this information some home automation can be triggered. In some stores items are fitted with iBeacon devices which can trigger pop ups on a phone giving extra information about the product. Yang even developed a positioning system for hospitals using iBeacons [58].

Bluetooth could be added to folders in the office environment so a smartphone could be used to re-find the documents. The precision of the Bluetooth range is not fine grained enough to give decent results when a lot of beacons are placed close to each other.

Near Field Communication

Near Field Communication (NFC) is a wireless technology based on RFID. Like RFID, it is designed to transmit small amounts of data over short distances. NFC is currently widely used for payment methods [48] [47]. Big companies like Vodafone, China Telecom an Google Wallet make use of NFC to be able to do some payments using the mobile phone. The credit card of a user gets connected to the phone and then can be used to pay at places

that accept NFC payment. Countries like Germany, Austria, Finland, New Zealand, Italy, Iran and Turkey have tried NFC ticketing systems for public transportation. A partnership between Google and the Equity bank in Kenya has introduced NFC payment systems for public transportation in Nairobi.

Using NFC for document tracking will generate the same advantages and disadvantage as RFID does. This with an extra disadvantage that NFC is designed to only work over a short distance. As with RFID one of the concerns with NFC is the security and privacy of the system [39] [55] [11].

2.4.2 Tracking Your Documents Without Physical Enhancements

There are already some possibilities to track documents without making use of physical enhancements. A system has been implemented by Kim using video [33]. In these kinds of systems the identification of documents can still be done in various ways. One example is making use of the SIFT algorithm [37] [38] [31] which will be explained more elaborately. SIFT is an algorithm that recognizes images using image features. Other examples of systems using image features to do recognizing are SURF [7] [7], a tool by Lindeberg [36], a tool by Mikolajczyk [44] and a tool by Tuytelaars and Van Gool [53] [54]

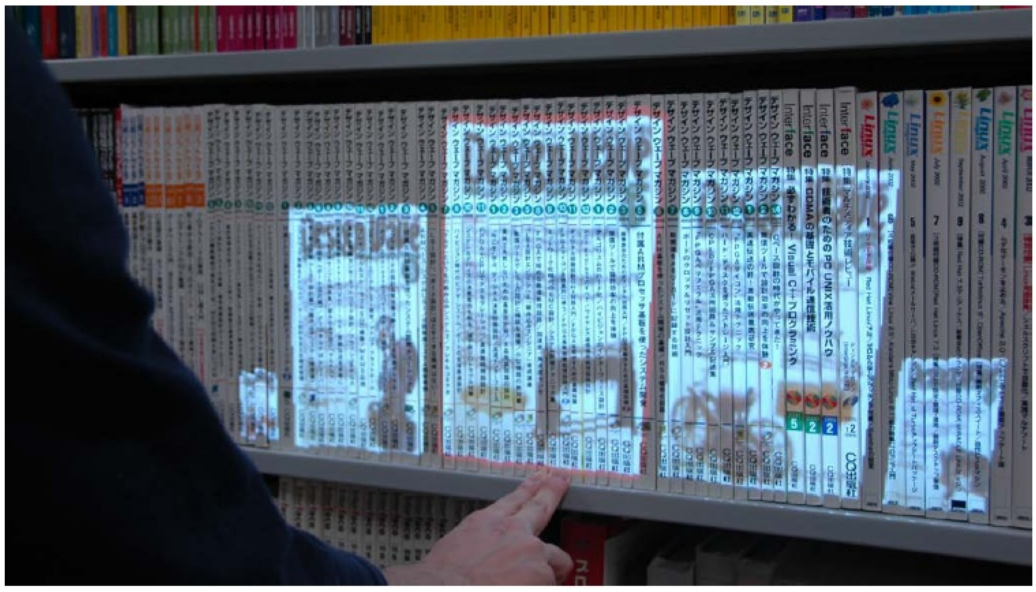


Figure 2.13: Matsushita's bookcase

2.5 Tracking the Office Environment

Tracking and recognising a document is useful, even though that documents are often placed in folders on shelfs or in a drawer. To keep track of these items, we need more than only the identification on documents. This could be done by providing folders with enhancements as well, but there has been done some research about how to do this without the need of these enhancements.

Matsushita [43] has done some research about how to track and find book sin a bookcase. Figure 2.13 shows the overview of the bookcase. When fingers are placed upon a certain point of the bookcase the covers of the books stored at that position of the bookcase are being projected. This way people do not have to pull out every book from the shelf to see its cover. When a user wants to place a book on to a shelf the user has to hold the book in front of the shelf and a line will show where the book should be put on the shelf.

Figure 2.14 shows an overview of the set-up used by Matsushita. On top a projector is used to display all information on the bookcase. On the right, a camera has been placed. This camera captures the presence of books and will extract the cover of the book placed upon the shelf. This way it can search for additional information on the internet and fit the book in with the other books already placed upon the shelf. A 1-D range sensor has been used to track where the user his hand is placed.

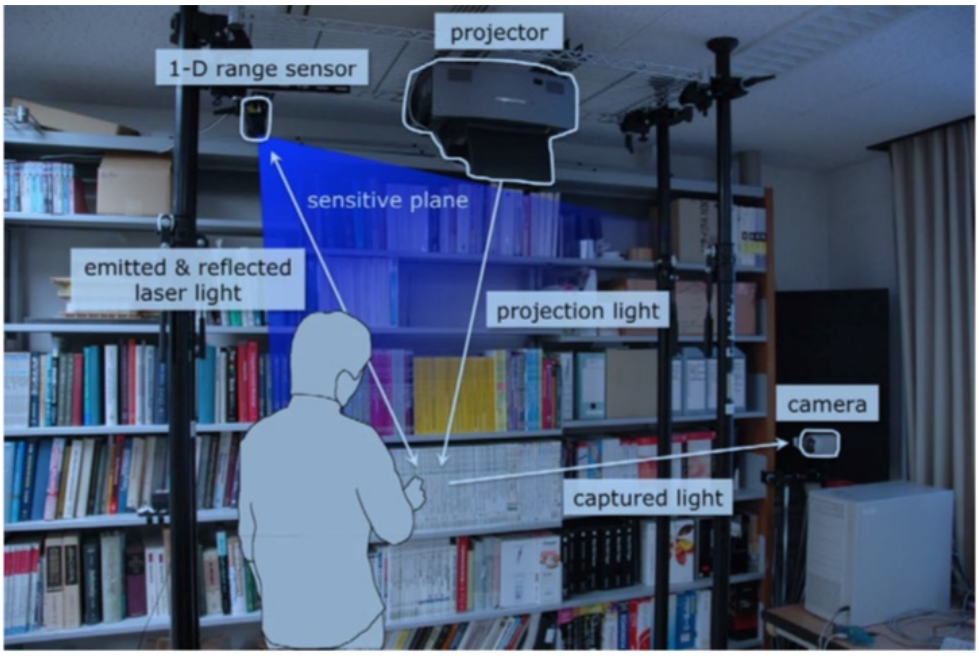


Figure 2.14: Matsushita's setup

Matthew Jervis from the University of Waikato, Hamilton, New Zealand is one of the only researchers who did research in using PIM with support for organisational structures. He built various systems that provided special attention to in which structure documents are being stored. Through the years Jervis developed multiple systems with their own advantages and disadvantages. Logically the focus of a newer version was to conquer the disadvantages of the previous version. The fist system that Jervis build is called "Smart Filing System" [49]. The goal of this system was to combine the advantages of both digital and physical representations of documents. This system made use of augmented folders in a filing cabinet. The filing cabinet, invented by Seibls [17] in 1898, already exists for a long time. It has been proven to be good to store documents but it has some flaws for re-finding documents. A more elaborate system of Jervis tackling the problems with the filing cabinet is called SOPHYA or "Smart Organisation of PHYsical Artefacts" [21]. The focus of SOPHYA is to fix the slowness of the hardware by placing a middleware between tangible interfaces and the client. SOPHYA fixed a lot of problems but there still the issues that documents could not be linked to a specific location. This because the system sees documents as containers and the system does not track to order of the containers. They are linked to a bus system that does not allow exact location detection. To fix the problems in SOPHYA, SOPHYA V2.0 has been built [22] [23] [24].

2.6 Summary

In this chapter some related work is investigated. Firs,t the way people store their documents was studied and later one some ways to keep track of physical documents were investigated. The ways to track physical documents was divided in two parts. The first part was with adding some extra hardware or tags to the document. Examples of these are RFID tags, fiducial markers or barcodes. These are trustworthy for tracking, because they have a low chance on errors. On the other hand, they ask some extra effort of the user to place this hardware on the documents. A second way physical documents can be tracked is without the use of extra hardware or tags. An example of this is SIFT. Having this kind of tracking, the user interaction is less important but this kind of tracking comes with less reliable results. Chances are real that documents return mismatches using this kind of systems. Both the tracking with and without tags will be used later on to do tracking. Depending on the situation, the most optimal way of tracking can be chosen.

To conclude this chapter an overview of how office environments are being tracked today is given. Examples of this are the bookshelf of Matsushita and

the office of Jervis. Both have their advantages and disadvantages. Based on the disadvantages of these systems the requirements for this framework can be determined.

To work with the hands or brain, according to our requirements and our capacities, to do that which lies before us to do, is more honourable than rank and title.

Albert Pike

3 Requirements

In this chapter we explain what requirements are needed to be met in order to tackle the problems mentioned in the introduction. These requirements are based on the principle that it is opportune to track as much as possible about physical documents in the document environment, with as goal that it is possible to use this information later on. This information can be used to re-find or explore the documents placed in an office environment.

We explain each requirement needed to achieve the goal of having an automated office environment that keeps track of the documents in it. These requirements can be deducted form the introductory chapter in which the problem statement was explained. Additionally to the requirements that need to be met to conquer the problems explained in the problem statement some extra requirements can be added based on the need to keep our solution user friendly.

3.1 Automation

The goal of every PIM system is to help users to keep track of their personal information. It should help them to re-find their documents both in a digital way and in a physical way. The major benefits of the PIM system is that when a user tracks their documents they can re-find them in an easy way or have the possibility to explore the documents.

The user wants to be aided to re-find their documents when needed. Even though, an important consideration to keep in mind is that the overhead to have this possibility cannot outweigh the burden it takes the user to re-find documents without assistance. The time that the user has to invest to keep the tracking up to date needs to be significantly less than the time the user would need to manually re-find their documents in the office environment. To lower this time, it is of essence to try to automate the tracking as much as possible. This includes that the user does not need to insert every document manually in the computer and does not have to update its location manually. This can be by automated using the tracking techniques discussed in the related work in Section 2.4. Even though these tracking techniques can be heavy duty on the user. Using RFID tags on documents would include that every document needs to be equipped with a sticker containing an RFID tag. Applying this kind of sticker on every document is an overhead that could scare the user to use this solution. This does not exclude the use of RFID tags in the office environment. This kind of tags can still be used on document folders of which there are far less present in an office and which are not as often added to the office environment as documents.

To keep the tracking automated, the user input needs to be kept to a minimum. The minimal user input can be achieved by tracking the environment by making use of hardware to track certain activities in the office. The minimal user input does not exclude the use of tags or other tracking mechanisms. Tracking equipment like barcodes, QR codes or TUIO codes should be limited to a minimum and when these are used, the overhead they bring along should be kept to a minimum. This can be achieved by implementing a special printer that provides these codes automatically on printed documents instead of relying on the user to insert this data about documents. For the reading of this kind of tags it is recommended that the users do not need to scan these codes manually. This can be achieved by predicting when and where it will be useful to read this tag to keep track of the document. On the places where the document should be tracked it can be opportune to fit certain mechanisms that read this kind of tags whenever they have been seen. These mechanism are ideally fully automated, so no interaction with the user is needed. However, also a minimal interaction like pressing a button next to the storing location can still be acceptable. A minimal interaction with the user can be accepted in contrast to compel the user to insert information like the title of the document, an id of a document or let the user scan every document that needs to be stored. This kind of overhead is acceptable when using containers, but not for every single document. Even though, even on the containers the overhead should be kept to a minimum.

3.2 Organisational Strategy

When documents are being stored in the system it is essential that as much information as possible is being kept. An important part of the research done in this thesis is to keep track of how the physical documents are being kept. This information can be useful for other projects that build on the information that has been tracked. When user interfaces are being developed, it is essential that the creator of the user interface can give a representation of the stored data that is closely related with the way the documents are stored in the physical space.

In this thesis we will make use of certain organisational strategies in which documents can be kept. For every physical document we will store the organisational strategy containing the document. The organisational strategies that will be used in this project are *files*, *piles* and *mixtures*. Files and piles were already introduced by Malone in 1983. In 2004 Trullemans and Signer added the invention of mixtures to these. An elaborated explanation about the difference between files, piles and mixtures can be found in the previous chapter. A short overview is seen in Table 3.1.

	Elements titled	Elements ordered	Groups titled	Groups ordered
Files	Yes	Yes	?	?
Piles	?	No	No	?
Mixtures	?	?	?	?

Table 3.1: Files and piles

In this table the major differences between files, piles and mixtures are shown. All elements contained in a file must have a title and must be ordered. A group of files can be ordered but this is not a requirement. Just as they can have a title. Piles on the other hand cannot have any order and a group of piles cannot be titled. Individual items within the pile can have a title, but this is no requirement. Groups of piles can be ordered, but this is not necessary. The third kind of organisational structure can be seen as a combination of the two organisational structures explained above. This can be seen as a pile with subdivisions providing labels or a file which has no order. As an example one can imagine paper trays. Paper trays are labelled, but there is no clear order in the trays. The content of a paper tray can be seen as a pile.

Because there is data kept about the way documents are being stored, this data can be used again whenever other project want to visualise the documents that have been stored. When a pile needs to be displayed in

a user interface on a computer the creator of the interface can take into account that a pile has no order and title. While piles have no order, a file has an order. For mixtures this could be determined on the instance of the mixture. A clear example can be seen when documents need to be re-found. The re-finding of documents relies on certain cues. These cues are a *Time Cue*, *Spatial Cue* and *Context Cue*. Each of these cues have a relation to an organisational strategy, as illustrated in Table 3.2. An organisational strategy has a time, spatial and context cue. This means that a file structure could be ordered based on the last time it was accessed, alternatively the location of a file could be used for re-finding or the context in which a file has been used could be used. For piles the spatial cue could be used because the location of a pile could have a meaning for the user. The time on the other hand within a pile provides no information. For a mixture the location of a mixture is of no essence, but the last time a mixture has been accessed by the user (time) could be relevant to re-find documents.

	Context Cue	Spatial Cue	Time Cue
Filing	x	x	x
Piling	x	x	
Mixing	x		

Table 3.2: Each organisational strategy supports certain cues

When a new file, pile or mixture will be made in the physical world, a digital representation of this needs to be made in the digital space. For files it is necessary to include a label on the file. The need for a label means that the user needs to insert the addition of an extra file using a user interface in the digital space. Piles have no label. This means that piles can be spotted by the software without that the user needs to insert the creation of a new pile. For mixtures it will depend on the kind of mixture. Because some mixtures have labels it will be opportune to create an interface for new mixtures that are being added. In some cases there is the possibility that no label is needed and the new creation of a mixture can be done automatically. This will depend on the setting in which the mixture is used.

3.3 Lifelines

Extra information about what happened to the physical document needs to be stored. Most information that needs to be stored is related to time. Storing all places a document has been seen or all moments a document has been

accessed needs to be stored together with a timestamp when this happened. To store this kind of information lifelines are used. Each document can have multiple lifelines tracking events that happened to the document.

One lifeline can keep all information about the places the document has been stored. Another lifeline can keep other events that happened to the document like when the document has been accessed, created, removed or any other interaction. Storing every place where a document ever has been stored will have some use when queries need to be done on this data. When a user wants to know what the *default* storage place of a document is, a query can been done over all the places a document has been kept. Combined with the timestamps when a document has been placed on a certain place it can be calculated how long a document was located at a certain place. The place the document has been stored more often or the longest can be suggested as the default storage place for a document.

A lifeline that stores the times a document has been accessed can provide data when a user wants to re-find documents accessed within a certain timeframe. Providing an interface to the user where the user can choose a certain timeframe in which they want to know which documents they accessed most often they can use the time a document was noted as accessed in a the lifeline. Besides storing whenever a document has been accessed, other extra information can be kept in the in same lifeline. This information can be general or very specific for the setting in which it is used. Extra entries in the lifeline can be things like *addToPile*, *removeFromPile*, *addToMixture*, *removeFromMixture*, *addToPile* or *removeFromPile*.

3.4 Context

When a user is working with a document it can be assumed that the user does this within a certain context. This context is mainly time based. On certain moments a user is active within a certain context. This context can differ on the task the user has at hand on a certain moment. One document can be used within multiple contexts, but the context of a user will not change quickly. When a user starts working on a particular task they can use a lot of different documents to complete this task, but this task can be seen as the context in which these documents are being used.

When this context can be added to a document every time it is being used, it opens the possibility to track the context in which a user has worked. This context can be used in a wide variety of cases. For example, the context can be used to do some data analysis to see on what task a user has spent most of their time or which task took less time than expected. In the context of

Personal Information Management (PIM), this context can also be used for exploring and re-finding documents.

Based upon a context, a user can re-find documents that were being used within this context. This can help a user re-find all documents that are interesting concerning a certain context. When a user wants to restart working on a project they can do a search based on the context of this project and find all relevant files.

To enable these possibilities, the context of each document needs to be tracked. To do so, the current context will be extracted from the Context Modelling Tool (CMT) developed at the WISE lab of the VUB. This tool provides the current context a user is working in at a given time. Every time an action with a document is tracked and stored within a lifeline, the current context can be added to this lifeline. Based on all contexts stored of a document in the lifelines the most important contexts for a document can be tracked. This enables the user to find all the relevant contexts for a certain document, but also gives the possibility to find the most important context of a document.

A document can be relevant in a lot of contexts. But certain contexts are more relevant than other contexts. A clear distinction between important en less important contexts is needed to be able to give relevant information to the user. To know which context is more relevant than another context, it is opportune to add a weight to the instance of a context. This means that the relevance of the context in which a document has been created is more important than the context that gets added when a document is just accessed briefly. For example, when a document is created with current context A, and later is accessed in context B, the weight of context A is higher than the weight of context B.

3.5 Extensibility

An important requirement for this framework is the extensibility. This extensibility comes to mind on multiple locations throughout the system. First of all, the framework itself needs to be accessible for the user. This means that the user can interact with the framework in any way they want, with all access provided. This will be so for the PIM system that will be used as for the extension that will be implemented specially for the tracking of documents. Even other parts of the framework that are implemented focussing on re-finding documents, or any other goal, the extensibility is a main objective. The extensibility is important so the users can create new applications using the framework. The framework needs to be a base component that can grow

to something interesting for every user, with loads of possibilities. The main philosophy behind this is that neither two offices neither two users are the same. They each have their own unique characteristics. Users have certain preferences which are quite unique. Because it is impossible to provide a system that suits every user and every office in all their needs, it is crucial to make an extensible framework, that the user can extend to meet all their requirements.

A second reason to make it as extensible as possible is because the users should be encouraged to extend what has been developed. This project will provide a basis to track physical documents and provide endpoints to do this tracking. Some plug-ins to do the tracking will be provided. But these will not cover every sort of tracking that needs to be done. The implemented plug-ins will function as a proof of concept. The main idea is that the users will implement their own plug-ins covering their needs as good as possible. These plug-ins can be kept for personal use but can also be provided for every user of the system. Additionally, the plug-ins consist out of two parts for doing the tracking. One part will try to recognise the document and provide it with an URI that can be kept in the framework. A second part will compare all URIs of a certain tracking technique to determine whether or not this document was already tracked by the system. This means that every recogniser has a certain comparator, but nothing says that a recogniser or comparator cannot be reused. For example, when a user wants to keep using an recogniser but thinks they can build a faster comparator, they should get the possibility to only supply a new comparator using the old identifier, or more important, that multiple identifiers use the same comparator. This could be the case when two or more identifiers return a single id for a document. The comparison of ids is for every identifier that returns an id is the same, so for every identifier that returns an id the same comparator could be used.

4
KOR Framework

To fulfil all requirements presented earlier we implemented called the Keeping-Organising-Re-finding, or KOR framework. This system has an underlying PIM system to store all information about the documents and has been extended with extra capabilities to fulfil other requirements. It consists of three parts, one being the extended version of the PIM system, a tracking component which makes it able to track documents using parts of the extended PIM system and a re-finding component which is not discussed in this dissertation.

4.1 Architecture

The well-accepted theoretical PIM framework Keeping, Organising and Re-finding (KOR) by Jones is the basis for this project. The goal is to create a framework that provides Keeping, Organising and Re-finding in a usable way. The goal of this framework is to provide a working entity which users can operate to keep track of their physical documents, but also provide an easy way to extend the framework.

The architecture of the Keeping-Organising-Re-finding (KOR) framework consists of three major components: Tracking (Keeping), ReViTa (Organising) and Re-finding as illustrated in Figure 4.1. All these components stand on the same level while only ReViTa communicates with the underlying PIM system. On top of the KOR framework we built a Tracking and a Re-finding

component. These components communicate with their respectable counterparts in the KOR framework. This means that the framework can be seen as some sort of middleware between the Tracking and Re-finding components and the PIM system.

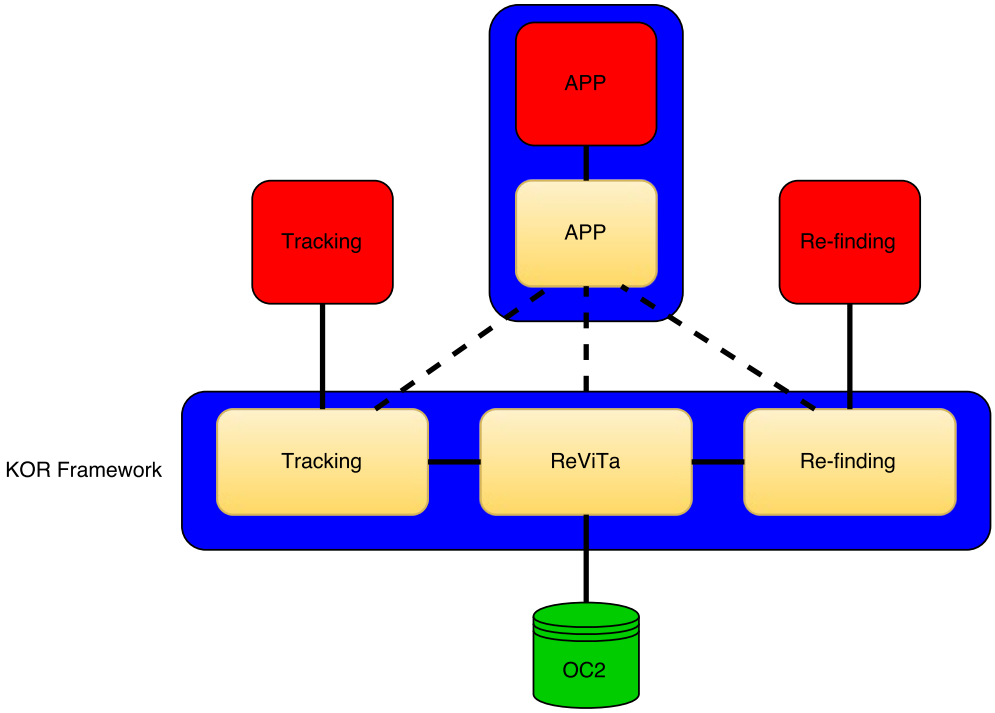


Figure 4.1: KOR Architecture

When taking a closer look at the architecture illustrated in Figure 4.1, it can be noted that the only communication done between the framework and the PIM system (OC2) goes through ReViTa. One can conclude that only having ReViTa accessible for users can suffice to make use of the power of the PIM system. The tracking and re-finding components are added to create a framework to make it easier for the user to do the core principles of KOR, being Keeping, Organising and Re-finding. Next to this, a user might want to add some extra functionality, and create his or her own application using these principles. The architecture shows how this can be done. The application could exist out of a practical layer and a smaller middleware which translates the output of the KOR framework to something the application can handle. The connection between the application and the parts of the KOR framework are shown in dotted lines because the architect of the application can choose which endpoints provided by the framework are being used for the application.

When a user wants to create a visualisation application, for instance, it can be interesting to use some of the functionality of ReViTa and the Re-finding component, but the user has no use for the data provided by the Tracking component. If the application would be created to fit extremely well on top of the KOR framework, the middleware part inside the application could be omitted, but yet, it can be seen as a better architecture when a layer between the core of the application and the KOR framework is made by use of a middleware.

4.2 Re-finding - Visualising - Tracking - Augmenting

The tracking module uses the ReViTa module, which on its turn is an extension of the Object-Concept-Context(OC2) model developed at the WISE lab of the VUB. Therefore, first the OC2 model is explained. Next, the extensions that have been made to the OC2 model to provide the ReViTa module are discussed. At last, the tracking model, which makes use of the ReViTa model are explained. All these parts together provide the tracking components of the KOR framework. Once the framework is created it is possible to provide it with tracking techniques to start tracking documents. This will be explained in the next chapter.

4.2.1 Object-Concept-Context Model

The entire tracking component is based upon previous work called the Object-Concept-Context Model (OC2). OC2 is a metamodel to keep track of objects. These objects can vary from digital documents to physical documents, pictures or other media. The goal of OC2 is to provide the user with documents stored in a database and useful links between these documents.

The OC2 conceptual framework consists out of three layers which are illustrated in Figure 4.2, being the object layer, the concept layer and the context layer. Objects stored in the object layer are physical and digital pieces of information, like physical documents, e-mails or photos. Each object is uniquely identified and can contain other objects, for example a folder containing documents. Objects can be linked to each other trough navigational links and structure links. Navigational links are links that represent a navigational relationship between objects. Structural links are links between object to express a structure, like a folder. Above the object layer there is the concept layer. The goal of the concept layer is to abstract the complexity of the real world, as an example one can image labels on a folder. To link

concepts there are associative links. A relation between a concept and an object is done by an extent links, imagine the digital representation of the label and the folder itself and its documents. Typically this can be seen as a categorisation of objects to a concept. The top layer of the model is the context layer. A context can be seen as the user their activity. Objects and concepts have a certain relevance to a context. This relevance is the context in which they have been used.

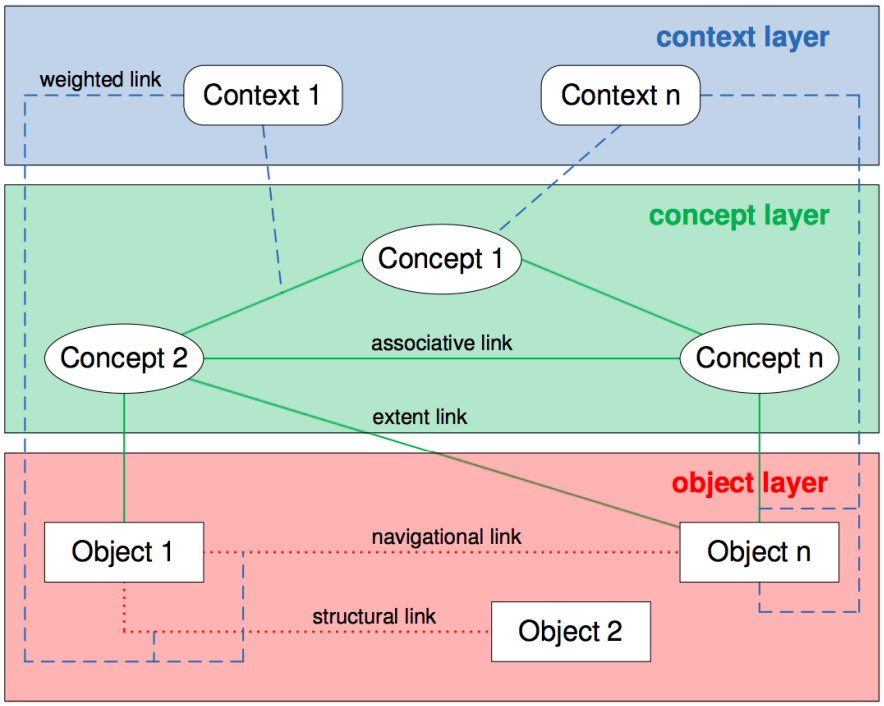


Figure 4.2: OC2 Conceptual model

The OC2 conceptual model has been translated in the OC2 metamodel, which can be seen in Figure 4.3. The OC2 metamodel is an extension of the Resource-Selector-Link (RSL) metamodel. The RSL model is a metamodel for cross-media information spaces [51]. In the centre of the metamodel are entities. An entity can be a resource, link or selector. Links represent the relationships between entities. Resources represent files or other information and selectors are used to define areas in a resource. By making them subclasses of the entity, individual pieces of resources can be linked. To support the OC2 metamodel the basic RLS model had to be extended to support the different kinds of links. Associative Links allow two concepts to be associated with each other. Extent Links allow a concept and object to be linked. Objects can represent both digital objects and physical objects.

When considering Figure 4.3, the difference between the core RSL model and the OC2 metamodel can be seen as the extensions made for the OC2 model are annotated in blue while the core RLS model is grey.

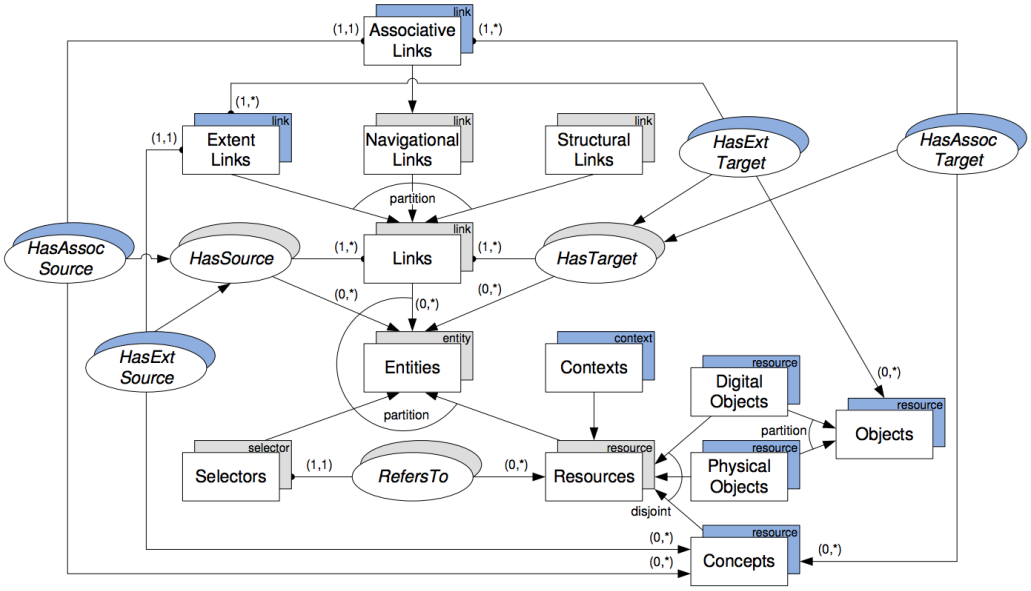


Figure 4.3: OC2 Metamodel

To be able to keep all extra information discussed above, some extensions will need to be made to the OC2 metamodel. There is need to add tracking data to objects and concepts in the OC2 model, there is need to add lifelines to the OC2 model and a location property needs to be added to the system to keep track of the location of the document. Next to this, extra data to store the concept of files, piles and mixtures need to be inserted in the model. To store the extra information about files, piles and mixtures, there is an extra superclass used, called *Organisational Structure*. An Organisational Structures is a Structure that has Structural Links. Via these Structural Links the Organisational Structure can be connected to the existing OC2 model.

In this extension, some extra features of the Organisational Structures can be included. As stated previously, each filing must have a label. To include this label in the model, a one-to-one relation between a label and a File can be included in the model. For a mixture this label should not be one-to-one because a Mixture can have a label but is not required to. Piles have no labels so no links need to be foreseen in this case.

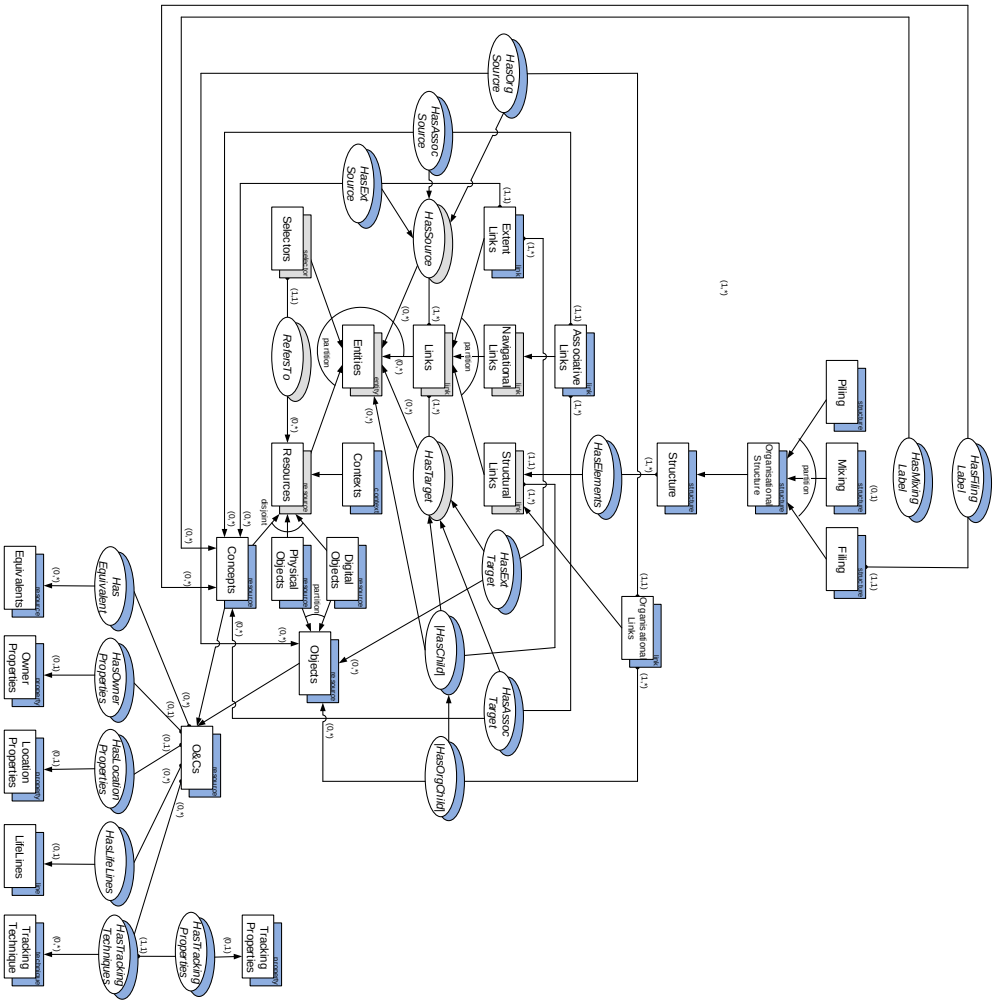


Figure 4.4: OC2 extended

4.3 Extensions of the PIM System

Starting from OC2, there are some modifications that need to be made. These modification have been explained in the previous chapter. Next to these modifications of OC2, a middleware needs to be placed between OC2 and the tracking devices. This piece of middleware will be the KOR framework which consists of three separate parts which all have their own functionality and provide their own endpoints to the users. The first component is ReViTa (Re-finding-Visualising-Tracking- Augmentation) which implements the extensions on the OC2 metamodel. The second part of the middleware

is the tracking component, which is the main subject of this thesis. A third component is a re-finding component. This component is part of an other Master's thesis and will give the user the possibility to re-find their documents tracked by the tracking component.

ReViTa (Re-finding, Visualising, Tracking and Augmenting) provides a RESTful interface between OC2 and the outside world. Next to this, it implements the extra functionality in OC2 to be able to store extra information. The design of OC2 created the possibility to not having to change the core of OC2. Every change needed in OC2 could be done by adding a mapping between the required extensions and the provided components in OC2. This means that the core database model of OC2 has not changed, and that ReViTa, next to providing a REST interface for some of the OC2 possibilities also provides the same interface for the extra requirements. An image of the new extended OC2 model is shown in Figure 4.4.

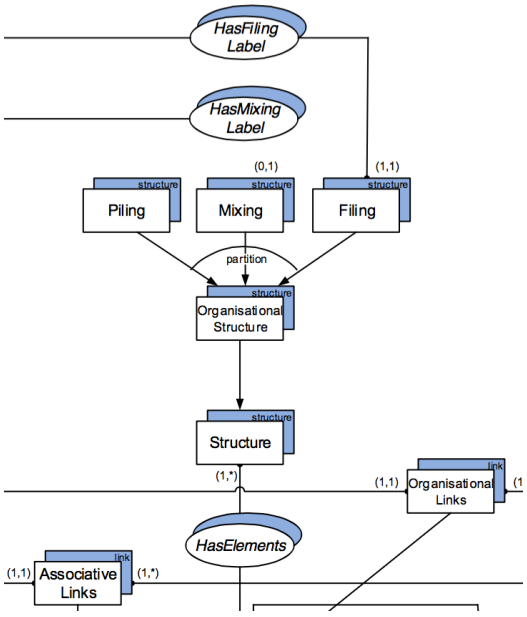


Figure 4.5: OC2 extensions for Organisation Structures

At the top of the extended framework the additional requirements for Organisational Structures can be found, as illustrated in Figure 4.5. Here it can be seen how Organisational Structures are added to the model. An Organisational Structure is the superclass of the Filing, Piling and Mixing classes. The Filing class has a one-to-one link with *HasFilingLabel* which in its turn is a concept. For the Mixing class this is not a one-to-one connection with *HasMixingLabel* because a mixture can have a label but is not required

to have a label. *HasMixingLabel* is also connected with a concept, which could represent the label.

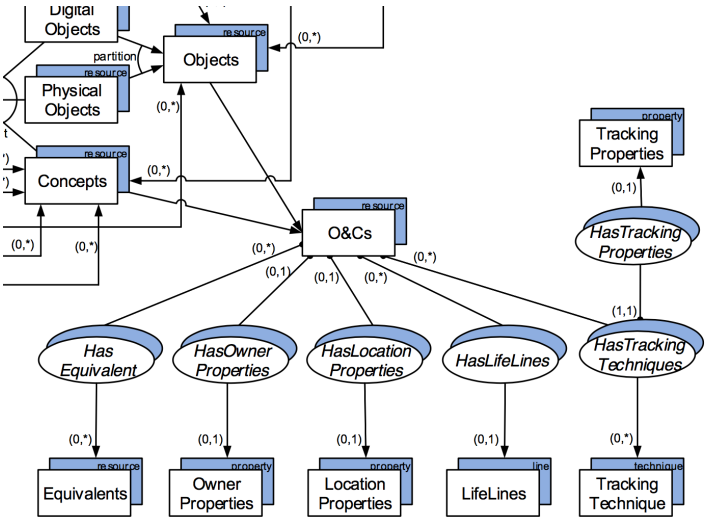


Figure 4.6: OC2 extended

At the bottom of the extended model it is shown that an Organisational Structure is a subclass of a structure, which on his turn has elements which are Structural Links from the original OC2 metamodel. The extensions that need to be made to the OC2 metamodel in order to keep LifeLines, Tracking Techniques, Locations, Owner Properties and Equivalents can be seen in Figure 4.6. In the original OC2 model there were already Concepts and Objects present. Objects is the superclass of Physical Objects and Digital Object. Concepts were already a widely used concept in the OC2 metamodel.

First, a superclass is made for Objects and Concepts. This is because both Object and Concepts need to be able to have some of the newly added properties. This superclass *O&C* (Objects and Concepts) is implemented to provide the extra needed properties for both Object and Concepts. In total there are 5 new properties required to be able to satisfy the requirements. These properties are added to the O&C class. The extra classes added to the OC2 model are: Equivalents, Owner Properties, Location Properties, LifeLines and Tracking Techniques.

- **Equivalents:** This contains equivalent objects or concepts to the current object or concept. This can occur when, for instance, a document has both a digital and physical version present in the system. Both the physical en digital version are of the same document so they are equivalent to each other. But because the other properties will differ for both

versions they can not be kept as the document in the system. Both the digital and physical representation will have different LifeLines and will have different Location properties.

- **Owner Properties:** In this class the properties of the owner can be kept. This is needed because multiple users can use the same database and the same system. Adding owner properties can help determining the owner of the document and can help when visualisations or augmentations need to be made for a certain user. This owner property also provides knowledge about which documents that are related are owned by which users, so that links can be found between documents and users. Even more a link can be drawn between the context of documents and the people working on them, what can illustrate the relation between the work two users.

- **Location Properties:** The Location Property class keeps the location of the document. In the context of this thesis this location property represents the physical location of the physical document. This can be a meaningful location like "Topmost shelf of the cupboard in the corner". In other cases this can be coordinates. When, for example, a pile is stored upon a desk the coordinates of one of the corners of the page can be saved in the locations property. These coordinates can on their turn be reused for re-finding documents within a pile. When the re-finding mechanism wants to put focus on a certain pile using an overhead projector it is needed for this projector to know the coordinates of one of the corners of the pile.

  The semantic meaning of the location of the document can on its turn be used when a user wants to re-find a document using a user interface that returns text or speech. If this user interface would return some coordinates in an absolute space this would be useless for the user. But, if the system would return "Topmost shelf of the cupboard in the corner" then the user has an idea where to look.

  Note that this property is not used to keep track of documents within organisational structures. When a document is stored within an organisational structure this is kept automatically in the system, there is no need to keep a separate property saying that a document is stored in a particular organisational structure.

- **LifeLines:** Every object or concept can have multiple lifelines. These lifelines can keep data about a document or organisational structure which can be placed on a timeline. For example, there can be a lifeline

which keeps all old locations of a document or organisational structure. The location that is being kept by the Location Properties class is the current location of a document or organisational structure. When the user wants to know the history of where a document or organisation structure has been kept, he can use lifelines storing all old locations of a document.

Other possible lifelines can be a lifeline with all interactions with an Object or Concept. When, for example, the lifeline, is added to a file, the lifeline can store whenever a document gets added to the file, or whenever the file gets accessed, without that there was a document added or removed from the file. Of course, a lifeline can keep information about how the file has been created. It can store a tag when the document is created, or when it has been removed from the system. This information can be useful to the user if they want to have a timespan in which a document or file has been created, frequently accessed and when it was deleted.

- **Tracking Techniques:** Where the previous additions to the OC2 metamodel were interesting to store some extra information about the document, the Tracking Techniques are the real core matter of this thesis. Of course the other additions to the model are needed to be able to do the tracking in a meaningful and easy manner, but still this is where the focus of this dissertation lies.

  The goal is to have an easily extendible framework where extra tracking techniques can easily be added. Therefore, these tracking techniques need to be stored in the OC2 database. But, only as loosely possible, to keep the system extensible. Therefore, the Tracking Technique class will keep as little information as needed to store a tracking technique. The idea is that a tracking technique registers itself to the system and the system returns an id to the tracking technique, so the tracking technique has a way to identify itself in the the framework.

  A tracking technique has an extra property called a Tracking Property. Tracking Properties are related to a Tracking Technique. Each tracking technique has a certain way to identify an object. These can vary widely and in some cases this can be more than one property per tracking technique. Imagine a tracking technique that simply stores the name of a document. How the tracking techniques determines this name is of no matter. This name has to be stored for each document, so that, when later another document is tracked, it is possible to compare these names with each other and see if this document has been seen before.

If another tracking technique would have been used, for example, a tracking technique working with ids, again, how the ids are received are of no matter, the ids would have been stored in the Tracking Properties. This again, so it is possible to compare ids of documents based on the tracking technique.

When documents need to be compared, we only want to compare documents which have been tracked by the same tracking technique. This to avoid comparing titles with ids. This is why, the tracking property is not a separate part of the document, but is strictly related to the tracking technique that is being used. This also enables a document to be tracked by multiple tracking techniques, possibly on multiple places around the office. Because the tracking will come from the instance of a Tracking Technique there is no chance that any other tracking properties will be checked besides these related to the tracking technique that is currently used.

4.3.1 Mapping

Starting from the OC2 model the extensions for ReViTa are implemented. In stead of extending the OC2 model with extra functionality, extra database entries and extra logic the implementation of OC2 was done in such a way that a mapping sufficed to insert the extra functionality. The mapping could be done by making Resources in the OC2 model of the Organisational Structures and Util Links for the ReViTa endpoints that check whether or not a certain item is implemented. A complete overview of the mapping between ReViTa and the underlying OC2 components can be seen in Table 4.1.

The main reason to choose for a mapping is that there is no need to make any changes to the database that is being used in OC2. The version of OC2 that is being used in this dissertation is called iServer and makes use of a DB4O database. This database allows iServer to store any Java Object. This is different from other databases that have a relational database schema.

Together with the mapping between ReViTa and OC2 an entire REST interface has been built on top of ReViTa and even more on top of OC2. This has as goal to make the framework as extensible as possible. By giving users access to certain parts of the OC2 system they are being allowed to built a wider variety of applications that use more of the underlying system than when only the extensions would be provided to the user.

ReViTa	OC2
Lifeline	Resource
HasLifelines	Util Link
Tracking Property	Property
HasTrackingTechnique	Util Link
HasTrackingProperties	HasProperties
Owner Property	Property
HasOwnerProperties	HasProperties
Location Property	Property
HasLocationProperties	HasProperties
Duplicate	Resource
HasDuplicates	Util Link
Recognizer	Resource
HasRecognizer	Util Link
Comparator	Resource
HasComparator	Util Link
Pile	Resource
File	Resource
HasFilingLabel	Util Link
Mixture	Resource
HasMixingLabel	Util Links

Table 4.1: Mapping from ReViTa to OC2

4.4 Context Awereness in KOR

The context in which organisation structures are being used have already been discussed. The OC2 system takes context into account and stores them inside the system. These contexts need to be provided to the system. This will be done by the use of the Context-Modelling-Toolkit (CMT). This toolkit has been developed at the WISE lab of the VUB and has as goal to keep track of the context in which a user is acting.

4.4.1 Getting the Current Context

CMT consists out of two parts, a client and a server part. These can also be seen as a database and a Drools[1] engine on top. The database stores the rules and can be seen as the server, the Drools engine operates on the rules of a user and can be seen as a client. Drools is a business rule management

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

system (BRMS) based rule engine. It is cross platform, developed in Java and licensed under the ASL 2 license. The idea of a rules engine is that when certain things are fulfilled an action can be taken. These rules can be things like "All customers that spent more than 100 Euro at one time get a 10 percent discount". This rule can be boiled down to an **IF**, **THEN** rule. Whenever the **IF** has been met, in this case more than 100 Euro has been spent at once the **THEN** rule can come in action, being the user gets a discount of 10 percent.

The challenge with this kind of rule engines is to cope with conflicting rules or rules that overpower each other. If, for example, there would be a second rule stating that whenever a client spends more than 250 Euro they get a 15 percent discount it is up to the system to cope with these two rules. Does the system need to give the customer first his 10 percent discount and additionally the 15 percent discount, does the system rely on the oldest rule, and ignores the rule of the 15 percent or is there a need for some prioritizing the rules, so only the second will be applied. For humans it is easy to determine that only the second rule will apply, but this has to be implemented in the rule engine, which is not as trivial as it is for humans.

In the setting of the office environment, the context will be determined using these kinds of rules. Determining the context can be done, for example, that whenever the user in a meeting room their context is meeting. If their system knows that every Tuesday the user has a meeting with their colleagues it can be determined that this a meeting with colleagues. Once again the difficulty about which rule is more important emerges.

4.4.2 Storing the Context

To store the current context during the tracking, the current context will be provided by CMT. The goal is to, whenever a document is accessed the current context is added to the document. There is off course the notation of the weight of the context. Whenever a document is created within a context it can be assumed that this context is far more important to the document than when it is only accessed in this context. The same system applies for when the document has been moved. When a document is being moved within a certain context to another organisational structure, it can be assumed that this context is of far more importance than when a document is only accessed within a context. It is clear that a document can be relevant in more than one context. There is no current context for the document, there are multiple contexts in which the document is relevant but not all of these contexts are equally important.

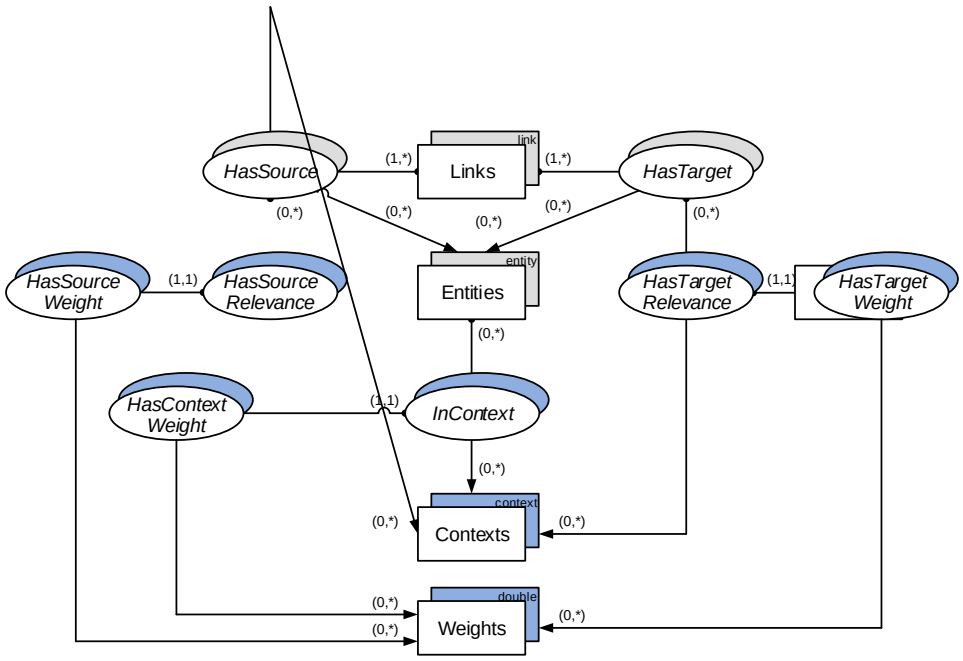


Figure 4.7: The context in the OC2 model

Adding all information needed to store the contexts and their weights in the OC2 can be done by adding some extra classes to the model. An overview can be seen in Figure 4.7.

4.5 Tracking Model Implementation

ReViTa, which provides the connection between OC2 and the outside world, is connected to the tracking component. The tracking component on its turn is connected to the tracking techniques. The architecture of the KOR framework is shown in Figure 4.1 can therefore be alternated to put the focus on the tracking components. In the general architecture there is only one tracking component above the KOR Framework, but in fact the goal of this tracking component is to connect multiple tracking techniques to the framework, as can be seen in Figure 4.8.

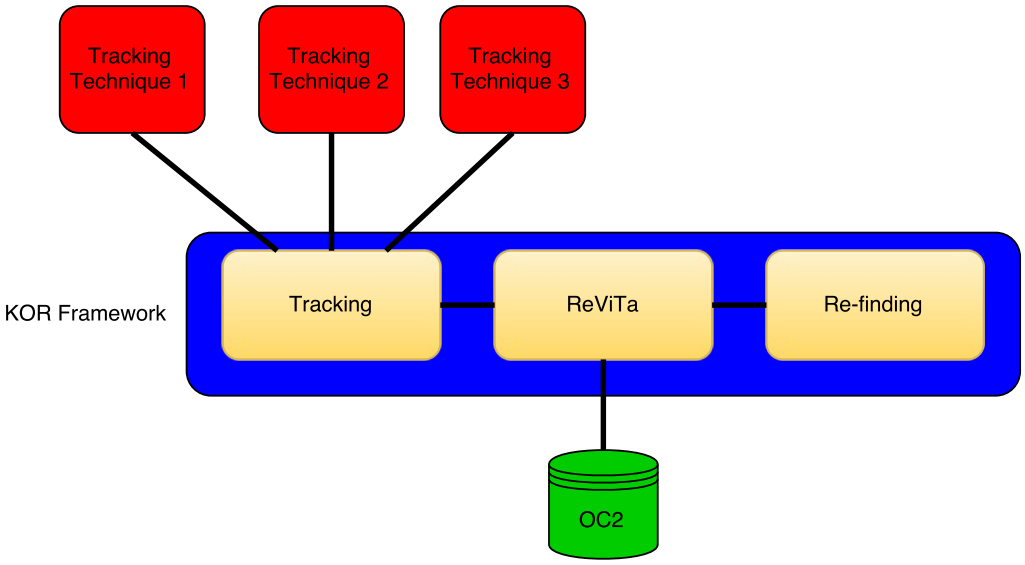


Figure 4.8: KOR architecture

The tracking component of the framework will provide an interface to the tracking techniques to communicate with the database of OC2. This will open up the ability to easily add tracking techniques. Adding tracking techniques will be made easy because adding the tracking technique to the system will be provided by a simple endpoint in the KOR framework. Additionally, an endpoint will be foreseen in the framework to add objects to the system. These objects can be files, piles, mixtures or just plain documents. For all of these, an interface will be provided to be able to add elements, delete elements, interact with elements, while in the background the tracking component of the framework will take care of the underlying changes that need to be made in the OC2 database. These underlying changes are implemented to keep track of the document. In the next part of this section, for each part it is explained what these changes include.

4.5.1 Objects

Objects can be seen as a superclass of Files, Piles and Mixtures, but also as the way to store a single document in OC2. This means that the interactions that are possible with Objects can be, or as a single document, or provided for all files, piles and mixtures.

Create Object

A first thing that needs to be done with an object, in the scope of this thesis a physical object, is creating an instance in the system. Creating this instance takes a few steps.

1. **Create new physical object:** To create a new Physical Object the first thing that needs to be done is create an instance, using ReViTa, in OC2. To do so, ReViTa provides an endpoint related to the core of OC2. *core/object/getNewPhysicalObject/* asks for the id of the user that wants to create the new physical object and returns the id of the freshly created object.

2. **Create "interactions" lifeline:** After creating the new object, some lifelines need to be added. The first lifeline is the "interactions" lifeline, which will store interactions with the object like its creation or when it has been accessed. To do so ReViTa provides the */extended/lifeLine/getNewLifeLine/* endpoint, which relates to the extended part OC2. To create a lifeline a name for this lifeline is needed. Note that the a freshly created lifeline has no relation to any object yet. This returns the id of the newly created lifeline.

3. **Create "locations" lifeline:** Similar to the "interactions" lifeline there is a "locations" lifeline created. This can be done using the same endpoint as is used to create the "interactions" lifeline.

4. **Add "create" entry to the "interactions" lifeline:** A first entry that needs to be added to the newly created lifelines is the notion that a new document has been created. To add an entry to a lifeline there is a call in ReViTa, */extended/lifeLine/addEntryToLifeLine/*, which takes a JSON object containing the lifeline id, a timestamp, an action label (in this case "create") and a parameter (by default 0) as parameter.

5. **Add "interactions" lifeline to the object:** As stated before, a newly created lifeline has no connection to an object. This means that these lifelines need to be added to the object, to create this relationship. ReViTa provides the *extended/object/addLifeLine/* endpoint for this. This endpoint gets as input a JSON object which contains the id of the object and the id of the lifeline that needs to be added.

6. **Add "locations" lifeline to the object:** Equally to the "interactions" lifeline, also the "locations" lifeline needs to be linked to the object.

7. **Add the current context:** A last thing that need to be done is adding the current context to the object. To do so the id of the object is required, the id of the current context and the weight we want to add to this context.

4.5.2 Files/Piles/Mixtures

In this section the interactions with files, piles and mixtures are explained. They are explained together even though, they have separate endpoints. There are endpoints for files, piles and mixture which are not related to each other. Even though, they will be explained together here. There are some minor differences between these endpoints, like for example, when a new pile is created it can just be created, while the creation of a new file requires a label. In the examples below the endpoints for piles will be used. These can whenever needed for a file or mixture be replaced. For example when the endpoint *extended/pile/getLifeLine/* is used in the example, this is obviously for a pile. When the same endpoints needs to be used for a mixture *extended/mixture/getLifeLine/* can be used and for a file *extended/file/getLifeLine/* can be used. For clarity only the pile endpoints will be used in this section. The same counts for notations on lifelines. Whenever notations like "addToPile" or "removeFromPile" are used these are interchangeable with "addToMixture" or "addToFile".

Create File/Pile/Mixture

Files,piles and mixture are a types of organisational structures that also have an interface in the tracking component of the KOR Framework. The creation of files, piles and mixtures is very similar to the creation of a physical object and runs through the same steps as the creation of a physical object.

1. **Create new file/pile/mixture:** A new file, pile or mixture is created. Depending if it is a file or a mixture a label, respectively will or could be needed to create this new organisational structure.

2. **Create "interactions" lifeline:** The interactions lifeline is equal to the interactions lifeline of an object.

3. **Create "locations" lifeline:** The locations lifeline is equal to the locations lifeline of an object.

4. **Add "create" entry to the "interactions" lifeline:** The interactions lifeline gets a "created" entry together with the time stamp on which the structure has been created.

5. **Add "interactions" lifeline to the structure:** The interactions lifeline gets linked to the structure.

6. **Add "locations" lifeline to the structure:** The locations lifeline gets linked to the structure.

7. **Add the current context:** The current context is added to the structure.

Add Document to File/Pile/Mixture

Besides creating a structure, some other operations on these structures should be provided. The first and most important one is adding a document to these structures. To add a document to these structures there are certain steps that need to be taken. To add a document to a structure there are only 2 things needed. These are the id of the structure and the id of the document that needs to be added. Starting from this, everything can be managed following these steps:

1. **Add "access" entry to lifeline:** Using the ReViTa endpoint *extended/pile/getLifeLine/* we can retrieve the correct lifeline from ReViTa. This is done by adding the id of the pile and the name of the lifeline to this call. The name of the lifeline in this case is "interactions". To add an "access" entry to lifeline the there is a function that will be discussed in the LifeLines section later in this chapter. The core idea is that the ReViTa endpoint *extended/lifeLine/addEntryToLifeLine/* is called, with a JSON object generated by the LifeLines class as body.

2. **Add a current context to a structure:** When a structure is used in a certain context, this context needs to be added to this structure. This is done by calling the *extended/pile/addToContext/* endpoint of ReViTa, which uses as body a JSON object containing information like pileId, contextId, and weight. How this weight and contextId can be retrieved will be explained later in this chapter.

3. **Add a document to a structure:** The most important part of the call is adding the document to the structure. This is done by using the call *extended/pile/addObjectInContext/* in ReViTa. This call takes the pile id, object (document) id, context id, weight and the index where the document needs to be inserted in the pile as parameters. When this index is 0 the document is added at the default location.

4. **Update document location:** Now that the document has been moved from its old place to the structure the location property of the document needs to be updated. Besides the update of the current location the old location needs to be stored in the "locations" lifeline of the document.

5. **Add "addPile" entry to lifeline of the document:** Again an entry needs to be added to a lifeline. How lifelines can be retrieved has been explained in adding the "access" entry to a lifeline. But instead of adding accessed to the lifeline here "addPile" will be added to the document, to store that the document has been stored in a pile.

Remove Document from File/Pile/Mixture

Next to adding a document to a structure there should also be the possibility to remove documents from the structure. Similar to adding a document to a structure this operation also requires the id of the structure and the id of the document to be able to be fulfilled.

1. **Add "access" entry to lifeline:** Equal to the first step of adding a document to a structure.

2. **Delete document from file/pile/mixture:** To delete a document from a structure ReViTa has the *pile/deleteObject/* endpoint. This endpoint requires the id of the pile and the id of the document that needs to be removed to be able to delete the document.

3. **Add "remFromPile" to document lifeline:** Equal to the "addPile" lifeline step in adding a document to a structure. Only the string that gets added to the lifeline differs for adding this entry to the lifeline.

Note that behind the scenes, ReViTa also takes care of the fact that the document could be the only document in a certain context in the structure. If so this context should be removed from the structure, because the structure no longer contains anything within this context.

Access in File/Pile/Mixture Without Adding or Removing Documents

A structure can be accessed by a user within a certain context without that there is document added or removed respectively to or from the structure. Still, this needs to be tracked because it is useful to know in which contexts a structure has been consulted.

To track this, it is only needed to know the id of the structure and then the following steps need to be taken:

1. **Add "access" entry to lifeline:** This step is equal to the first steps of adding or removing a document to or from a structure.

2. **Add "access" entry to lifelines of all objects in the structure:** When a structure is accessed, it can be assumed that every document within this structure has been accessed. This means that every document in the structure needs to be traversed and gets an accessed entry added to their lifeline. This entry can be added in the same way as it has been done for the entire structure, but then on an object in stead of on a structure. Traversing the structure can be done by calling the *extended/pile/getTargets/* endpoint provided by ReViTa. When given the id of the structure this returns a JSON object containing an array with all the ids of documents contained by this structure. Traversing over this array gives the possibility to add the accessed entry to their lifelines.

3. **Add current context to structure:** When a structure is accessed, it has been done within the current context. This means that the current context needs to be added to the structure. This can be done by calling the *extended/pile/addToContext/* ReViTa endpoint with as body a JSON object containing the structure id, context id and the weight, as has been explained before.

Add new File, Pile or Mixture

There is the possibly to store, next to documents, files, piles or mixtures inside structures. This can be done by adding a new file, pile or mixture to the existing structure. To do so, the structure to which the file, pile or mixture needs to be added is needed. If it is a new file that needs to be added a label is needed, for piles no label is needed and for mixtures it is optional. Using this label a new file, pile or mixture can be created which later on gets stored in the structure. To add the file, pile or mixture the index where it needs to be added to the structure needs to be provided. If no index is provided the new organisational structure gets added at the default location of the structure. For files this is at the end of the file, for piles this is at the top of the pile.

To add this new file, pile or mixture to the existing structure a few steps need to be taken.

1. **Create new file/pile/mixture:** Depending on the type of the organisational structure that needs to be added a label needs to be provided. For each type of organisational structure that can be added a new endpoint is provided.

2. **Add the new file/pile/mixture to the existing structure:** As for every organisational structure that can be added a different endpoint is foreseen.

Add a File, Pile or Mixture to the File/Pile/Mixture

Next to adding a new file, pile or mixture to a structure also an existing file, pile or mixture can be added to a mixture. To be able to do so the id of both the organisational structure that gets added as the id of the structure it gets added to are required. With these two ids the following steps can be taken.

1. **Add "access" entry to lifeline of structure:** The structure to which a new organisational structure is added is accessed. This needs to be placed on its lifeline.

2. **Add current context to structure:** This structure is accessed within a certain context. Also this context needs to be added.

3. **Add file, pile or mixture to selected structure in current context:** The organisational structure gets added to the original structure.

4. **Update the organisational structure that is added its location:** The location of the organisational structure that was added to another structure needs to be updated, as it has been moved to this other structure.

5. **Add "filed" entry to added organisational structure's lifeline:** The organisational structure that was filed needs to get this entry in its interactions lifeline.

Remove a File, Pile or Mixture From the Structure

After adding organisational structures to a structure there is the need to be able to remove them as well. This can happen when an organisational structure from the structure gets moved into another organisational structure.

1. **Add "access" entry to structure his lifeline:** There was an interaction with the structure, so this should be placed on its lifeline.

2. **Remove the file, pile or mixture from the structure:** The actual removing of the structure from the structure needs to be done.

3. **Add "remFromFile" to file, pile or mixture lifeline:** The structure that has been removed needs to get this noted on its interactions lifeline.

4.5.3 Tracking Techniques

Even though tracking techniques are a core matter of this dissertation and a very important part of this research, within the KOR framework they are only a minor part. This is because the tracking techniques need to be loosely coupled to the framework, which makes it easier to add other tracking techniques without having any knowledge of how the framework handles them.

Something that has to be possible within the framework is creating a new tracking technique. To do so only a name for this tracking technique is needed. ReViTa provides an endpoints which, getting this name, returns the id of the newly created tracking technique. To add this tracking technique to an object there is an endpoint on an object to add a tracking techniques. This endpoint requires the id of the object, the id of the tracking technique and the URI that identifies the document within this tracking technique.

Of course, it is needed to know what has been tracked by a tracking technique. To find this, there is a call in ReViTa */extended/trackTech/getObjectsTrackedByMe/* that returns every object that is tracked by a Tracking Technique. This of course requires the id of the Tracking Technique as input and returns a JSON object containing an array with all these objects.

When a Tracking Techniques gets triggered, which means that he tracks a new object, the first goal is to know whether or not this object has been seen before. To know this, there should be a way of knowing if a single document is tracked by the tracking technique or not. To know this, the array with all objects that are being tracked by a tracking technique needs to be traversed.

While traversing this array the new object's generated URI needs to be compared to those of the objects in the array. Retrieving the URI of an object, using a specific tracking technique can be obtained by calling the */object/getTrackTechPropertyURI/* endpoint in ReViTa. This call takes the id of the object and the id of the tracking technique as input and returns the URI. Based on this URI, some comparison can be done individually for each tracking technique, to determine if these documents are equal.

Objects have some extra interactions with tracking techniques. A first one is to delete a tracking technique. To delete a tracking technique from an

object it is needed to know the ids of the object and the tracking technique.

The URI of a tracking technique, related to an object, can be updated by providing the id of the object, the tracking technique and the new URI.

Next to this, ReViTa provides an interface to add extra tracking technique properties. These properties are added by providing the object id, tracking technique id and property id. These properties can be removed as well. Next to adding these properties they need to be handled as well. This is why there is an endpoint that returns all tracking properties of the object.

4.5.4 LifeLines

LifeLines have already been discussed extensively. The tracking techniques have no direct interaction with the LifeLine implementation of ReViTa. Most communication with LifeLines happens through the Tracking part of the KOR framework. ReViTa provides an endpoint to create a new lifeline, which only requires a name. This returns the id of the new lifeline. To add an entry to the lifeline there is an endpoint in ReViTa that asks the id of the lifeline, a timestamp, an action label and a parameter. Using this input the lifeline entry can be added to the lifeline.

Previously, endpoints to create lifelines and add entries to lifelines have been used. Also the endpoint to retrieve a lifeline from a file/pile/mixture using the name of the lifeline has been explained. There are some extra endpoints concerning lifelines implemented. Except finding a lifeline using the id of an object and the name of a lifeline, a lifeline can also be returned by using the id of the lifeline. This will return a JSON object containing an array with all entries of the lifeline. These entries consist of the timestamp and another array containing the actions. These actions contain the action itself and a parameter.

Lifelines are used for a lot of interactions with files, piles and mixtures. Often the same entry is added at different places. Therefore there are some frequently used entries for lifelines listed in a special lifeline class. The class creates the JSON object that is needed add a certain entry to a lifeline. This JSON object contains the id of the lifeline, a timestamp, an action label and a parameter. Special entries have been foreseen for the following action labels: *create*, *access*, *filed*, *remFromFile* (removed from file), explore, *addMix* (added to mixture), *remFromMix* (removed from mixture), *addPile* (added to a pile) and *remFromPile* (removed from pile). Extra entries can be created on the fly or extra entries can be added to this lifeline class.

5 Proof of Concept Plug-ins

The KOR framework that has been introduced has no use unless there are some Tracking Techniques attached to it. To make use of the framework, as a proof of concept some Tracking Techniques have been implemented. The main idea is that user develops their own plug-ins. The Tracking Techniques which are presented here are designed to work within certain situations. These examples have been carefully chosen so every part of the KOR framework could be used.

Recogniser	**Comparator**
SIFT	SIFT
OCR	Levenshtein Distance
Colour Intensity	Matrix Comparator
RFID Tags	String Comparator
TUIO Tags	String Comparator

Table 5.1: Identifiers and their counterpart comparator

As discussed before these plug-ins consist out of two parts, being the recogniser and the comparator. There are five recognisers implemented and four comparators. This because the RFID tag recogniser and the TUIO tag recogniser both use string comparator. An overview of each identifier and the comparator they could use can be seen in Table 5.1.

5.1 Identification Using Images of Documents

In the next section the recognising of documents will be explained. For some of these recognising techniques, like OCR, SIFT and Colour Intensity there is need for an image of the document. Gathering an image of a document is not as easy as it seems. To gather an image of the document one can start from a picture containing the document. Then it is the case to extract the image of only the document from this picture. To extract the image from the picture some actions need to be taken, but even more, some other questions arise, like when to take a picture?

5.1.1 When to Make a Picture

To determine the identity of a document the first thing that is needed is a photo of this document. This photo needs to contain a relevant image to extract the document. This means this photo needs to be taken at a moment that is most opportune to take a snapshot of the pile. This moment is every time there was some action around the pile. To know when this happens, it is necessary to keep a mechanism on the pile that tracks actions around the pile.

Figure 5.1: Microsoft Kinect

To do this kind of tracking there is need to track the user interactions with the pile. Tracking people is mostly done by the use of a Microsoft Kinect camera, or some likewise camera, as illustrated in Figure 5.1. The Kinect can track people around the pile. But as the pile is used on a desk it does not mean that when a person is near the desk an interaction with the pile will or

has happened. It is only the pile that needs tracking. As users interact with a pile using their hands it is only the movement of hands above the pile that needs to be tracked.

As the Kinect is created to do full body tracking the software provided by the Kinect will not suffice to do this kind of tracking. 3Gear[1] provides a library that, using a Kinect, or likewise camera, allows the tracking of hands.

3Gear will allow to track hand movements above the pile. This software is not enhanced enough to track whether or not a document has been added or removed from the pile. Therefore, this software will only be used to track if hands are present above the pile. Whenever hands have been seen and have been removed again, an intensive will be given to create a new image. Based on the knowledge of the previous state of the pile, and the possibility to identify the document on top of the pile, it is possible to determine if one or more documents have been removed or if a document was added.

5.1.2 Create the Picture

When it has been determined when the picture needs to be taken, the next goal is to create this picture. Now that it is decided when the picture needs to be taken, there is need for the possibility to take this picture without that the user needs to be involved. To be able to create this picture, we need a device that can take a picture by command from a piece of software. This can be achieve by attaching a webcam the to pc that runs the program that decides when the picture needs to be taken. This can than force the webcam to create a snapshot. The quality from a webcam, however, is not good enough to be able to do decent image recognition. This is why there is need for an IP webcam. This IP camera can be a dedicated device, but to make it more extensible it is more interesting to use an Android phone equipped with an application to offer the possibilities of an IP camera.

Figure 5.2: Samsung Galexy Camera 2

[1]http://www.threegear.com/

To be certain the images are of good quality, there has been chosen for a camera that supports Android, in stead of a smartphone that is equipped with a camera. In this case, the Samsung Galaxy Camera 2 illustrated in Figure 5.2 has been chosen as device. This device has approximately 16.3 megapixels and a 21x Zoom Lens, which provides photos of a good enough quality to do any kind of image recognition upon.

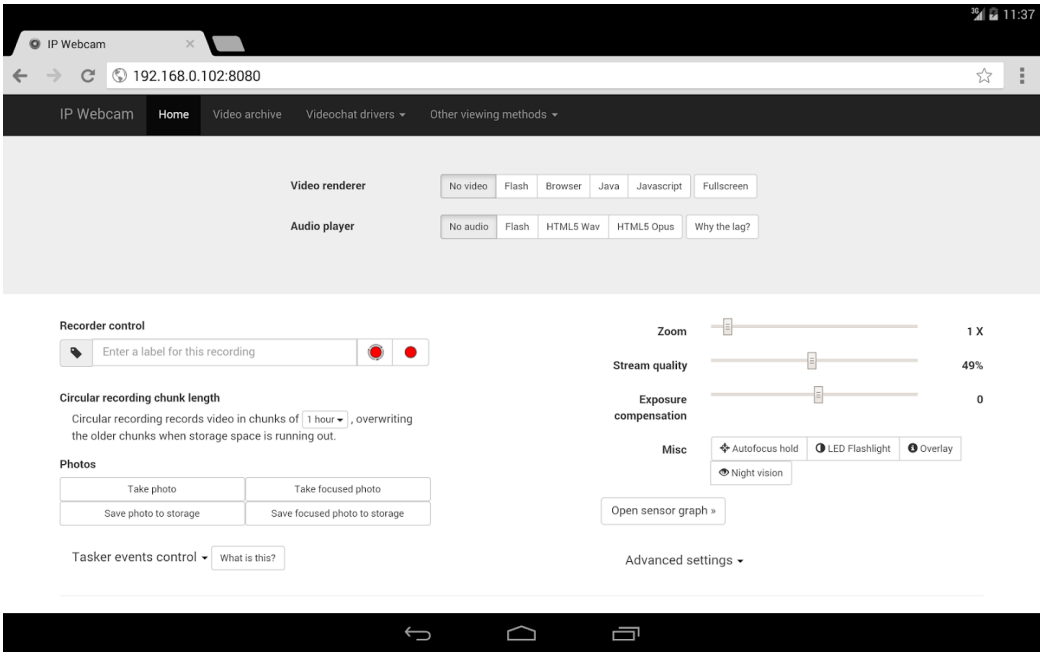


Figure 5.3: IP Webcam web-interface

The Android application that is being used for this is IP Webcam[2] developed by Pacel Khlebovich. This application provides a web interface for the camera at the Android device, as illustrated in Figure 5.3. Next to this web interface which provides live streams of the camera and options like zoom, stream quality and exposure compensation the app also provides certain URIs which return images. In the context of this thesis, only two URIs are interesting. These are /photo and /photoaf. The latter one takes a focussed photo. It is this photo that will be used to extract the image of the pile.

[2]http://ip-webcam.appspot.com/

5.1.3 Extract the Image of the Document

To extract the image of the document from the picture that has been taken by the IP Webcam application, the same software can be reused as was developed last year at the WISE lab. This software makes use of the JavaCV[3] library, which is a translation of the Open Source Computation Vision[4] (OpenCV) library for Java. OpenCV is a library that aims to provide real-time computer vision functions.

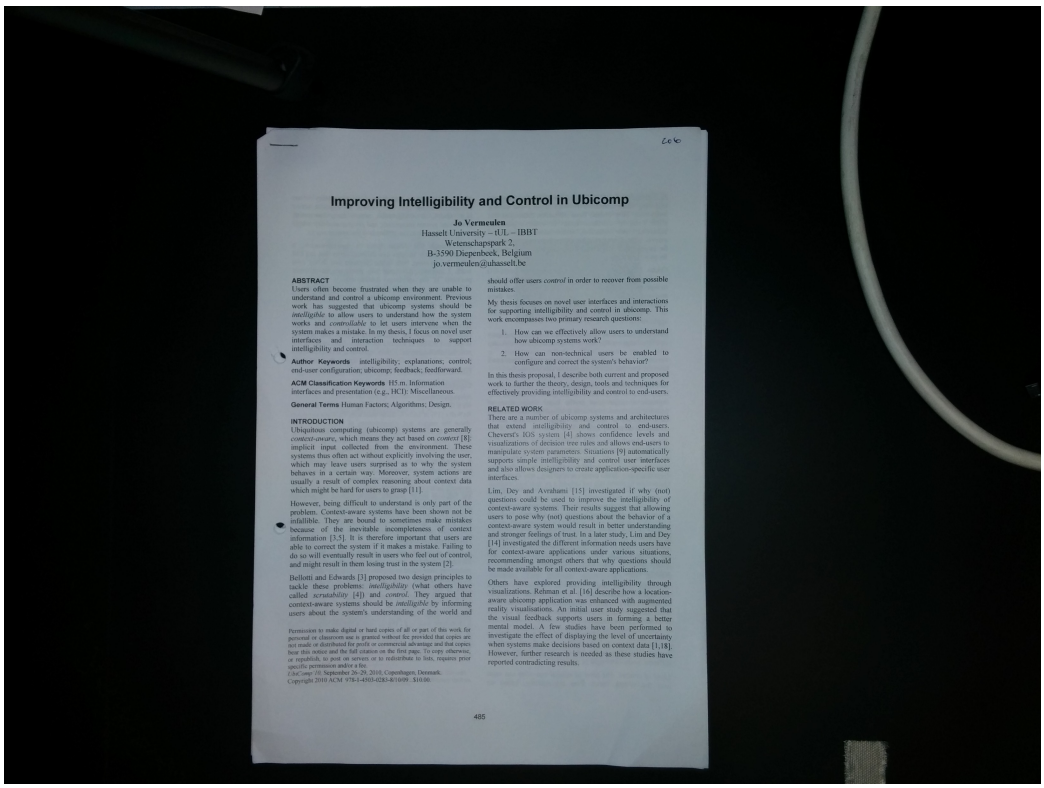
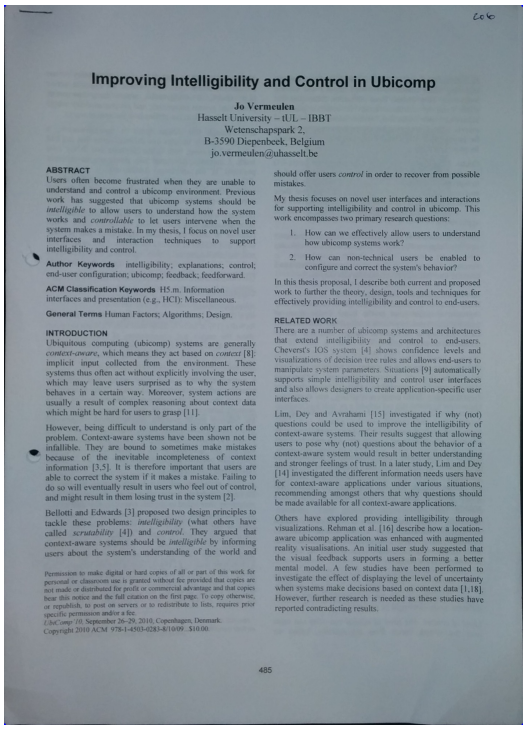


Figure 5.4: The picture of the document

To extract the image of the document from the picture, a series of steps needs to be taken. These steps will be explained here in more detail, but in short the idea is to use the contrast between the document and the background. This is why it is a requirement that the background of the image is black. This is a minor requirement for otherwise having a totally automated system to track documents in piles, an example of a picture of a document with a black background can be seen in Figure 5.4.

[3]https://github.com/bytedeco/javacv
[4]http://opencv.org/

The first step that needs to be taken to extract the image is convert the original image to its grey image. Even though, a document is white and the background is black it is still more opportune to convert to a grey image because this will enhance the difference in contrast in the image.



Improving Intelligibility and Control in Ubicomp

Jo Vermeulen
Hasselt University – tUL – IBBT
Wetenschapspark 2,
B-3590 Diepenbeek, Belgium
jo.vermeulen@uhasselt.be

ABSTRACT
Users often become frustrated when they are unable to understand and control a ubicomp environment. Previous work has suggested that ubicomp systems should be *intelligible* to allow users to understand how the system works and *controllable* to let users intervene when the system makes a mistake. In my thesis, I focus on novel user interfaces and interaction techniques to support intelligibility and control.

Author Keywords intelligibility; explanations; control; end-user configuration; ubicomp; feedback, feedforward.

ACM Classification Keywords H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms Human Factors; Algorithms; Design.

INTRODUCTION
Ubiquitous computing (ubicomp) systems are generally *context-aware*, which means they act based on context [8]: implicit input collected from the environment. These systems thus often act without explicitly involving the user, which may leave users surprised as to why the system behaves in a certain way. Moreover, system actions are usually a result of complex reasoning about context data which might be hard for users to grasp [11].

However, being difficult to understand is only part of the problem. Context-aware systems have been shown not be infallible. They are bound to sometimes make mistakes because of the inevitable incompleteness of context information [3,5]. It is therefore important that users are able to correct the system if it makes a mistake. Failing to do so will eventually result in users who feel out of control, and might result in them losing trust in the system [2].

Bellotti and Edwards [3] proposed two design principles to tackle these problems: *intelligibility* (what others have called *scrutability* [4]) and *control*. They argued that context-aware systems should be *intelligible* by informing users about the system's understanding of the world and should offer users *control* in order to recover from possible mistakes.

My thesis focuses on novel user interfaces and interactions for supporting intelligibility and control in ubicomp. This work encompasses two primary research questions:

1. How can we effectively allow users to understand how ubicomp systems work?
2. How can non-technical users be enabled to configure and correct the system's behavior?

In this thesis proposal, I describe both current and proposed work to further the theory, design, tools and techniques for effectively providing intelligibility and control to end-users.

RELATED WORK
There are a number of ubicomp systems and architectures that extend intelligibility and control to end-users. Cheverst's IOS system [4] shows confidence levels and visualizations of decision tree rules and allows end-users to manipulate system parameters. Situations [9] automatically supports simple intelligibility and control user interfaces and also allows designers to create application-specific user interfaces.

Lim, Dey and Avrahami [15] investigated if why (not) questions could be used to improve the intelligibility of context-aware systems. Their results suggest that allowing users to pose why (not) questions about the behavior of a context-aware system would result in better understanding and stronger feelings of trust. In a later study, Lim and Dey [14] investigated the different information needs users have for context-aware applications under various situations, recommending amongst others that why questions should be made available for all context-aware applications.

Others have explored providing intelligibility through visualizations. Rehman et al. [16] describe how a location-aware ubicomp application was enhanced with augmented reality visualisations. An initial user study suggested that the visual feedback supports users in forming a better mental model. A few studies have been performed to investigate the effect of displaying the level of uncertainty when systems make decisions based on context data [1,18]. However, further research is needed as these studies have reported contradicting results.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
UbiComp '10, September 26–29, 2010, Copenhagen, Denmark.
Copyright 2010 ACM 978-1-4503-0283-8/10/09...$10.00.

Figure 5.5: The extracted image of the document

Next, the contours of the image need to be searched. This can be done by a built-in function of the JavaCV library and will find points with the highest contrast and mark them as contours. Based on these contours the built in function to find polygons can be used. This function will, based on the place of the contours, check if there are polygons formed by these points.

When a polygon is found, the four correct corners for the document need to be selected. The selection of the right polygon is based on the fact that if four points make some sort of rectangle, the rectangle can be distorted by the angle of the camera upon the document, and the size of the polygon. This size is an extra check that is created to be sure no that small squares that are on the picture can be seen as a document. Once the four corner points of the document have been detected the image can be cropped based on these four points. The cropped image can then be altered to be easier to use. This can be a rotation of the image, when the pile is not lined up with

the camera, to have a good cropped image, or as stated before, when the camera is placed upon an angle the image can be altered to recreate a decent rectangle. When the picture is distorted, often the top of the document is wider than the bottom. To fix this problem the bottom can be stretched out so it document is a nice rectangle again. The result of a cropped image can be seen in Figure 5.5.

5.2 Recognising

Tracking a document consists out of two component. A first component is recognising the document. This means that the documents needs to be identified to a single URI. This URI can be stored in the OC2 database. Using this URI it can be determined if a document has been seen before or not. This means that the URI that defines the document needs to be unique and every time the same document is scanned the same URI needs to be returned.

This theory does not only apply to documents. Whenever any sort of organisational structure needs to be tracked a single URI to identify the organisational structure needs to fulfil these requirements.

5.2.1 Optical Character Recognition

The first way to uniquely identify a document is storing its title. Storing the title means that the title needs to be provided. As in the context of tracking documents, there is only a view of the document at hands. As discussed in the previous section an image of the document can be extracted. Using this image there are possibilities to identify the title of a document. This can be done by Optical Character Recognition, or OCR.

OCR is field of research in computer science, more specifically pattern recognition, artificial intelligence and computer vision. The goal of OCR is the mechanical or electronic conversion of typed, handwritten or printed information into machine-encoded text.

OCR has a wide variety of uses. One of the original reasons for inventing OCR is for the blind and visually impaired. OCR can help to translate written documents to sound, so that these people do not need help of other people to identify the content of documents. In 1974, Ray Kurzwell started a company called Kurzweil Computer Products Inc. to develop the use of OCR to help the blind and visually impaired.

Currently OCR is widely accepted and used in various ways. Google Drive, for example, has a function to convert text that can not be altered,

for example on a .jpg, .png, .gif or more often .pdf to Google Docs, which the user can alter. This enables users to make changes in documents of which only a photo is present, or a pdf because they have been scanned, or sent by someone without providing the file which can be changed.

Less obvious ways that OCR can be used is, for example, in automatic number plate recognition. Currently, there are camera's on roads which register the number plate of a car using OCR. This can be done to do some checks if the car is in the database of the police, for example, to know if the car is insured or is owned by someone who the police is paying extra attention to. Next to this, there are things like section control on motorways, where the time a car passes a certain point is registered and once again the time is registered when the cars passes another point further down the road. Based on the time the car needed to go from the first pass to the second and the distance between the two points the average speed of the car can be computed. This can catch speeding drivers.

Besides these, there are a number of ways users want to register the text from a written document in a digital representation. For example, add the content of a business card to a contact list, data entry for documents, or make digital representations of printed documents.

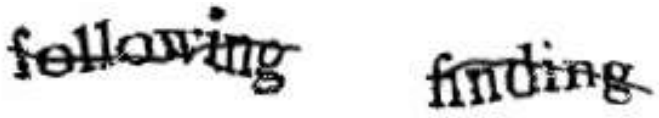


Figure 5.6: A captcha

Not everything about OCR is good of course. Currently there are a lot of users with bad intentions that use OCR to automatically fill in forms on websites. With automatically filling forms these users fill databases with useless users and try to take down server by overloading the number of accesses. To prevent computers from making accounts a CAPTCHA was invented. CAPTCHA is an acronym for Completely Automated Public Turing test to tell Computers and Humans Apart. A CAPTCHA is a number, some letters, or some words which are unreadable for OCR. An example of a CAPTCHA can be seen in Figure 5.6.

Because OCR cannot handle the characters inside a CAPTCHA there is no way a computer can create accounts that require a CAPTCHA. This can ensure the owner of the website that the account is created by a person and not by a computer.

Tesseract

In the context of creating a URI for a document, the focus can be put on solely the title of the document. It can be assumed that the title of the document can be used as a unique identifier. This means that the document can be cropped to only the place where the title is present. On this part of the document OCR can be done. An example of the image on which the OCR could be done is shown in Figure 5.7.

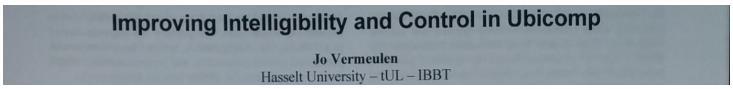


Figure 5.7: Image of the title of a paper.

To do this tracking the Tesseract library is being used. Tesseract is an Open Source OCR Engine. It was originally developed at Hewlett-Packard Laboratories Bristol and at Hewlett-Packard Co, Greely, Colorado between 1985 and 1994. In 1996 there were some extra changes made to port it to Windows and there was some migration done from C to C++ in 1998. In 2005 it got released as open source by Hewlett-Packard and the University of Nevada, Las Vegas. Currently development is being sponsored by Google, since 2006. Now the code in most recent distribution is licensed under the Apache License.

Currently, Tesseract is available to do OCR in over a hundred languages. Tesseract can be used directly by programmers by use of an API. It does not support a GUI but there are some third party GUIs available. To interact with Tesseract there is a command-line interface present. The use of the command is shown below.

```
tesseract imagename outputbase [-l lang] [-psm pagesegmode] [configfile...]
```

If a user wants to do a basic scan of the document called 'myscan.png' and save it to 'out.txt' the command would look like this:

```
tesseract myscan.png out
```

Or in German:

```
tesseract myscan.png out -l deu
```

In the context of extracting the title only the normal way of scanning an image is needed. Later on, the 'out.txt' file can be read and its content can be put in the database as unique URI for this document.

5.2.2 Scale-Invariant Feature Transform

Having extracted the image of the document, rests the task to be able to compare different images with each other to know if this document has been seen before, and, if not, to store some essentials data for the image so it can be compared again. To do so, there are multiple possibilities. One example of a way to store information about a document is Scale-Invariant Feature Transform (SIFT). SIFT is an algorithm in computer vision that finds, detects and describes features in an image. The algorithm was first published in 1999 by David Lowe [37].

Figure 5.8: In the top three photos the training data for the system is used. Later this data is used upon the photo in the middle. The photo at the bottom shows the matches that have been found by the system.

The idea of the SIFT algorithm is to define local features in an image. These features are then saved in a large collection of local vector features, each of which is invariant to image translation, scaling and rotation. Because features are being used, it is possible to re-find the same item on different photos. Even more, because only a number of features needs to match, not even the entire item needs to be shown in the picture. An example can be seen

in Figure 5.8. The system is trained with the pictures of a shoe, a phone and a teddy with a black background. When these items are afterwards placed in another picture, as can be seen in the picture in the middle, and the algorithm is ran, SIFT can still find these items in the picture. In the bottom picture it can be seen how the algorithm enlights the features that match the features that were stored in the vector.

Figure 5.9: The result of the SIFT algorithm when finding items which are not entirely visible

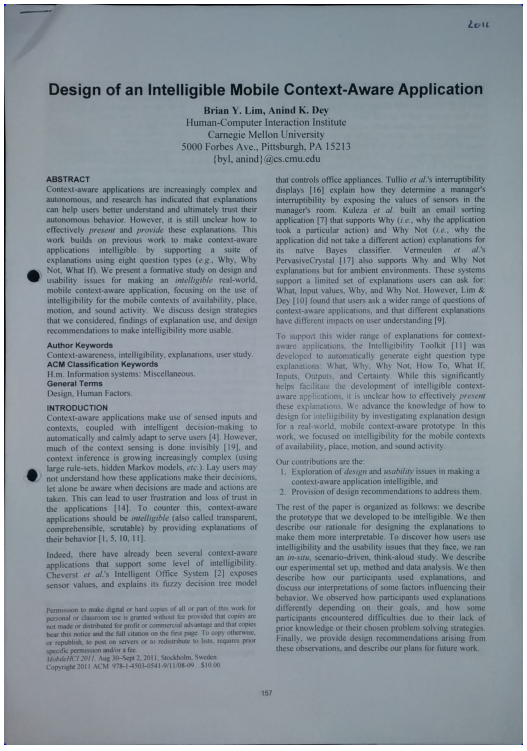
Even when the items are only partly visible, the SIFT algorithm can re-find them in a picture. In Figure 5.9 can be seen how the items that have been learned are placed in a room where they lay partly on top of each other or are partly covered by other items. In both pictures it can be seen how the algorithm is still able to find these items in the clutter. If an item can be found or not, is based upon the fact that only a number of features need to match before the algorithm decides that the item is present in the picture. Whenever a certain threshold of matching features, also called keys, is passed the algorithm decides that the item is present, but just not entirely visible.

Not only having some parts of the image covered can be overcome by the SIFT algorithm. In Table 5.2 is shown how the algorithm reacts under all sorts of distortion of the image. In this table the effect of having the image stretched, scaled or rotated are shown. Just as having the contrast of the image enlarged, the intensity decreased or added pixel noise to the image.

For this test 20 diverse images are used which results in about 15000 keys. The first column gives the percentage of matching keys in the transformed image. For the second column an extra rotation of 20 degrees was added

Image Transformation	**Match %**	**Ori %**
A. Increase Contrast by 1.2	89.0	86.6
B. Decrease intensity by 0.2	88.5	85.9
C. Rotate by 20 degrees	85.4	81
D. Scale by 0.7	85.1	80.3
E. Stretch by 1.2	83.5	76.1
F. Stretch by 1.5	77.7	65.0
G. Add 10% pixel noise	90.3	88.4
H. All of A,B,C,D,E,F,G.	78.6	71.8

Table 5.2: For various transformations applied to a sample of 20 images, this table gives a percent of keys that are found at matching locations and scales (Match %) and that also match in orientation (Ori %)



Design of an Intelligible Mobile Context-Aware Application

Brian Y. Lim, Anind K. Dey
Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
{byl, anind}@cs.cmu.edu

ABSTRACT
Context-aware applications are increasingly complex and autonomous, and research has indicated that explanations can help users better understand and ultimately trust their autonomous behavior. However, it is still unclear how to effectively *present* and *provide* these explanations. This work builds on previous work to make context-aware applications intelligible by supporting a suite of explanations using eight question types (e.g., Why, Why Not, What If). We present a formative study on design and usability issues for making an *intelligible* real-world, mobile context-aware application, focusing on the use of intelligibility for the mobile contexts of availability, place, motion, and sound activity. We discuss design strategies that we considered, findings of explanation use, and design recommendations to make intelligibility more usable.

Author Keywords
Context-awareness, intelligibility, explanations, user study.
ACM Classification Keywords
H.m. Information systems: Miscellaneous.
General Terms
Design, Human Factors.

INTRODUCTION
Context-aware applications make use of sensed inputs and contexts, coupled with intelligent decision-making to automatically and calmly adapt to serve users [4]. However, much of the context sensing is done invisibly [19], and context inference is growing increasingly complex (using large rule-sets, hidden Markov models, *etc.*). Lay users may not understand how these applications make their decisions, let alone be aware when decisions are made and actions are taken. This can lead to user frustration and loss of trust in the applications [14]. To counter this, context-aware applications should be *intelligible* (also called transparent, comprehensible, scrutable) by providing explanations of their behavior [1, 5, 10, 11].

Indeed, there have already been several context-aware applications that support some level of intelligibility. Cheverst *et al.*'s Intelligent Office System [2] exposes sensor values, and explains its fuzzy decision tree model that controls office appliances. Tullio *et al.*'s interruptibility displays [16] explain how they determine a manager's interruptibility by exposing the values of sensors in the manager's room. Kuleza *et al.* built an email sorting application [7] that supports Why (*i.e.*, why the application took a particular action) and Why Not (*i.e.*, why the application did not take a different action) explanations for its naïve Bayes classifier. Vermeulen *et al.*'s PervasiveCrystal [17] also supports Why and Why Not explanations but for ambient environments. These systems support a limited set of explanations users can ask for: What, Input values, Why, and Why Not. However, Lim & Dey [10] found that users ask a wider range of questions of context-aware applications, and that different explanations have different impacts on user understanding [9].

To support this wider range of explanations for context-aware applications, the Intelligibility Toolkit [11] was developed to automatically generate eight question type explanations: What, Why, Why Not, How To, What If, Inputs, Outputs, and Certainty. While this significantly helps facilitate the development of intelligible context-aware applications, it is unclear how to effectively *present* these explanations. We advance the knowledge of how to design for intelligibility by investigating explanation design for a real-world, mobile context-aware prototype. In this work, we focused on intelligibility for the mobile contexts of availability, place, motion, and sound activity.

Our contributions are the:
1. Exploration of *design* and *usability* issues in making a context-aware application intelligible, and
2. Provision of design recommendations to address them.

The rest of the paper is organized as follows: we describe the prototype that we developed to be intelligible. We then describe our rationale for designing the explanations to make them more interpretable. To discover how users use intelligibility and the usability issues that they face, we ran an *in-situ*, scenario-driven, think-aloud study. We describe our experimental set up, method and data analysis. We then describe how our participants used explanations, and discuss our interpretations of some factors influencing their behavior. We observed how participants used explanations differently depending on their goals, and how some participants encountered difficulties due to their lack of prior knowledge or their chosen problem solving strategies. Finally, we provide design recommendations arising from these observations, and describe our plans for future work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MobileHCI 2011, Aug 30–Sept 2, 2011, Stockholm, Sweden.
Copyright 2011 ACM 978-1-4503-0541-9/11/08-09...$10.00.

Figure 5.10: The image of a document.

5.2.3 Colour Intensity

Using the image of the document, there should be an easy way to store this image and make it comparable to another image. Last year, at the WISE lab at the VUB, a Rapid Prototyping Framework for Identifying and Tracking Paper Documents on Desks has been developed. This framework used the same kind of image that is being used in this thesis and transforms this into an easy storable and comparable format.

The idea behind this framework is to store the colour intensity of every pixel in the image at hand. This pixel his colour intensity would be stored in a matrix, which makes the comparing easy. Starting from an image like Figure 5.10, such an matrix could be created. This matrix on its turn could then be stored as the URI defining this document in the OC2 database.

To get to this matrix there are some steps that need to be taken. Using the openCV software that was used for the extraction of the image of the document from the picture it can be determined what the value of the pixel colour is.

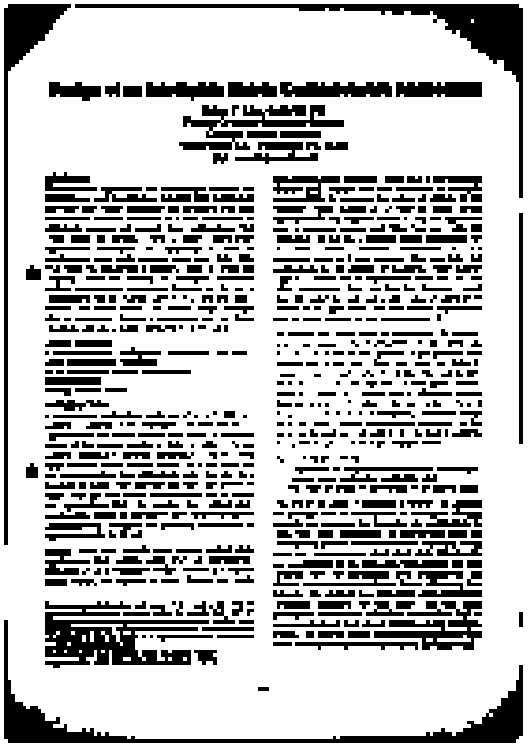
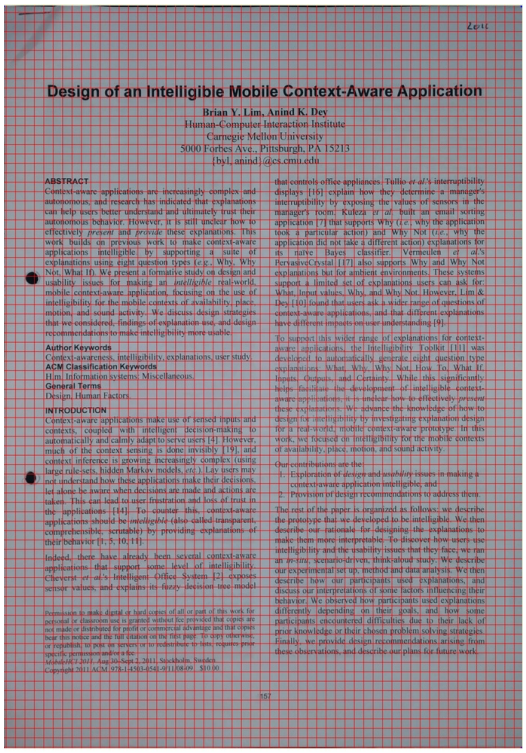


Figure 5.11: The contours of the image of a document.

Because the papers that are being used in the office environment are mostly black and white and only these parts are interesting to create the

URI for this document it is enough to only use the black-and white version of the image. Because a picture is never entirely black and white because the white always has a light taint of grey, it is more interesting to convert all pixel values below a certain threshold to white and all pixel values above that threshold to black. In Figure 5.11 an image can be seen of what the image would look like when converted from its original colors to a black and white version. This reduced the size of what needs to be stored to keep a representation of the document in the database. In a black and white environment there is only one digit needed to store the colours in stead of a 3 digit number varying from 0 to 255. Because the conversion from to total range of white to black to only to pure black and white the colors can be stored as ones or zeros.


Design of an Intelligible Mobile Context-Aware Application

Brian Y. Lim, Anind K. Dey
Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
{byl, anind}@cs.cmu.edu

ABSTRACT
Context-aware applications are increasingly complex and autonomous, and research has indicated that explanations can help users better understand and ultimately trust their autonomous behavior. However, it is still unclear how to effectively *present* and *provide* these explanations. This work builds on previous work to make context-aware applications intelligible by supporting a suite of explanations using eight question types (e.g., Why, Why Not, What If). We present a formative study on design and usability issues for making an *intelligible*, real-world, mobile context-aware application, focusing on the use of intelligibility for the mobile contexts of availability, place, motion, and sound activity. We discuss design strategies that we considered, findings of explanation use, and design recommendations to make intelligibility more usable.

Author Keywords
Context-awareness, intelligibility, explanations, user study.

ACM Classification Keywords
H.m. Information systems: Miscellaneous.

General Terms
Design, Human Factors.

INTRODUCTION
Context-aware applications make use of sensed inputs and contexts, coupled with intelligent decision-making to automatically and calmly adapt to serve users [4]. However, much of the context sensing is done invisibly [19], and context inference is growing increasingly complex (using large rule-sets, hidden Markov models, etc.). Lay users may not understand how these applications make their decisions, let alone be aware when decisions are made and actions are taken. This can lead to user frustration and loss of trust in the applications [14]. To counter this, context-aware applications should be *intelligible* (also called transparent, comprehensible, scrutable) by providing explanations of their behavior [1, 5, 10, 11].

Indeed, there have already been several context-aware applications that support some level of intelligibility. Cheverst *et al.*'s Intelligent Office System [2] exposes sensor values, and explains its fuzzy decision-tree model that controls office appliances. Tullio *et al.*'s interruptibility displays [16] explain how they determine a manager's interruptibility by exposing the values of sensors in the manager's room. Kulesz *et al.* built an email sorting application [7] that supports Why (*i.e.*, why the application took a particular action) and Why Not (*i.e.*, why the application did not take a different action) explanations for its naïve Bayes classifier. Vermeulen *et al.*'s PervasiveCrystal [17] also supports Why and Why Not explanations but for ambient environments. These systems support a limited set of explanations users can ask for: What, Input values, Why, and Why Not. However, Lim & Dey [10] found that users ask a wider range of questions of context-aware applications, and that different explanations have different impacts on user understanding [9].

To support this wider range of explanations for context-aware applications, the Intelligibility Toolkit [11] was developed to automatically generate eight question type explanations: What, Why, Why Not, How To, What If, Inputs, Outputs, and Certainty. While this significantly helps facilitate the development of intelligible context-aware applications, it is unclear how to effectively *present* these explanations. We advance the knowledge of how to design for intelligibility by investigating explanation design for a real-world, mobile context-aware prototype. In this work, we focused on intelligibility for the mobile contexts of availability, place, motion, and sound activity.

Our contributions are the:
1. Exploration of *design* and *usability* issues in making a context-aware application intelligible, and
2. Provision of design recommendations to address them.

The rest of the paper is organized as follows: we describe the prototype that we developed to be intelligible. We then describe our rationale for designing the explanations to make them more interpretable. To discover how users use intelligibility and the usability issues that they face, we ran an *in-situ*, scenario-driven, think-aloud study. We describe our experimental set up, method and data analysis. We then describe how our participants used explanations, and discuss our interpretations of some factors influencing their behavior. We observed how participants used explanations differently depending on their goals, and how some participants encountered difficulties due to their lack of prior knowledge or their chosen problem-solving strategies. Finally, we provide design recommendations arising from these observations, and describe our plans for future work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
UbiComp'09, Sept.30–Sept.2, 2011, Stockholm, Sweden.
Copyright 2011 ACM 978-1-4503-0541-9/11/09.09 ... $10.00.

Figure 5.12: A grid placed over the image of a document.

Because storing a number for every pixel in the image can still be a lot of data to keep in a database, it is more interesting to keep a more compressed form. To do so the image can be divided in a grid, as can be seen in Figure 5.12. For every little square in the grid the average value of the pixel colours can be computed. This average value for each square within the grid can then be kept in a matrix, which then can be stored in

the database as URI. The smaller the square the more detail that has been stored, but on the other hand, the more data is needed to store the URI and the more computing that is needed to compare multiple URIs.

Storing this matrix in OC2 as URI means that there is a fairly easy and fast way to compare two URIs. This can be done by some easy matrix comparison. This comparing will be explained more elaborately in the comparing section later in this chapter.

5.2.4 RFID Tags

Because there is need to track every kind of organisational structure there is need for a recognising mechanism that does not make use of an image of the document. When folders need to be tracked, taking an image of the front of the folder will be of no help to determine a unique URI identifying the folder. Therefore, there is need for an identifier that does not determine the URI based on looks. This means that the folder needs to be manually provided with an identifier.

A popular and easy kind of identifier is an RFID tag. These tags do not need direct sighting to be scanned and thus can be hidden inside a folder instead of on the outside. As discussed in the related work chapter using an RFID tag has some positive and negative side effects. The most important negative one is that these tags need to be linked to the system manually. This results in an overhead to keep these ids in the system. Additionally, this keeping up to date requires the need of an interface to do this. On the other side, the use of an RFID tag brings some nice advantages as well. The chances on errors are very slim. Whenever a tag has been read, the resulting URI will be correct. There are nearly no errors possible on the reading of a tag. Once a tag has been read the resulting id will be the correct id.

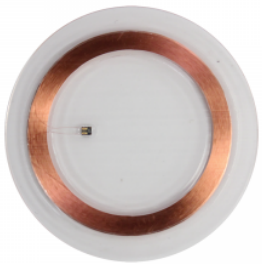


Figure 5.13: An image of a RFID tag.

An interesting way to place RFID tags in folders can be RFID stickers, as illustrated in Figure 5.13. These stickers can be placed discretely in the folder and be read by any kind of reader placed in the office. There could be readers placed on the shelf where the folder is normally kept, or placed under the working station of a user, so it can be tracked which folder is placed upon his desk.

5.2.5 TUIO Tags

Where the advantage of using RFID tags is the low chance of errors, the overhead of needing to place the tags and keep a digital representation is a downside. To find some middle-ground there is the option of barcodes. In this case a fiducial marker, this because fiducial markers are better to use as a moving target. This ensures that there is no need to actively scan the barcode, but it is enough to just pass the document, or folder, provided with a barcode over a camera which can register the tag. The solely overhead that is still present is that the barcode needs to be placed upon the document or folder. Even though, as discussed in the related work chapter, an easy program could be written that whenever a document is sent to the printer a tag gets added to the document and automatically printed. The only thing left then is to keep the link between the physical representation of the document and its digital counterpart stored in OC2

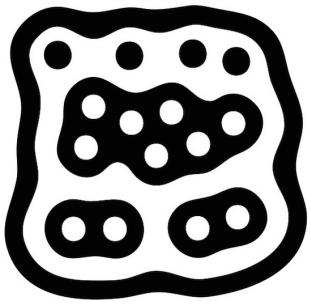


Figure 5.14: A fiducial marker

One of the possible fiducial tags that can be used are TUIO markers, which can be seen in Figure 5.14. These tags have been developed by reacTIVision [28] and can be used on moving objects.

5.3 Comparing

Having every document stored as a single URI and having a way to retrieve these URIs once again opens up the possibility to track documents around the office environment. Every time a document is seen the URI of this document can be identified. Using this URI, the digital representation of the document can be found inside the OC2 database.

But, every time a document is seen and its URI is determined there is still need for a way to know if this URI is the same URI as one that has been seen before. Based on the way the URI is determined, the tracking technique that is used is known and thus all other URIs ever stored by this Tracking Technique his recogniser can be loaded. Leaves the task to compare the newly created URI with all URIs that ever have been stored by this tracking technique.

This comparing is sometimes very unique for the URI that has been stored. For every recogniser that is used a comparing technique is needed. Even though, some techniques can be used for more than one recogniser his URIs, there will be multiple comparing programs needed to be able to compare every kind of URI stored.

5.3.1 OCR

In the previous section it was discussed how OCR can extract the title from the image of a document. A downside of OCR is that it is not 100 percent trustworthy and chances are real that some errors will occur. For example, the difference between the capital 'i' and the letter 'l' are very hard to identify for OCR. This kind of errors are quite normal, but still there is need for a way to cope with these. This means that two strings should be called equal, even though they are not entirely equal. To allow a certain number of mistakes there is need for a mechanism to test this. An easy solution is to just compare each character of the string with the corresponding character in the other string, but this would lead to bad results when, for example, one string misses a character. If one string would miss a character, this would result that both strings could be equal, except for the one string that is missing the character but the system concludes that they are entirely different because the corresponding characters are not matching. This means that there is need for a mechanism that not only checks character per character but takes things into account like missing characters. One solution for this is to compute the edit distance between the two strings.

Levenshtein Distance

In Computer Science the edit distance is a way of qualifying how dissimilar two strings are. This distance is measured by counting the minimum operations needed to transform one string into the other. The edit distance finds its applications especially in natural language processing. A well known example of the usage of the edit distance is in spelling correctors, where, whenever a word is not recognized, the words in its dictionary with the smallest edit distance are suggested.

One way to compute the edit distance is computing the Levensthtein distance. The Levensthtein distance between two words is the minimum amount of single-character edits (i.e. insertions, deletions or substitutions) that are needed to change one string into the other. It is named after Vladimir Levenshtein, who considered this distance in 1965. The Levenshtein distance may also be referred as edit distance.

As an example of how the Levenshtein distance is computed, the working of the Levenshtein distance will be illustrated. In this example the Levenshtein distance between "kitten" and 'sitting" will be computed. This computation consists out of a number of steps:

1. **k**itten → **s**itten (substitution of "s" for "k")
2. sitt**e**n → sitt**i**n (substitution of "i" for "e")
3. sittin → sittin**g** (insertion of "g" at the end)

As there are three steps taken to go from "kitten" to "sitting" it can be concluded that the Levenshtein distance between both strings is three.

The formal definition of the Levenshtein distance is as follows:

The Levenshtein distance between two string a, b is given by $lev_{a,b}(|a|, |b|)$ where

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & if\, min(\text{i,j}) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_i)} \end{cases} & otherwise. \end{cases}$$

The Levenshtein distance has several simple upper and lower bounds. These include:

- It is always at least the difference of the sizes of the two strings
- It is at most the length of the longer string
- It is zero if and only if the strings are equal
- If the strings are the same size, the Hamming distance is an upper bound on the Levenshtein distance
- The Levenshtein distance between two strings is no greater than the sum of their Levenshtein distances from a third string.

To compute the Levenshtein distance in Java the following code can be used. Note that the $minimum(int, int, int)$ function is just a function that combines the $Math.min(int, int)$ two times and returns the minimum of three integers.

```
public static int computeLevenshteinDistance(String str1, String str2) {
 int[][] distance = new int[str1.length() + 1][str2.length() + 1];
 for (int i = 0; i <= str1.length(); i++)
  distance[i][0] = i;
 for (int j = 1; j <= str2.length(); j++)
  distance[0][j] = j;
 for (int i = 1; i <= str1.length(); i++)
  for (int j = 1; j <= str2.length(); j++)
   distance[i][j] = minimum(
    distance[i - 1][j] + 1,
    distance[i][j - 1] + 1,
    distance[i - 1][j - 1]
     + ((str1.charAt(i - 1) == str2.charAt(j - 1)) ? 0 : 1));
 return distance[str1.length()][str2.length()];
}
```

Based on the Levenshtein distance between two strings, a fair assumption can be made whether or not two strings are equal. After testing several boundaries, a fair boundary to say that two strings are equal is when their Levenshtein distance is smaller than 10. In future work this parameter could be changed according to the length of the title.

5.3.2 SIFT

To do comparison using the SIFT algorithm, the harder part was already implemented in the identifier. The comparator gets a key-file from the current image and a list of key-files containing all key-files of the other images tracked by this tracking technique. To compare both images, the key-files of each image should be provided to the system. In this particular example the

SIFT implementation of David Love[5] has been used. This version of SIFT can be run using an executable through the command line. For every image it generates a .pgm file, containing the image with the features shown and a .key file containing the keys. If two files need to be compared the following statement can be used:

```
match -im1 book.pgm -k1 book.key -im2 scene.pgm -k2 scene.key > out.pgm
```

Here it can be seen that, to compare two images, four parameters are needed. The first parameter is the image in .pgm format of the first image, the second is its key file. In this case these are the book.pgm and book.key files. The other two parameters are for the second image, and are the image in .pmg format and the key-file in .key format. In this case these are the scene.pgm and scene.key files. The result is presented in the out.pgm file. This is a file containing both images and lines drawn between the matching key-points, as can be seen in Figure 5.15.

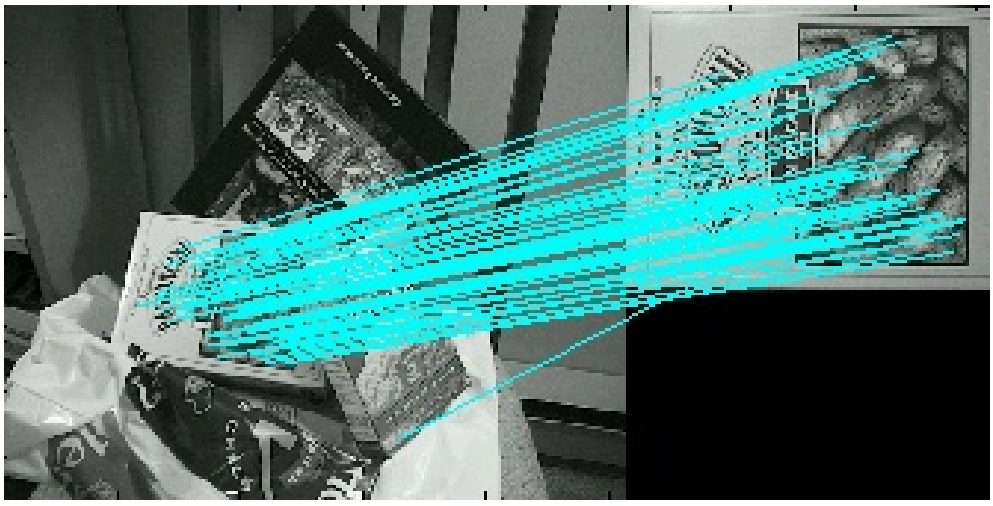


Figure 5.15: Links between matching key images

In this particular set-up there is no need for this image. The return statement that is given in the command line is enough to determine whether or not two images are equal.

[5]http://www.cs.ubc.ca/ lowe/keypoints/

5.3.3 Colour Intensity

When the colour intensity recogniser has been used, then the saved URIs in the framework are matrices. The size of these matrices is dependable on the settings of recogniser, but should for every instance of this recogniser be the same. This means that the comparator for colour intensity gets matrices which are all of the same size containing integers, which are the averages of the colour intensity of the document. To decide whether or not two matrices are equal some matrix comparison needs to be done. As with every comparator it will receive the matrix that has just been determined of the newly seen document and a list of matrices being all matrices ever determined by this tracking technique.

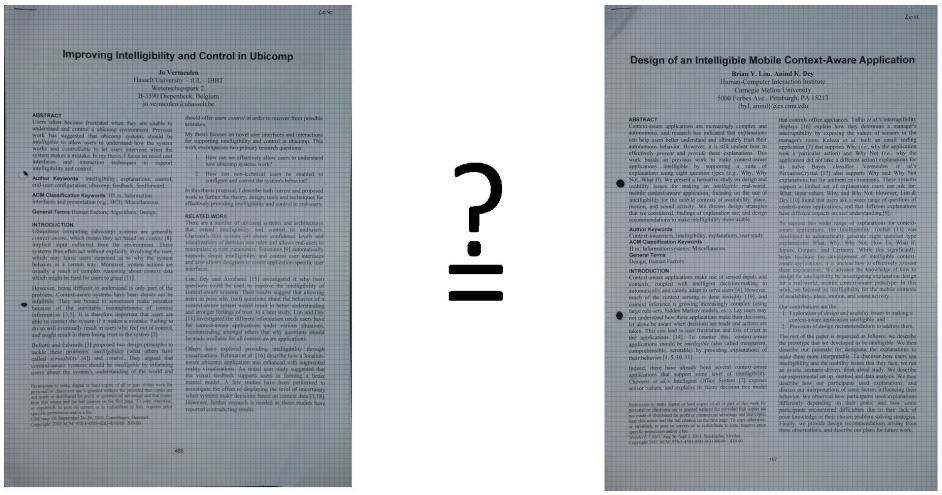


Figure 5.16: Are these two documents equal?

To know if two matrices are equal, there is need to compare them cell by cell and see if their respectively content is equal. In this case however this is not entirely true. Because there are certain elements that can make the averages of the same document diverse a bit. These reasons can be that the squares on the image do not exactly map on the same location on the document itself, the light on the moment the first picture was taken was different than when the second picture was taken or sometimes there is shadow on the picture. This is why a threshold should be considered. Whenever the value of one cell of the matrix differs less than a certain threshold than the corresponding cell in an other matrix they can be called equal.

While comparing the matrices an integer will be kept saying how many mismatches have been seen between both matrices. Also here a certain tolerance for errors should be taken into account. This in case, that there is a big

shadow over the paper or there was a small object placed on the document. To prohibit that this would result in a mismatch it is opportune to also take a certain threshold for errors in mind here.

5.3.4 String Comparators

One of the easiest comparators is the String Comparator. In contrary to the OCR comparator the strings that need to be compared need to be exactly the same before it can be called a match. This comparator can be used for when the recognising technique returns an id as URI. In the tracking techniques discussed above this is the case with the use of RFID tags and TUIO tags. Both these return always exactly the same string when they have been detected. It would be non-logical if a slight error would be allowed, because an id that differs only even one digit from another id can be a totally different entity.

In contrary to the other comparators, the String Comparator does not really need any other program or piece of software to work. Where SIFT has a piece of software doing the comparing, OCR requires some additional applying of the Levenshtein distance on the result and the colour intensity matrix need a whole matrix comparison with some error thresholds the String Comparator can use the built in string comparator of the language that is being used.

5.4 Use Cases: Tracking in Action

Now that there some tracking techniques in place, it is time to leave them out in the open. Let them have some fun and show the world what they are capable of. Therefore, there are really practical set-ups made to show how the tracking technique work. For each of these set-ups it is explained what their use is, what tracking techniques they are using and which calls that they make made to the framework to store information.

The information that is returned by the tracking devices is sometimes not enough. This especially when it comes to decide which organisational structure that is being used. To decide which organisational structure that is being used there are some extra devices in place. These devices will be explained more elaborately whenever they are being encountered during the explanation of one of the use-cases. Besides the extra hardware that is needed to decide which organisational structure that is used, there is one other pre-requisite that is used. The prerequisite that is used here is that everything within the use cases happens in a closed world. This closed world assumption

makes it easier to decide which document has been moved to another place. When one place sees the departure of a document and another tracking technique sees the arrival of a document, but cannot decide which document this is, it can be assumed this document will be the one that left the first place.

Because, even in a closed world, more than one document can be not located it is opportune to have a way to still keep track of these documents. Whenever a document has been seen at a location and is known to be removed from it, for example, if there is a pile of documents with document A on top of the pile and the next time the same pile is spotted document B is on top, while it was known that document B lied underneath document A, it can be assumed that document A has been removed from the pile. The knowledge that this document has been removed and it is not sure where it is left at this precise moment should be kept. For this a pool of 'un-located' documents could be used. Whenever the system is insecure of the location of a document it can be placed in this pool. Whenever a tracking device picks up that a document is placed in a organisational structure but it was unable to identify this document the system could pick the document which is placed in the pool. When multiple documents are in the pool, the user should be offered the choice between all documents in the pool. This choice could be guided or eventually even automated based on the current context in which the user is working. Based on the current context the document in the pool which relates to this context the most could be assumed as being the document that has been located.

For every tracking technique that is installed, there is some additional information that needs to be kept. This information is the organisational structure that is being tracked by this tracking technique and the tracking technique his id within the system. The reason that the id of the organisational structure of the tracking technique needs to be saved is because it is opportune to keep a as low coupling as possible between tracking techniques and structure. When a tracking technique sees a new document, it just calls an endpoint adding a document to an organisational structure. For this call the id of the organisational structure and the id of the document are needed. The id of the document is determined by the tracking technique but the id of the structure it tracks needs to be kept. This could, later on be moved to the system itself, but this would result in a higher coupling between the tracking technique and the organisational structure, which may make certain extensions, like using one tracking technique for multiple organisational structures, impossible. When the tracking technique is responsible to provide the correct id of the organisational structure, it can be made smart enough to pick the right one out of multiple possibilities.

The reason the framework his id of the tracking technique is kept in the tracking technique itself is to be able to cope with failures of the tracking technique. Whenever a new tracking technique comes online it says this to the framework and the framework than provides a new tracking technique id for the tracking technique. But what if the tracking technique was already registered in the framework and already had an id within the framework? There is no possible way for the framework to see whether a tracking technique requesting an id is new or a tracking technique that went offline. To cope with this, a tracking technique that already has an id within the system needs to provide this id whenever it comes back online after a failure.

5.4.1 Piles

A first use-case that is discussed are piles. To track piles there are multiple tracking techniques that could work. In this case we will only focus on tracking techniques that do not need any additional information. Yet again, there are multiple possibilities here. Above each pile that needs to be tracked a camera should be placed, as illustrated in Figure 5.17. As explained before this camera should take a snapshot on every interesting moment and crop an image of the document. Starting from this image of the document on top of the pile, there are still a lot of possibilities of how to track the documents. With the implemented plug-ins and the requirements defined above there are still three ways to identify a document. The first is making use of the SIFT algorithm, the second is making use of OCR on the title and the third is making use of colour intensity of the document.

In Figure 5.17 the set-up of the pile can be seen. There is a camera, in this case a Samsung Galaxy Cam 2 as discussed in section 5.1.2. Above it a Kinect can be seen to track if there is movement above the pile as discussed in Section 5.1.1. In this case there should be a file kept keeping what organisational structure, pile, is being tracked and the id by which the tracking technique is stored in the framework. The idea is that the tracking technique starts whenever a new image, sent by the camera, is received. Whenever a new image is received, the recognising component will determine a new URI for this document, specified for this tracker. This URI is stored in the framework. When this URI is stored the recogniser requests all URIs of the documents tracked by this tracking technique. There is an endpoint in the tracking component of the framework returning all URIs that are being tracked by a tracking technique. This is why the id of the tracking technique needs to be stored. When the framework returns the URIs, the recogniser will feed these together with the new determined URI to the comparator. After comparing, the comparator will return an array with all ids of the

matches for this document. The document could be seen more than once, so more than one match could be returned.

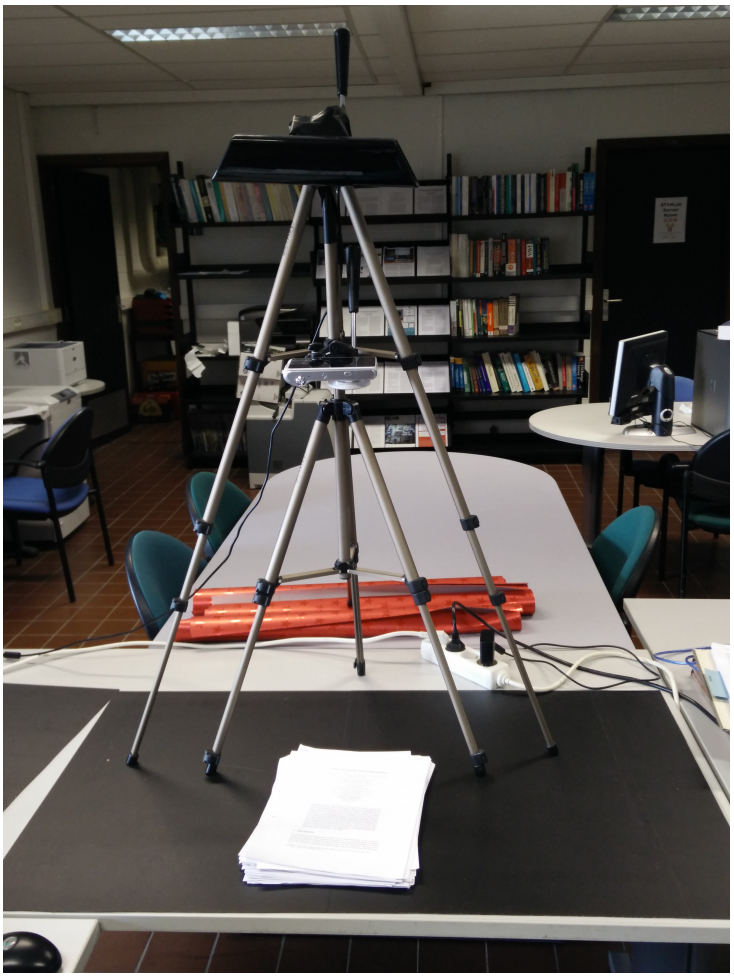


Figure 5.17: Set-up: Pile

When no matches are found by the system there is a problem. As everything is happening in closed world, the document should always be found. It could happen that there is no match found. To still be able to determine which document is being seen the pool of un-located documents is presented to the user, this so the user can choose from the pool which document is seen now. Whenever a match is found, the information about this document should be updated in the system. This means that the current context needs to be added to the document and the location of the document needs to be updated to the current location. In Figure 5.18 there is an overview of the calls that happen when a document is tracked by the system.

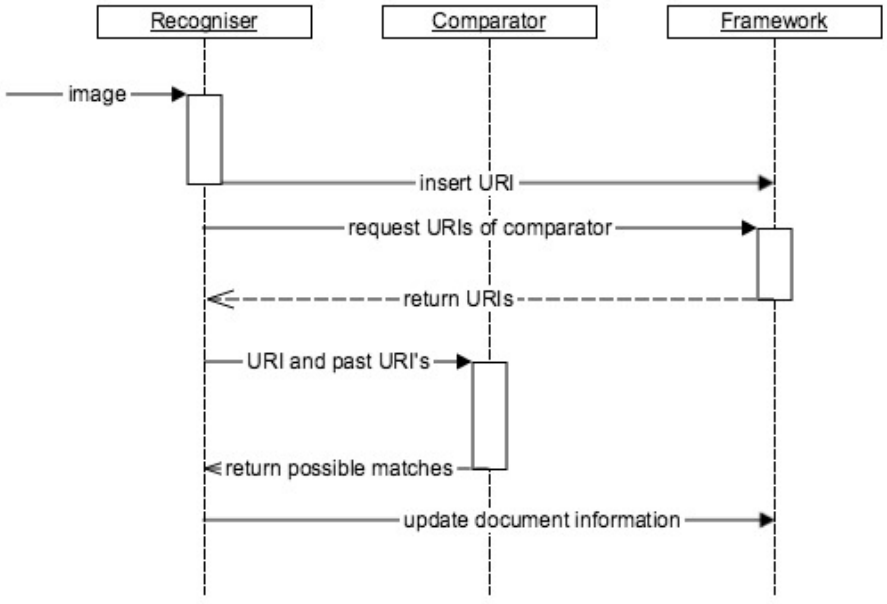


Figure 5.18: Control flow when tracking a pile

5.4.2 Paper Trays

For this example we will make use from 5 paper trays. The documents that will be stored in the paper trays will be equipped with Fiducial tags for identification of the documents. To know in which tray the document has been placed the height will be measured, by which can be determined in which tray a document has been placed. In Figure 5.19 an overview can be seen of the set-up. These paper trays are equipped with LEDs and buttons on each tray, but they are for the re-finding of the documents. For the tracking component there is a sonar in front of the trays and a camera pointed up to see the bottom of the documents inserted in the trays.

To determine in which tray a document is being placed it is measured how high the document is being held. To do so, a sonar is mounted in front of the paper tray. This sonar calculates distance. Whenever the measured distance is smaller than a certain threshold it can be assumed that there is something seen. Normally the sonar should be able to measure from where it

is placed until the ceiling. The sonar will return the distance between itself and the document. This makes it possible to calculate in front of which tray the document is being held. A paper tray is approximately 7 cm high. If the distance between the bottom of the paper tray and the sonar meter is 5 cm it can be assumed that, when the sonar records a distance between 5 cm and 12 cm the document is held in front of the lowest tray. When this distance is between 12 cm and 19 cm it can be assumed the document is held in front of the second tray. This can be repeated for every tray and an almost unlimited amount of trays. Using this method it is possible to determine in which tray a document has been placed.

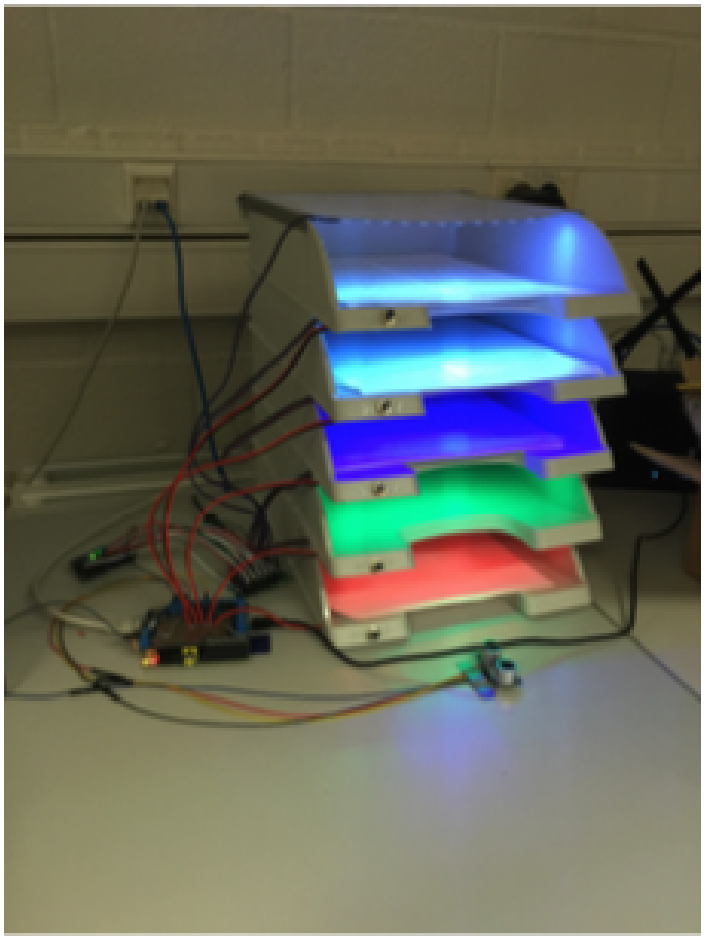


Figure 5.19: Set-up: Paper Tray

Whenever the sonar records a document, it can be assumed that the camera will be able to scan a TUIO tag as well. Both the camera and sonar can independently determine when a document is seen and they do not need to trigger each other. It is good practise to couple these and check the height of the sonar whenever the camera sees a new TUIO tag. This way it can be assumed that the document is nearly in its tray and is at the right height.

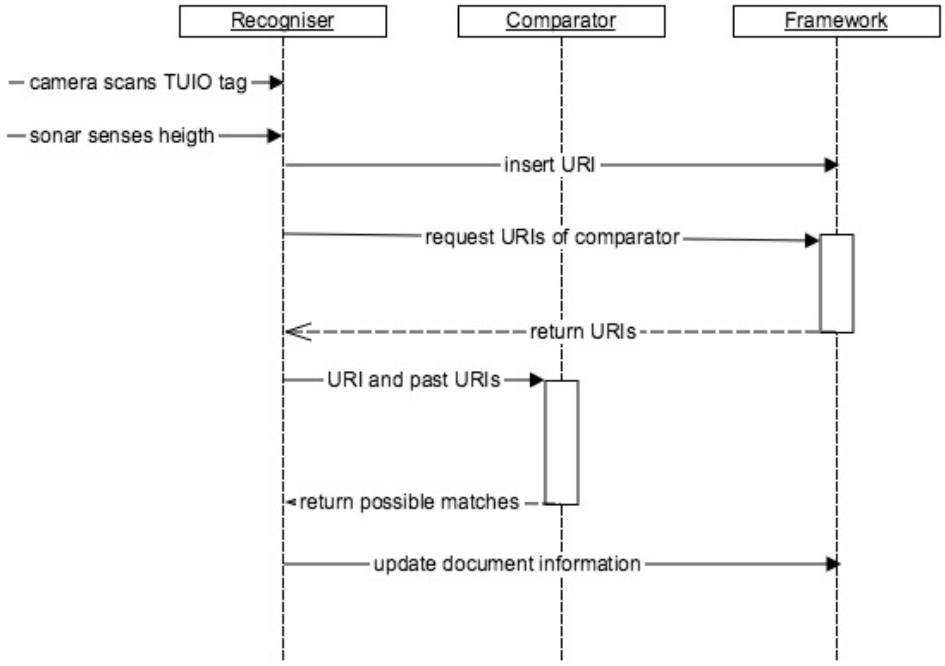


Figure 5.20: Control flow when tracking a paper tray

The sequence that happens between the recogniser, comparator and framework are basically the same as with the pile. First, the identifier gets the id that has been read by the camera. Next to this, the sonar will provide the id of the tray. As with the pile there is a file kept in the tracking technique storing the id of the tracking technique and the ids of the trays. Even though the sonar can return the tray that has been used, this is only relative to the paper tray, being the bottom one, second one, or any other, but not the id by which the tray has been stored in the framework. Therefore, the file needs to contain a mapping between each tray and the id that represents them in the

framework. Having the id of the tray and the URI for the new document, the same steps can be taken as were used with the pile. First, the URI is inserted in the framework, this related to the tracking technique id. Next, all URIs of documents tracked by the same tracking technique are requested. Based on these URIs and the URI of the new document the comparator can do its work. After the comparator returned the matches the information about the document can be updated within the framework.

5.4.3 Ring Binders

A third example are ring binders. These are a bit double, because the ring binder itself is part of a file structure, each file is labelled and binders are ordered. But inside the ring binder there could be another organisational structure that could be tracked.

Figure 5.21: Set-up: Ring Binders

The idea with the ring binders is that every ring binder is equipped with an RFID tag. The working area on the desk is provided with an RFID reader which can alert the system every time a new ring binder is placed upon the desk. The placement of a new ring binder on the desk can on his turn notify the system that tracks the organisational structure inside the ring binder. This way there is no need for extra devices like a Kinect to track when this organisational structure needs to be tracked. The difficulty that this brings along is that there is no clear mapping between the organisational structure that is being tracked by the tracking technique and the tracking technique. To overcome this problem there is a need to keep a mapping between the contents of a ring binder and the organisational structure inside the ring binder. This mapping is stored in the framework itself, as every organisational structure can contain another organisational structure. This means that by identifying the ring binder it is instantly known which organisational structure is contained by it, and so if there is any action in this organisational structure it can be handled.

In Figure 5.21 there is an image of the folders that are being used. The RFID tag is not visible here because it will be placed inside the ring binder. The LED-screen is attached to help the user re-find their documents.

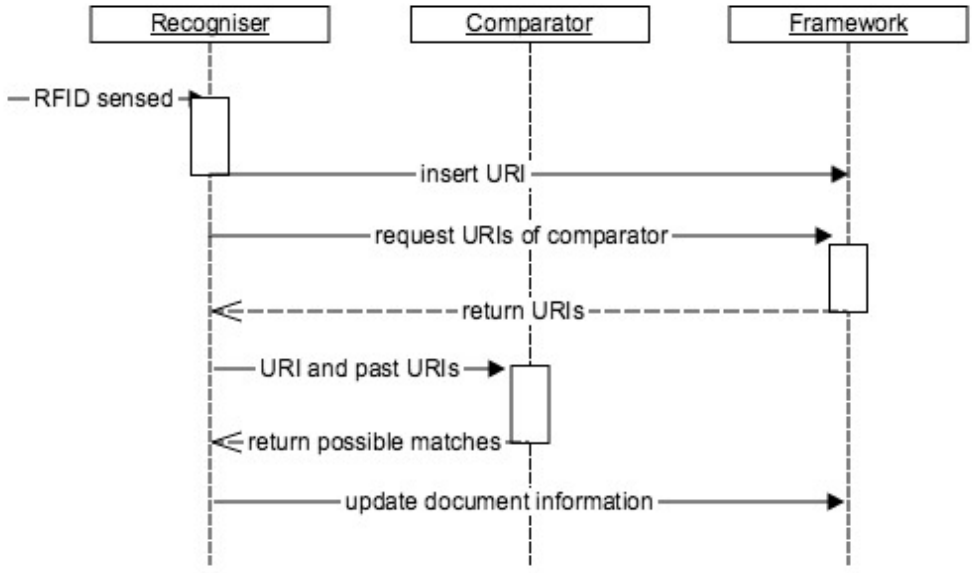


Figure 5.22: Cotrol flow when tracking ring binders

Even though in first instance there is no need to record whenever anything happens to track the inner organisational structure the time the folder is open there is a lot that can happen inside it. Therefore it still is needed to keep track of this organisational strategy separately. To do so any other system to do this tracking can be used. For example there could be a pile inside the folder that needs to be tracked. The only difference between the tracking of the pile explained previously and the pile that is being tracked now is that this system can track multiple piles, every time another folder is placed upon the desk. Therefore the id of the organisational structure that is being tracked no longer needs to be kept in the system, but, as explained previously, can be determined by the id of the current folder placed on the workspace.

In Figure 5.22 it can be seen how the communication between the identifiers, comparator and framework happens. The first intensive is the scanning of an RFID tag. This could on his turn trigger a tracking technique for the enclosed organisational strategy. To determine which ring binder is being tracked the framework returns all ids tracked by this identifier and the comparator can decide which ring binder is placed upon the desk.

5.4.4 File Cabinet Drawer

As a last use-case a file cabinet drawer will be discussed. This is a drawer in a file cabinet filled with folders. These folders are all labelled and obviously are placed in a certain order. This means that the content of this cabinet drawer is a file structure. In Figure 5.23 an image of the cabinet drawer can be seen. Every file in the drawer is equipped with a buttons and a led. The leds are purely for re-finding purposes, but the buttons do have their use with the tracking.

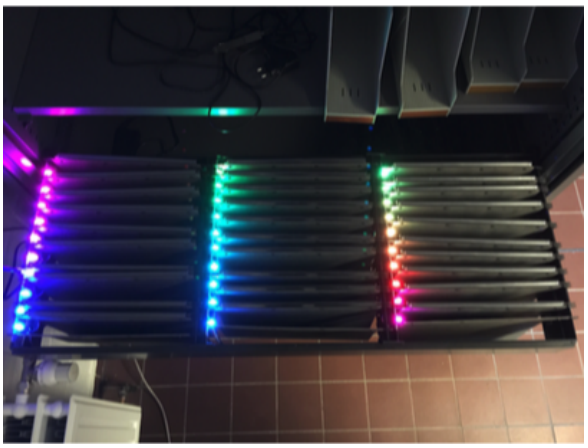


Figure 5.23: Set-up: File Drawer

In Figure 5.24 there is an overview of the calls that are being made by system. Here it can be seen that the system gets activated whenever a buttons is pressed in the file cabinet drawer. When this happens the user needs to select the current document from the pool with un-located documents. This clearly makes optimal use of the fact that we work in a closed world environment. When the system would not be in a closed world, it could happen that a new document is inserted in one of the files only based on the id of the file in which the new document is put. There is no way to know which document this is. Hence the reason the pool is used and the closed world assumption is needed. When the system would be used in an environment that is not a closed world there would be need for a way to identify this document. There are still a wide variety of possibilities to do so. The same system as with the paper trays could be used and a camera could be mounted above the file drawer reading a tag on the documents, or, like with the ring binders every document could be provided with an RFID tag and a RFID reader could be mounted inside the drawer. Because this are only use-cases and the users of the system could later choose to not built on the closed world assumption and implement one of the mentioned alternatives, or their own alternative suiting their needs even better.

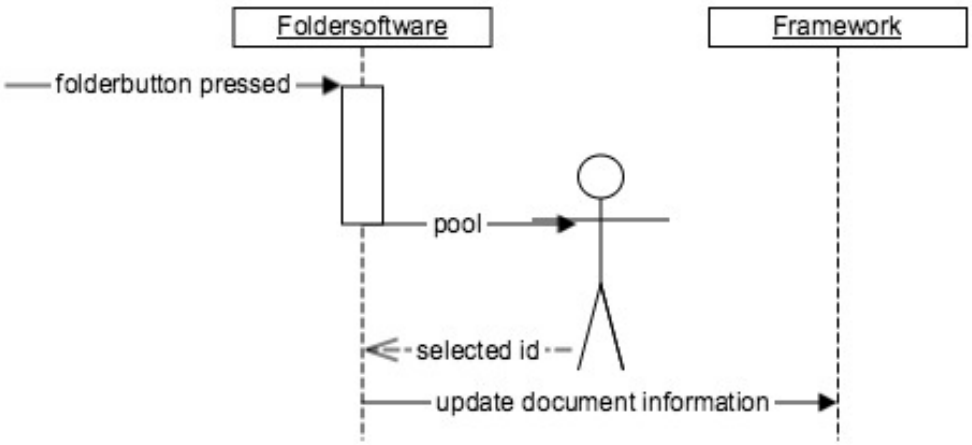


Figure 5.24: Control flow when tracking a file drawer

In the overview of the calls that are being made, illustrated in Figure 5.24, the difference between the previous use-case is quite clear. This use-case makes no use of an recogniser or comparator. Even though, like in the previous use-cases there is need for a file keeping a mapping between the id of the files in the framework and the buttons in the drawer.

6 Conclusions and Future Work

In this last chapter we discuss the outcome the thesis and present possible future work. First, a short reminder of the goal of this thesis is given and later we briefly explain how this goal has been achieved. Together with this short summary of the delivered work a short reflection about the delivered work is given.

In the related work section, the focus was on any work that had to do with possible ways to deduct a unique identifier from an organisational strategy. This was divided into two categories, where one needs extra hardware to determine this id, and the others can just determine an identifier using an image of the document. Additionally, there was an overview of how current PIM systems keep track of physical documents in the office environment. Based on the knowledge of this related work, the research goals and requirements could be determined.

6.1 Discussion

The main goal of this thesis was to offer a solution for the fact that in current work there was no relation between the kind of structure that is being tracked and the digital representation. The idea behind integrating the structures in the digital representation opens up a number of ways to interact with these structures that are being tracked. The best example of this is, that based

on the structures the re-finding of documents can be refined. Every type of organisational structure has certain cues. Based on these cues specific re-finding mechanisms can be used. In Table 6.1 the overview of the queues in relation to the use cases presented in previous chapter can be seen. It is clear that based on the type of the organisational structures, different cues are being used, and based on these cues different kinds of re-finding mechanisms could be implemented.

		Context Cue	Spatial Cue	Time Cue
Filing	Files in a file drawer	x		x
	Ring binders on a shelf	x		x
Mixing	Paper trays	x		x
	Ring binder on its own	x		
Piling	Piles on a desk	x	x	

Table 6.1: An overview of which re-finding mechanisms could be implemented

To be able to store the organisational structures in a digital representation, there was need for a place to store this organisational structure. This means that the underlying system that is being used needs to offer the opportunity to store these organisational structures. OC2 was used as an underlying system. This because it was developed at the WISE lab and there was a good understanding of how this system works. Even though OC2 did not yet support storage of organisational structures, the knowledge about the system opened up the possibility to make use of OC2 as an underlying system where, by the use of a mapping these organisational structures could be integrated. This mapping saved the need of developing an entire new PIM system to store the information about documents. Not only did it save this project the development of an entire new PIM system, it also extended the existing tool that was developed at the lab, so it had a wider range of use.

Next to the mapping on top of the OC2 system still some extra functionality was needed to be able to track these structures. This mapping not only enabled the possibility to keep track of organisational structures, additionally some extra features were implemented to store extra information about the document that is being tracked. One of the key features of this extra information are lifelines. These lifelines can keep track of the interactions done with the documents and the locations the document has been seen. This extra information can on their turn be used in extra applications.

Next to the place to store this information about the documents, there was need for a way to get this information in this system. Therefore, there are some plug-ins implemented to do this tracking. The developed plug-ins

are working as a proof of concept. The goal of the framework was to be extensible and to open up the possibility for users to create their own plug-ins to track documents the way they want to. These tracking techniques are consisting out of two parts, one part is to recognise the organisational structure, and determine a URI that can be kept inside the framework. The other part is to compare these URIs and determine whether or not these are the same. Both are needed to do some decent tracking, but there is no need for high coupling between both parts. One recogniser could make use of multiple comparators, which might not be very useful, but the other way around, multiple recognisers could make use of the same comparators. This can save developers a lot of work. They can take a look at which comparators are already existing and based on this they can decide for their recognising component to take this into account and return a URI fit for a particular comparator to do the comparing. This can save developers a lot of time and effort and even better, a component they can use to test their recogniser.

6.2 Evaluation

Based on the fact that the framework can only show its true potential by the use of plug-ins, there is no clear way to evaluate the use of the framework. The framework's dependency on the plug-ins to be evaluated only leaves the possibility to or evaluate the framework in combination with the plug-ins, or evaluate individual plug-ins. As the provided plug-ins only function as a proof of concept and the plug-ins are in fact more the responsibility of the user. Evaluating these would have no contribution. These plug-ins could be developed by anyone and run in any place on the world. As the correctness of plug-ins cannot be guaranteed all the time, there is no use to evaluate the plug-ins that were developed together with the framework. The correctness of these plug-ins have no clear contribution to the project, as they are only a very small base of what can be done with the framework.

The fact that the plug-ins are undependable makes them a liability to the system. The fact that the correctness and the availability of all plug-ins are not the responsibility of the framework, or the maintainer of the framework, creates this kind of liability. There is a chance that these third-party plug-ins do not deliver what is expected. Recognisers can return inconsistent URIs or are not able to determine URIs when they should be able to. Comparators could not find matches, because they do not see matches, or they cannot handle an error margin that should have been provided. Next to the correctness, there is the matter of availability. As these plug-ins will not be running at the same location as the framework the responsibility of the availability of

the plug-in lies with the creator of the plug-in. These are drawbacks which are known issues to having a crowd invest in a project. These are choices made to offer the user a wide variety of use of the platform, and even better adjust it to their personal need, personal setting or personal hardware. This kind of extensibility comes at a price that needs to be paid.

6.3 Future Work

From the beginning, the idea behind the framework was to be extensible and leave it over to the users to extend it, make use of the framework, and this hopefully in ways it was not yet seen possible yet. Giving users this kind of freedom comes at a cost as explained in the previous section, but still it remains exciting to see what kind of possibilities giving the users this freedom will bring.

Next to the possibility to extend the system by creating some extra plug-ins there is still the possibility to extend the framework itself. The framework has been foreseen to store a of lot of information about the documents that are being stored, but there is still a chance that there is extra information that could be stored about the organisational structures. Even though the implementation of the lifelines offers a lot of possibilities to the user to store the information they want to keep, it can still be that there is some sort of information important enough to store in the database. Even though, the current extensions made to the framework were done with the use of them in the physical world in mind. The same framework can be used to track digital documents. The digital representation of these documents are subject of another thesis written at the WISE lab.

As stated before, the addition of lifelines opens up an extra range of possibilities. These possibilities can already be exploited, because they are accessible for users to use in any way they want, but still inside the framework itself some extra information could be kept. This is work that could be done by the people maintaining the framework and could not be left to the user.

Next to the storing of extra information there are still a wide variety of applications that could be developed using the information that is already being stored. The lifelines could be a great asset to develop extra applications, but also just the general storage of information about the documents can help the user to develop extra applications. One application that is already being developed was part of the re-finding module explained earlier in this document. This re-finding was part of another thesis at the WISE lab. Equally as in this project, first the framework needed to be developed and extended and later on some use cases for the re-finding of documents were

implemented. These use cases make use of the information that is being stored by the project which is subject of this thesis.

In general, this thesis can be seen as a part of something bigger. But still it has its own purpose and its own reason to live. Every other part of the framework together with extensions could be seen as future work from this tracking component. Besides the obvious extensions as re-finding and tracking of digital documents there is still a wide number of ways to expand the use. The stored information offers the possibility to create a digital representation of the physical documents. This information could be used to explore what documents are being kept, which documents belong together and which documents are relevant to the documents that are being used within a user his current context. Additionally the information in the lifelines offer a full history of the physical document. This history can be the source for an entire range of new applications. One interesting use of these lifelines is helping the user to store documents. Based on the locations the document has been kept and the time that it spent there the *default* location of a document could be determined. This would still be subject of a study on how one can determine the default location of a document. Is it based on the location the document has been seen the most times, reasoning that a document could be used everywhere but would always return its default location, and so there would be the most entries about this location on the documents lifeline. Another way of reasoning about the default location of a document is that there is no place like home. Using the timestamps that accompanies the location entries on the locations lifeline of a document it could be determined where the document spent most of its time. Using the times the document has spent at various locations could then suggest the location that can be seen as default for the document. A third possible important factor that is relevant to a document's default location is the kind of organisational structure that is being kept. As explained before, a file can be seen as a more permanent way of storing documents. Documents are often kept longer in folders than in other organisational structures. On the other hand are piles more active organisational structures. They are often placed upon a desk containing documents that are begin used quite often on an active project. Mixtures can be seen as an in-between organisational structure. They are often used to store documents which are not hot any more and are probably waiting to be stored away. Based on the the organisational structure it can be determined what the default location of the document is. The best of the three options given above will probably be a combination of the three. Based on where the document was seen the most and for the longest time it could be determined what the document's default location is.

Even better could this combination not only determine the default location, but also the next location of a document could be guessed. If next to the three parameters that just have been discussed a fourth parameter, being the context of a document would be taken into account, the future of a document could be predicted. Using the context of a document and other documents within the same context's history the future of a document could be predicted. When documents in the same context are being stored it could be predicted that the current document his next location will be a more permanent one as well. This could then be seen as a file containing documents within the same context. These are only a few examples of the endless possibilities that the framework brings.

6.4 Conclusion

This framework opens up a lot of possibilities. It has some reliabilities because there is a lot of responsibilities left to the users, but this leaves room for a lot of excitement. There is no way to predict what users will do with the provided work and one can only hope some interesting projects may flow out of it. One downside of the project is that the focus was on physical documents. Even though it has already been stated that they probably will not leave this world soon, their future is still debatable and insecure. Yet again it was very interesting working on this project. The fact that it could be or totally useless in a few years or be widely implemented in both the private as corporate sector leaves a lot of room for excitement for the future.

One of the main goals of the framework was to keep track of the way documents are being stored. The KOR framework can do this because every document is stored in an organisational structure. These organisational structures keep the relation between documents and store information about documents that can be used again by other applications.

Also the extensibility of the framework was important. A REST interface was foreseen on top of OC2, ReViTa and the tracking component. This gives the user the possibility to interact with the system in any possible way and it opens up the system for all sorts of extensions. Additionally the tracking component was designed in such a way that only a minimal amount of knowledge about the component is needed to be able to create new tracking techniques. By the use of only a few endpoints the users can create their own tracking techniques and integrate them in the system.

The invention of the lifelines for every document opened up the possibility for the user to store extra information, without having to interact with the system itself to much. Lifelines can be created easily and store every sort of

information the user wants to store. This makes sure that extra information about documents can be kept without that there are any changes to the framework needed.

Bibliography

[1] Anand Agarawala and Ravin Balakrishnan. Keepin' It Real: Pushing the Desktop Metaphor with Physics, Piles and the Pen. In *Proceedings of CHI 2006, Conference on Human Factors in Computer Systems*, pages 1283–1292, Montréal, Canada, April 2006.

[2] Joao Aires and Daniel Gonçalves. Personal Information Dashboard - Me, At a Glance. In *Proceedings of CSCW 2012, Conference on Computer Supported Cooperative Work*, pages 1–8, Seattle, USA, February 2012.

[3] Damián Arregui, Christer Fernstrom, François Pacull, Gilbert Rondeau, Jutta Willamowski, Elisabeth Crochon, and François Favre-Reguillon. Paper-based Communicating Objects in the Future Office. In *Proceedings of SOC 2003, Smart Object Conference*, Grenoble, France, 2003.

[4] John Ayoade. Roadmap to Solving Security and Privacy Concerns in RFID Systems. *Computer Law & Security Review*, 23(6):555–561, 2007.

[5] Mortaza S Bargh and Robert de Groote. Indoor Localization Based on Response Rate of Bluetooth Inquiries. In *Proceedings MobiCom 2008, Conference on Mobile Computing and Networking*, pages 49–54, San Francisco, USA, September 2008.

[6] Deborah Barreau. Context as a Factor in Personal Information Management Systems. *Journal of the American Society for Information Science*, 46(5):327–339, 1995.

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

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

[9] Christian Decker, Uwe Kubach, and Michael Beigl. Revealing the Retail Black Box by Interaction Sensing. In *Proceedings of ICDCS 2003, Conference on Distributed Computing Systems*, pages 328–333, Providence, USA, May 2003.

[10] Susy dHont. The Cutting Edge of RFID Technology and Applications for Manufacturing and Distribution. *Texas Instrument TIRIS*, 16:10–15, 2004.

[11] Hasoo Eun, Hoonjung Lee, and Heekuck Oh. Conditional privacy preserving security protocol for NFC applications. *Consumer Electronics, IEEE Transactions on*, 59(1):153–160, 2013.

[12] Simson L Garfinkel, Ari Juels, and Ravi Pappu. RFID Privacy: An Overview of Problems and Proposed Solutions. *IEEE Security & Privacy*, 3(3):34–43, 2005.

[13] Jim Gemmell, Aleks Aris, and Roger Lueder. Telling Stories with Mylifebits. In *Proceedings of ICME 2005, International Conference on Multimedia and Expo*, pages 1536–1539, Los Alamitos, USA, July 2005.

[14] Jim Gemmell, Gordon Bell, and Roger Lueder. MyLifeBits: a Personal Database for Everything. *Communications of the ACM*, 49(1):88–95, 2006.

[15] Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong. MyLifeBits: Fulfilling the Memex Vision. In *Proceedings of MULTIMEDIA 2002, Conference on Multimedia*, pages 235–238, Juan Les Pins, France, December 2002.

[16] Jim Gemmell, Roger Lueder, and Gordon Bell. The MyLifeBits Lifetime Store. In *Proceedings of SIGMM 2003, Workshop on Experiential Telepresence*, pages 80–83, Berkeley, California, October 2003.

[17] George Brown Goode, Charles Doolittle Walcott, Richard Rathbun, and William de Chastignier Ravenel. *Report on the Progress and Condition of the United States National Museum for the Year Ending June 30...* US Government Printing Office, 1897.

[18] Bae Hark, Jong Choi, and Choon Leem. A Database Design of RFID Document Management System with e-Ink Technology. In *Proceedings of NCM 2008, Conference on Networked Computing and Advanced Information Management*, pages 14–17, Gyeongju, Korea, September 2008.

[19] Steve Hinske. Determining the Position and Orientation of Multi-Tagged Objects Using RFID Technology. In *Proceedings of PerCom 2007, Conference on Pervasive Computing and Communications*, pages 377–381, New York, USA, March 2007.

[20] Steve Hinske and Marc Langheinrich. Using a Movable RFID Antenna to Automatically Determine the Position and Orientation of Objects on a Tabletop. In *Proceedings of EuroSSC 2008, Conference of Smart Sensing and Context*, pages 14–26, Zurich, Switzerland, October 2008.

[21] Matthew Jervis and Masood Masoodian. Digital Management and Retrieval of Physical Documents. In *Proceedings of TEI 2009, International Conference on Tangible, Embedded, and Embodied Interaction*, pages 47–54, Cambridge, UK, February 2009.

[22] Matthew 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, January 2010.

[23] Matthew Jervis and Masood Masoodian. Evaluation of an Integrated Paper and Digital Document Management System. In *Proceedings of INTERACT 2011, International Conference on Human-Computer Interaction*, pages 100–116, Lisbon, Portugal, September 2011.

[24] Matthew Jervis and Masood Masoodian. Visualization of Physical Library Shelves to Facilitate Collection Management and Retrieval. In *Proceedings of EICS 2013, Conference on Engineering Interactive Computing Systems*, pages 133–138, London, UK, June 2013.

[25] Troy Johnson and Patrick Seeling. Localization Using Bluetooth Device Names. In *Proceedings MobiHoc 2012, International Symposium on Mobile Ad Hoc Networking and Computing*, pages 247–248, Hilton Head, USA, June 2012.

[26] William Jones, Harry Bruce, and Susan Dumais. Keeping Found Things Found on the Web. In *Proceedings of CIKM 2001, International Conference on Information and Knowledge Management*, pages 119–126, Atlanta, USA, November 2001.

[27] Ari Juels. *RFID Privacy*. Springer, 2006.

[28] Martin Kaltenbrunner and Ross Bencina. reacTIVision: A Computer-vision Framework for Table-based Tangible Interaction. In *Proceedings*

of TEI 2007, International Conference on Tangible and Embedded Interaction, pages 69–74, Baton Rouge, USA, February 2007.

[29] David Karger, Karun Bakshi, David Huynh, Dennis Quan, and Vineet Sinha. Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data. In *Proceedings of CIDR 2005, Conference on Innovative Data Systems Research*, Asilomar, USA, January 2005.

[30] David Karger and Dennis Quan. Haystack: A User Interface for Creating, Browsing, and Organizing Arbitrary Semistructured Information. In *Proceedings of CHI 2004, Conference on Human Factors in Computing Systems*, pages 777–778, Vienna, Austria, April 2004.

[31] Yan Ke and Rahul Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Proceedings of CVPR 2004, Computer Society Conference on Computer Vision and Pattern Recognition*, pages 506–513, Washington, DC, USA, June 2004.

[32] Alison Kidd. The Marks Are on the Knowledge Worker. In *Proceedings of CHI 1994, Conference on Human Factors in Computing Systems*, pages 186–191, Boston, USA, April 1994.

[33] Jiwon Kim, Steven Seitz, and Maneesh Agrawala. Video-based Document Tracking: Unifying Your Physical and Electronic Desktops. In *Proceedings of UIST 2004, Conference on User Interface Software and Technology*, pages 99–107, Santa Fe, USA, October 2004.

[34] Mark Lansdale. The Psychology of Personal Information Management. *Applied ergonomics*, 19(1):55–66, 1988.

[35] Hyangjin Lee and Jeeyeon Kim. Privacy Threats and Issues in Mobile RFID. In *Proceedings of ARES 2006, The First International Conference on Availability, Reliability and Security*, pages 510–514, Vienna, Austria, April 2006.

[36] Tony Lindeberg. Feature Detection with Automatic Scale Selection. *International journal of Computer Vision*, 30(2):79–116, 1998.

[37] David Lowe. Object Recognition from Local Scale-Invariant Features. In *Proceedings of ICCV 1999, International Conference on Computer Vision*, pages 1150–1157, Kerkyra, Greece, September 1999.

[38] David Lowe. Distinctive Image Features from scale-invariant keypoints. *International journal of Computer Vision*, 60(2):91–110, 2004.

[39] Gerald Madlmayr, Josef Langer, Christian Kantner, and Josef Scharinger. NFC Devices: Security and Privacy. In *Proceedings of ARES 2008, Conference on Availability, Reliability and Security*, pages 642–647, Barcelona, Spain, March 2008.

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

[41] Richard Mander, Gitta Salomon, and Yin Yin Wong. A Pile Metaphor for Supporting Casual Organization of Information. In *Proceedings of CHI 1992, Conference on Human Factors in Computing Systems*, pages 627–634, Monterey, USA, May 1992.

[42] Anne Mangen, Bente R Walgermo, and Kolbjørn Brønnick. Reading Linear Texts on Paper Versus Computer Screen: Effects on Reading Comprehension. *International Journal of Educational Research*, 58:61–68, 2013.

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

[44] Krystian Mikolajczyk and Cordelia Schmid. Scale & Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[45] George Miller. Psychology and Information. *ACM SIGDOC Asterisk Journal of Computer Documentation*, 17(3):3–6, 1993.

[46] Dmitry Namiot and Manfred Sneps-Sneppe. On Mobile Bluetooth Tags. *CoRR*, abs/1502.05321, 2015.

[47] Jan Ondrus and Yves Pigneur. An Assessment of NFC for Future Mobile Payment Systems. In *Proceedings of ICMB 2007, Conference on the Management of Mobile Business*, pages 43–43, Toronto, Canada, July 2007.

[48] Marc Pasquet, Joan Reynaud, and Christophe Rosenberger. Secure Payment with NFC Mobile Phone in the SmartTouch Project. In *Proceedings of CTS 2008, International Symposium on Collaborative Technologies and Systems*, pages 121–126, Irvine, USA, May 2008.

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

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

[51] Beat Signer and Moira Norrie. A Framework for Cross-Media Information Management. In *Proceedings of EuroIMSA 2005, Conference on Internet and Multimedia Systems and Applications*, pages 318–323, Grindelwald, Switzerland, February 2005.

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

[53] Tinne Tuytelaars and Luc Van Gool. Content-based Image Retrieval Based on Local Affinely Invariant Regions. In *Proceedings of VISUAL 1999, Visual Information and Information Systems*, pages 493–500, Amsterdam, The Netherlands, June 1999.

[54] Tinne Tuytelaars and Luc Van Gool. Wide Baseline Stereo Matching based on Local, Affinely Invariant Regions. In *Proceedings of BMVC 2000, British Machine Vision Conference*, pages 1–14, Brustol, UK, September 2000.

[55] Gauthier Van Damme, Karel Wouters, and Bart Preneel. Practical experiences with NFC security on mobile phones. *Proceedings of the RFIDSec*, 9:27, 2009.

[56] Steve Whittaker and Julia Hirschberg. The Character, Value, and Management of Personal Paper Archives. *ACM Transactions on Computer-Human Interaction*, 8(2):150–170, 2001.

[57] Steve Whittaker and Candace Sidner. Email overload: Exploring personal information management of email. In *Proceedings of the CHI 1996,*

Conference on Human Factors in Computing Systems, pages 276–283, Vancouver, Canada, April 1996.

[58] Jingjing Yang, Zhihui Wang, and Xiao Zhang. An iBeacon-based Indoor Positioning Systems for Hospitals. *International Journal of Smart Home*, 9(7):161–168, 2015.