Vrije Universiteit Brussel

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

Tracking my digital documents

Preparation Bachelorthesis

Arno De Witte (0500504)

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

Academic year 2014-2015

Contents

Abstract

Users face a number of problems when dealing with information. Personal information management research consist of identifying these problems and solving them. A number of personal information management systems were developed to help users with these problems. Some existing systems will be discussed as well as the OC2 framework. This framework tries to improve the weaknesses of the other systems but does not support automatic import of information. For this a solution is proposed which separates importing and tracking of objects.

1 Introduction

> Personal Information Management (PIM) refers to both the practice and the study of the activities a person performs in order to acquire or create, store, organise, maintain, retrieve, use, and distribute the information needed to complete tasks (work-related or not) and fulfil various roles and responsibilities.
>
> W. Jones, 2007

1.1 Personal Information Management

The personal information management (PIM) research field studies how organising information is done and how it can be improved both in the digital- and physical space. Both on digital and physical desktop there can be made improvements. For example the file system as we know it isn't optimal for storing information (Henderson & Srinivasan, 2011; Freeman & Gelernter, 1996; Karger, Bakshi, Huynh, Quan, & Sinha, 2003). Files can only be stored once and when making copies of a file, fragmentation occurs which can be a problem when editing one file while not editing the exact copy. Similar problems occur on the desktop, when handling too much files the desk becomes messy and files have to be organised. Every place where infor-

mation is kept is vulnerable to un-orderliness which is a serious issue for e.g. knowledge workers (Whittaker & Sidner, 1996).

In PIM research three activities are defined: keeping, organising and re-

Figure 1.1: Messy desktop example

finding (Jones & Teevan, 2007). Keeping is the activity where information is acquired and managed, organising is the way information is stored and re-finding is finding the right information after it is stored. All of these activities are error-prone (Malone, 1983; Jones & Teevan, 2007).

1.2 PIM systems

To help with the issues mentioned in the previous section, a few personal information systems have been created. These systems can be categorised into three groups.

The first kind of systems try to implement a unified repository where all the information is stored in. This means that all the information is stored in one system which includes the interface that displays the information. The second group tries to create a context-aware systems. These systems provide information depending on a certain context. For example show work emails when at work. The last group tries to combine both unified repositories and context aware features.

1.3 OC2 framework and its expansion

The Object-Concept-Context (OC2) framework is a conceptual framework that is an example of a PIM system that combines both unified repositories and still has context aware features (Trullemans & Signer, 2014). It supports both digital and physical types of information. This means both physical documents like letters and written notes and digital documents like spreadsheets can be imported. However it still lacks the ability to automatically or semi-automatically import information. In its current state information has to be manually imported.
In the next chapter other PIM systems, their criticisms and their approach to this problem will be discussed.

2

Personal information management

2.1 Requirements

A personal information management system should fulfil a number of requirements in order to solve information problems users face. A first problem is information overload. Information overload is the phenomenon where the information required to be processed, being to large for a person not make mistakes, cause stress or lose focus (Edmunds & Morris, 2000; Whittaker & Sidner, 1996). Another major problem is information fragmentation. Information fragmentation occurs when not all information available is stored in a unified system. For example personal information is stored over different devices which makes it harder to find a certain piece (Beets & Wesson, 2013). Finally users face the classification problem, where users store their information in a way to re-find it easily. For example doctors saving patients files by their last name. Users do this according a certain storing method. It's hard to store a single piece of information in a single place, pieces of information can be complex and belong to different categories, as is explained later.
In order to solve these problems a couple of requirements are given. While not all systems introduced in the next section will meet all of the given requirements, we can still define them as a PIM system as long as they help users with keeping, organising and re-finding.

1 Help the user with storing information.

2 Make re-finding of information easier.

3 Allow linking between different resources.

4 Support one or multiple methods of storing information such as piling, filing and mixing described by (Whittaker & Sidner, 1996; Malone, 1983; Henderson & Srinivasan, 2011; Trullemans, 2013; Bälter, 1997).

5 Possibility to add extra information, for example tags, to resources.

6 Provide the possibility to put information in a certain context.

7 Make reminding possible.

These properties were vaguely described in 1945 by Bush (Bush, 1945). He described a system containing all personal information which he called a Memex. His paper is viewed as the start of personal information research.

2.2 PIM systems

As mentioned in the introduction (section 1.2) we can divide PIM systems in 3 groups: unified repositories, context aware systems and combined systems. Unified systems try to solve the information fragmentation problem by unifying all the information into a single system and providing features to help re-find information. Context aware systems try to solve the information overload by providing the right information at the right time. The combined system we will discuss will does both and solves the classification problem. In the following sections some systems for each category will be discussed. Note that most systems are unified repositories since the context aware and combined systems are fairly new.

2.2.1 Unified repositories

The first unified repository system we will discuss is the semantic file system. The semantic file system was a research project from the MIT computer science laboratory (Gifford, Jouvelotl, Sheldon, & O Toole, 1991). This project was created in 1991. It's a low level implementation of a PIM system. It's implement on top of a file system NFS and it support commands like *ls* and *cd*. But it's also possible to use queries in these commands. Virtual directories are then created when these commands with queries in them are executed. Some examples:

```
$ ls -F /sfs/field:
author:/ exports:/ owner:/ text:/
category:/ ext:/ priority:/ title:/
date:/ imports:/ subject:/ dir:/
name:/
$ ls -F /sfs/owner:
jones/ root/ smith/
$ ls -F /sfs/owner:/smith
bio.txt@ paper.tex@ prop.tex@
$ ls -F /sfs/owner:/smith/text:/resume
bio.txt@
```

The directories ending with a ':' are fields and the next directory is the value for this field. As shown in the examples users can search which entries there are for a field and then specify a value. Additional fields and values can be added to further limit the files. It's also possible to list all the possible fields as the first command in the example illustrates. The generation of values for each field is done by transducers. There is a transducer for each type of files. For example mail, latex and text files each have a different transducer. See figure 2.1 for their functionality. This system solves some problems that

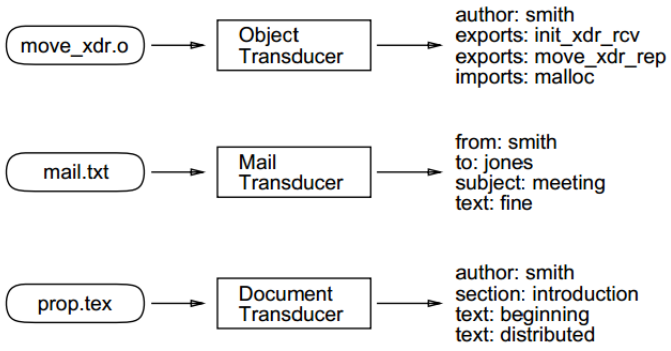

Figure 2.1: Different transducers for the semantic file system

a traditional file system has, like fragmentation of files and the difficulty to re-find files. However some (Henderson & Srinivasan, 2011; Freeman & Gelernter, 1996; Karger et al., 2003) argue that a file system is not an optimal solution for storing files. One of the problems for example is that a file system does not support context based linking of different files.
The semantic file system cannot automatically import documents. Documents have to be stored in the file system and are then indexed and tracked by a background process.

Telenotes is another unified repository system that was created to fill the gaps in other technologies like email, telephone and video conference have while still having a light-weight communication system (Whittaker, Swanson,

Kucan, & Sidner, 1997). Telenotes focused on fulfilling a number of properties: task threading, one-way dropping of messages, quick synchronous connect to a user, shared real-time objects and context regeneration. They also wanted to resemble the messy desktop. This is done by different tasks which accumulate on the desktop and piling notes on top of each other as seen in figure 2.2. This system does support some PIM requirements like

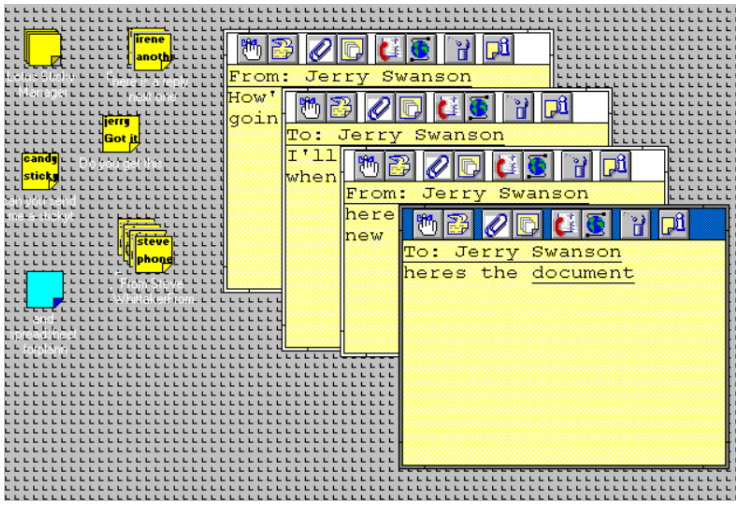

Figure 2.2: Piling in telenotes

helping users to store their information, although only the piling method of storing information is supported. This system automatically imports its information because it's rather a communication tool than a PIM system.

Another unified repository system we will discuss is Lifestreams. Lifestreams is a pure PIM system designed by Freeman and Gelernter in 1996 (Freeman & Gelernter, 1996). In Lifestreams a user can have a number of streams. These are essentially piles of documents on which there are a set of actions: "*new*" to create a new document to the stream, "*clone*" to take an existing document and add to the stream, "*transfer*" to move a document across a stream and "*summarise*" which takes a stream and compresses it into a document. Further there are some additional features like "*freeze*", which makes the stream read-only, "*print*" which prints the current document and "*find*" to search in a stream. The interface is shown in figure 2.3
There is also a possibility to create substreams which are a filtered view of documents. The filter can be a search query or for example removed or archived documents. They are similar to the virtual directories in the semantic file system, seen earlier in this section.

Personal agents are a way to automate certain tasks, for example, when a user uses the phone call agent, this agent will search for phone numbers in the stream and give the option to call a certain person.

Lifestreams supports a lot of the requirements for PIM systems. It supports

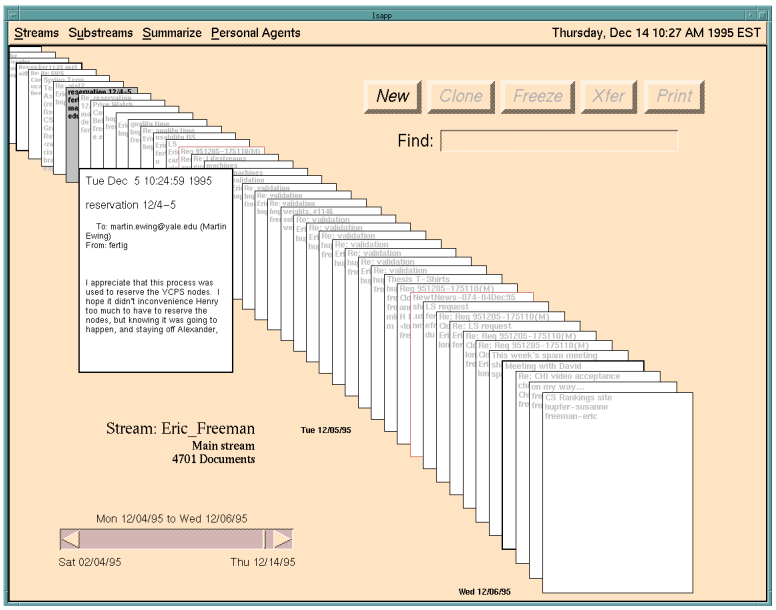

Figure 2.3: Lifestreams interface

storing different files and make them accessible on different devices. Also making the same files accessible in different streams is a step away from the current file system. Linking can be done by storing different files in the same stream. Re-finding information is supported by allowing search (through the find feature). But as Teevan (Teevan, Alvarado, Ackerman, & Karger, 2004) mentioned in *The Perfect Search Engine is Not Enough: A Study of Orienteering Behavior in Directed Search* is a search engine not the right solution for re-finding. A lot of users re-find their information through browsing a structure (for examples folders in a file system) which he called orienteering. In Lifestreams simple orienteering is supported through substreams. While linking between documents is possible by putting them in the same stream. The system doesn't support the possibility to make nested substreams, this means that methods like filing and optimal orienteering, where data is found by passing through folders, isn't possible. Further is the interface focused on piling. The streams are time based, while some may argue that this isn't the best solution as we may want to create semantic links between pieces of

information (Trullemans, 2013). Some modern systems [1] sort information based on popularity.
Lifestreams imports its data from the file system and mail servers. This is not done automatically, users have to manually add their documents and emails into the substream to be able to use it in it.

Haystack is a project created by the MIT computer science lab in 2003 (Karger et al., 2003). Haystack has a RDF data model which is a framework designed to represent information on the web[2]. Their goal was to make a system that fulfilled the needs of every user. Since every user has a different need for information (as previously discussed in this paper). Thus being able to handle different types of information, different relationships and attributes, different ways of manipulating and presenting the information and providing a coherent workspace.
In Haystack users can store any type of information. This information is put into the RDF datamodel through *extractors* which take information from traditional applications. This data model links different resources, for example an email from a friend about a song he liked will be linked to that song.
Haystack is primarily an email client but will also show RSS feeds, appointments and other relevant information in views. These views can be customised, for example a user can see his emails in a list or in a calendar type of view this is show in Figure 2.4. Users can make their own view according to their information needs. Haystack lets users add annotations to anything. They can annotate a user, email and then use the annotations to search and re-find information. The main constraint of this system is the lack of re-finding features. It only supports search and re-finding through piling. Orienteering is mostly ignored, but the system provides both piling and filing through their views feature. Reminding can also be done and context can also be provided through the RDF data model. Linking different resources is automatic but can also be forced by users. Further is it possible to make annotations to all resources which enhance the search feature.

None of the discussed systems provide native support for physical documents. While they try to be an unified repository this repository does not contain all the information available to users.

[1] Facebook https://www.facebook.com/help/166738576721085 and Reddit http://amix.dk/blog/post/19588

[2] RDF 1.1 Concepts and Abstract Syntax W3C Recommendation 25 February 2014 http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

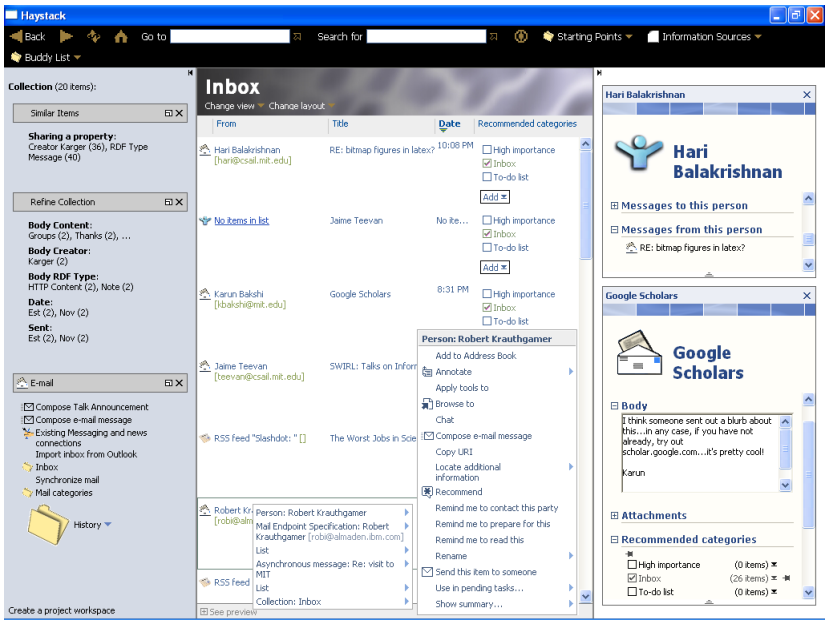

Figure 2.4: Different haystack views

2.2.2 Context Aware systems

Context aware or task-centred information management (TIM) systems rely on personal ontology's. Ontology's describe a set of data and relations between these data objects (Berners-Lee, Hendler, & Lassila, 2001). The concept ontology's was introduced in the semantic web which described the web as a set of agents who convert data through a set of logical rules into new data. These agents use ontology's to be able to conclude new information. This concept is also implemented into the semantic desktop (Sauermann, Bernardi, & Dengel, 2005; Sauermann, 2005) which works in the same way. Personal ontology's consist of relation between data objects which is specific to each individual. The challenge for TIM systems is to build these personal ontology's as accurate as possible.

OntoPIM (V. Katifori, Poggi, Scannapieco, Catarci, & Ioannidis, 2005) is the only TIM system we will discuss, as it is the most import one. In the OntoPIM framework ontology's are very important. The system provides a PIM feature called *Semantic Save*. This will extract information like the date and author from documents. These are domain independent types. But when the user indicates this is for example a travel note, based on ontology's about travel notes OntoPIM will extract attributes. These attributes are the domain specific data. The attributes obtained can be queried by the user to

obtain the processed information.
OntoPIM uses the "*spreading activation*" algorithm to show objects relevant to a certain task (A. Katifori, Vassilakis, & Dix, 2010). This algorithm assumes that each element in the human memory is a node. Each node is associatively linked to other nodes where each link has a bidirectional relevance. When a node is handled it receives a certain activation level. Nodes linked receive a part of that activation level and so on. For example Brussels and Belgium are linked, when you read an email about a conference in Brussels, Belgium also comes to mind because it receives part of the activation given to Brussels by the email you just read.
OntoPIM supports different domains therefor supporting various different contexts or tasks. OntoPIM does only have a query system for re-finding information. Users also have to indicate for each object in which domain it's placed. This doesn't allow for automatic import of information of different domains. When a domain is given automatic extraction of information is guaranteed as long as there is a ontology in for the domain.

2.2.3 Combined systems

The OC2 framework introduced by S. Trullemans (Trullemans & Signer, 2014), combines both a unified repository and context aware or TIM systems. OC2 is not a functioning system like Haystack. However some proof of concept plugins are developed (Trullemans, 2013) to show its use. Main features of this framework are the ability to use multiple user interfaces, native support for physical documents, support for different storing strategies and support for contexts and concepts which allow for linking different objects. The OC2 framework and its model is discussed in the next section.

2.3 OC2 Metamodel

The Object-Concept-Context (OC2) conceptual framework is a framework for context-aware personal cross-media information management systems. This means that these systems provide information based on context (and thus are context-aware) and this information can be both physical and digital. The OC2 framework was designed to support two types of human memory: episodic and semantic memory (A. Katifori et al., 2010). In episodic memory information is linked to events and perceptible properties. For example when you receive a paper at work, you'll remember that you received that paper in a work environment. Systems like MyLifeBits are build for this kind of memory since they store information in groups (stories in MyLifeBits). Se-

mantic memory is the relationships between concepts. This relationship has a bidirectional weight which describes how the concept is related. For example the concept *penguin* maybe stronger related to *black white* than the inverse. Systems like Haystack where objects can be associated with each other, rely more on semantic memory.
This framework is built according to the metamodel shown in Figure 2.5, which is divided into three layers: the context layer, the concept layer and

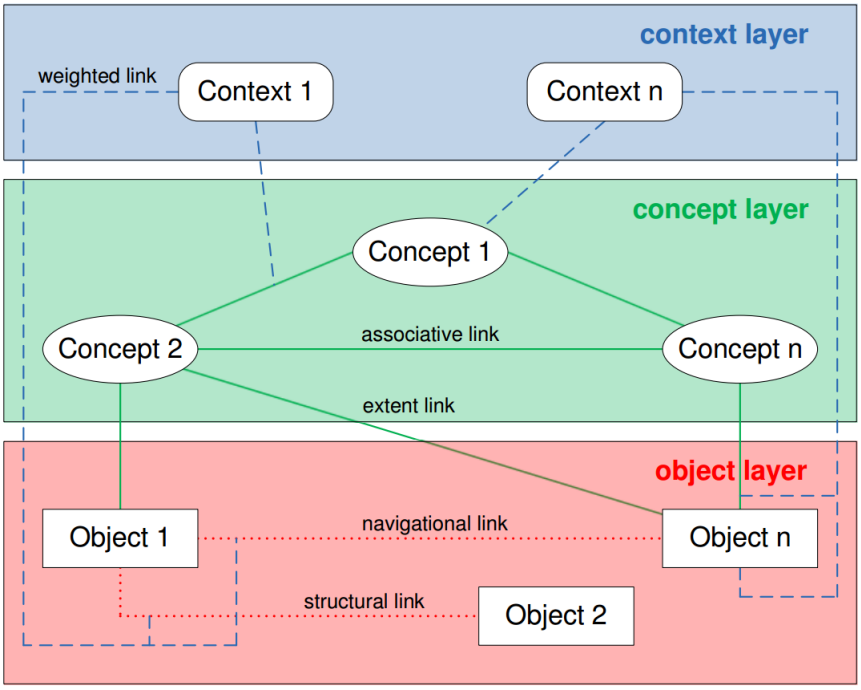

Figure 2.5: OC2 metamodel

the object layer. The object layer consists of objects, which are physical or digital pieces of information like emails, post-its and documents. All objects can be uniquely identified. Furthermore objects can be complex which means they can contain other objects as well. For example a document containing different paragraphs. Objects can be part of navigational and structural links. Navigational links represent a navigational relationship to another object for example hyperlinks between webpages. Structural links on the other hand represent the relationship between a complex object and its parts. A layer higher is the concept layer. Concepts are described as general ideas which abstract the complexity of the real world. As described earlier in this section it's possible concepts are associated with each other, these relations are represented by associative links. The relation between concepts and ob-

jects is represented by extend links. The highest layer is the context layer. A context is defined as an unique composition of contextual factors which are conditions or observation which make objects and concepts more understandable. For example for a context "meeting" the contextual factors can be the date, attendees, the report and the place.
In Figure 2.6 an example of this metamodel is shown. Here we assume the

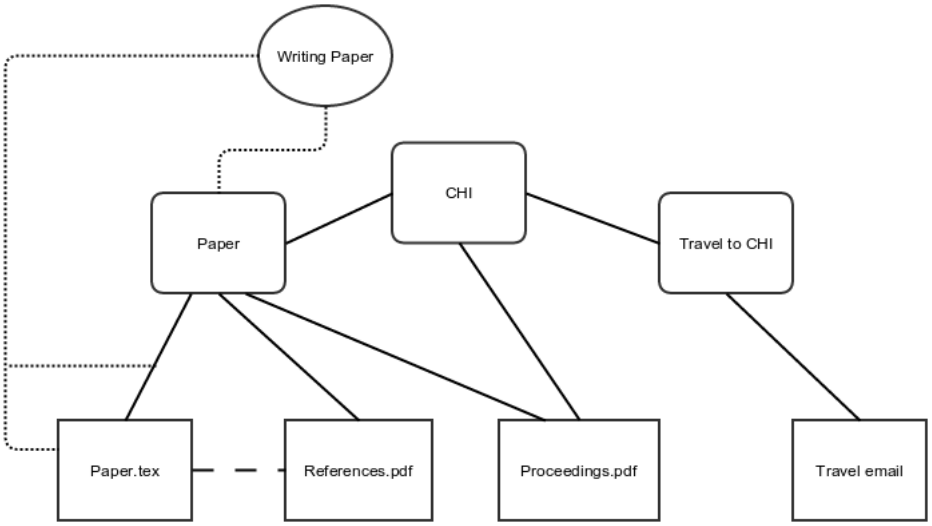

Figure 2.6: An example of the OC2 metamodel

user is a researcher in the PIM field. Three information objects are shown: "*Paper.tex*", "*References.pdf*" , "*Proceedings.pdf* and "*Travel email*". Three concepts are also shown: '*Papers*", "*CHI conference*" and "*Travelling to CHI*" and are all associatively linked. The first three objects are linked to papers. These links are extend links. While *Proceedings.pdf* is linked to both Papers and CHI conference, it has more relevance to CHI than it has to Papers. *Paper.tex* and *References.bib* are structural linked. The context "*Writing Paper*" is linked to the concept *Papers*, the *Paper.tex* object and the extend link between them.
The OC2 framework is implemented by an extension of the RSL model introduced by Signer and Norrie (Signer & Norrie, 2005). The database framework behind this is called iServer which stores all the elements into an db4o database[3] (Ispas, Signer, & Norrie, 2008). As seen in figure 2.7 is the RSL model centred around entities. These entities can be resources, links or selectors. Links are relations between different entities, they have a source and a target. Resources can be files or other pieces of information. Selectors are

[3]http://www.db4o.com/

parts of resources so that these can be linked as well. Each entity can have a number of properties.
The first extension of the RSL model is in the link part. Links can be either navigational links, structural links or extent links as mentioned earlier. In the RSL model an extent link is an associative link but between a concept and an object.
The last extension to the RSL metamodel is the extension of resources. Resources can be concepts, digital objects or physical objects. Either of these can be a context. For example the context *PIM research* can be a concept *PIM research*. Entities can have a relevance to a context. When entities are in a link they can have a relevance to a context as source or as target.

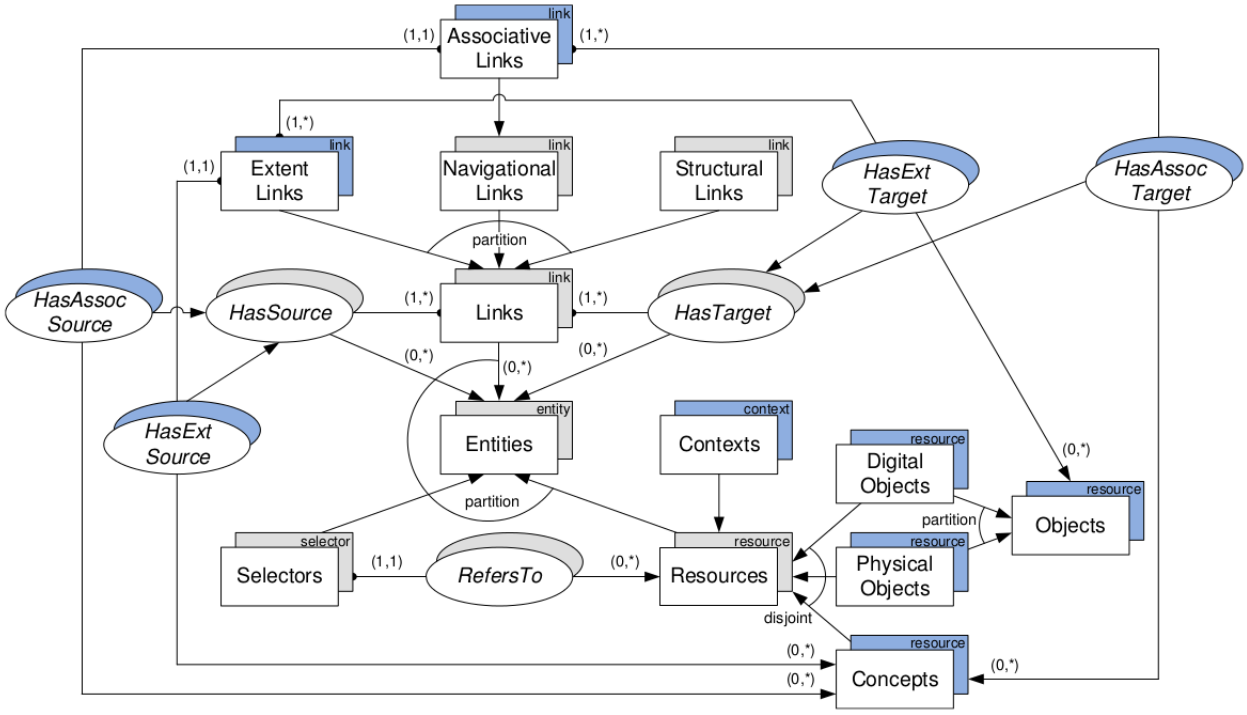

Figure 2.7: Extended RSL metamodel for OC2.

3 Conslusion

A number of PIM systems have been introduced. Their shortcomings have been identified and compared to the OC2 framework. The OC2 framework solves some of the major issues other PIM systems face. It's the only system discussed, apart from OntoPIM, that enables the use of physical documents. It also enables different storing methods such as piling, filing and mixing. This is improvement over other systems (mostly unified repositories) where only one of these methods is supported. But also provides the user with contextual information when using the system.
It however still lacks the ability to automatically or semi-automatically import its information. While some unified repositories are built on top of the data and systems like Haystack and PID have plugins which import their data, is this not the case for OC2 where users have to manually import the data. This system has to be able to both import and track information added to the system. Because information is constantly added, changed or moved.

4 Proposition

4.1 Importing

Importing information automatically into OC2 has to be done in a unified way. Because OC2 provides for both physical and digital objects to be managed, both should be able to be imported automatically. Automatically or semi-automatically importing physical documents is beyond the scope of this bachelor thesis and is handled in a master thesis.This bachelor thesis will focus on importing digital objects into OC2, but will keep importing physical objects in mind while developing the extension of the OC2 framework.
The information imported can be on different locations and systems. Thus an interface is proposed on top of which plugins can be implemented. Plugins can be implemented in a way that suits the source of information well.

4.2 Tracking

Objects should not only be imported from their respected locations (mail servers, file systems, etc.) but should also be tracked. Because tracking is fundamentally different from importing, separate components should be developed into the interface so that different plugins can be developed. An example of tracking is a text file should be tracked so that when changes are

made, that these changes are tracked as well.

4.3 OC2 framework extension

The current RSL model for the OC2 framework does not support that objects are actually stored on a different location than in the framework itself. The RSL model should be expanded with references to the actual location of objects, the kind of resource and which tracking or import is responsible for the resource.
After expanding the RSL model, its implementation should be expanded equally. This implementation is done in OMS Java, this implies that the expansion should be in OMS Java as well (Kolber & Norrie, 2002). The OC2 framework itself also needs to be expanded with the interface mentioned in previous sections and as seen in Figure 4.1.

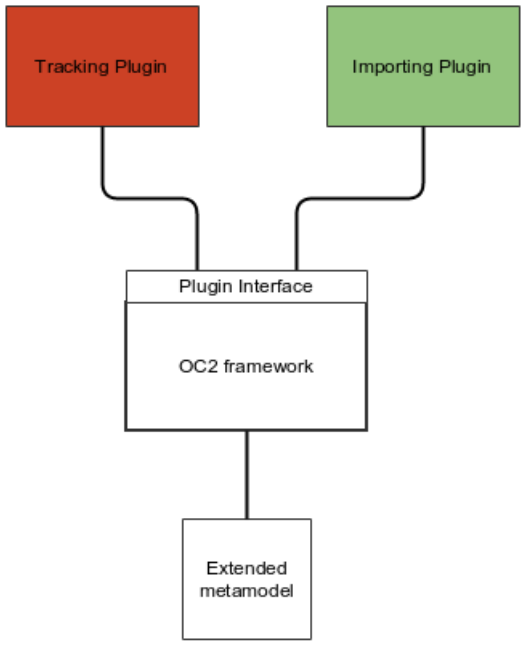

Figure 4.1: Proposed extension of the OC2 framework.

4.4 Proof of concept

To proof the functioning of this expansion of the OC2 framework, two plugins (two import and two tracking plugins) will be developed. These plugins will

handle two types of the same sort of digital objects. For example files on a file system. The import should check new files in a folder or on the file system and the tracking plugin should monitor any changes to files and the file structure.

References

Beets, S., & Wesson, J. (2013). Managing personal information across multiple devices: Challenges and opportunities. In *Human-computer interaction–interact 2013* (pp. 185–192). Springer.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*.

Bush, V. (1945). As We May Think. *Atlantic Monthly*, *176*(1), 101-108.

Bälter, O. (1997). Strategies for organising email. In *People and computers xii* (pp. 21–38). Springer.

Edmunds, A., & Morris, A. (2000). The problem of information overload in business organisations: a review of the literature. *International journal of information management*, *20*(1), 17–28.

Freeman, E., & Gelernter, D. (1996). Lifestreams: A Storage Model for Personal Data. *SIGMOD Record*, *25*(1), 80-86.

Gifford, D., Jouvelotl, P., Sheldon, M. A., & O Toole, J. (1991). Semantic File Systems. *ACM SIGOPS Operating Systems Review*, *25*(5), 16-25.

Henderson, S., & Srinivasan, A. (2011). Filing, piling & structuring: Strategies for personal document management. In *Proceedings of hicss 2011, hawaii international conference on system sciences.*

Ispas, A., Signer, B., & Norrie, M. C. (2008, December). An Extensible Framework for Personal Cross-Media Information Management. In *Proceedings of workshop on cross-media information analysis, extraction and management.* Koblenz, Germany.

Jones, W., & Teevan, J. (2007). *Personal information management.* University of Washington Press.

Karger, D., Bakshi, K., Huynh, D., Quan, D., & Sinha, V. (2003, January). Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data. In *Proceedings of cidr 2003, 1st biennial conference on innovative data systems research.* Asilomar, USA.

Katifori, A., Vassilakis, C., & Dix, A. (2010). Ontologies and the brain: Using spreading activation through ontologies to support personal interaction. *Cognitive Systems Research*, *11*(1), 25–41.

Katifori, V., Poggi, A., Scannapieco, M., Catarci, T., & Ioannidis, Y. E. (2005). Ontopim: how to rely on a personal ontology for personal information management. In *Semantic desktop workshop.*

Kolber, A., & Norrie, M. C. (2002, May). Oms java: A persistent object management framework. In *Java and databases* (p. 46-62).

Malone, T. W. (1983). How do people organize their desks?: Implications

for the design of office information systems. *ACM Transactions on Information Systems (TOIS)*, *1*(1), 99–112.

Sauermann, L. (2005, February). The Gnowsis Semantic Desktop for Information Integration. In *Proceedings of ioa 2005, 1st intelligent office appliances.* Tuscon, USA.

Sauermann, L., Bernardi, A., & Dengel, A. (2005, November). Overview and Outlook on the Semantic Desktop. In *Proceedings of iswc 2005, 4th international semantic web conference.* Galway, Ireland.

Signer, B., & Norrie, M. C. (2005). A framework for cross-media information management. In *Euroimsa* (pp. 318–323).

Teevan, J., Alvarado, C., Ackerman, M. S., & Karger, D. (2004, April). The Perfect Search Engine is Not Enough: A Study of Orienteering Behavior in Directed Search. In *Proceedings of chi 2004, acm conference on human factors in computing systems* (p. 415-422). Vienna, Austria.

Trullemans, S. (2013). *Personal cross-media information management.* Unpublished master's thesis, Vrije Universiteit Brussel.

Trullemans, S., & Signer, B. (2014, October). Towards a conceptual framework and metamodel for context-aware personal cross-media information management systems. In *Er 2014, 33rd international conference on conceptual modelling.*

Whittaker, S., & Sidner, C. (1996, April). Email Overload: Exploring Personal Information Management of Email. In *Proceedings of chi 1996, acm conference on human factors in computing systems* (p. 276-283). Vancouver, Canada.

Whittaker, S., Swanson, J., Kucan, J., & Sidner, C. (1997). TeleNotes: Managing Lightweight Interactions in the Desktop. *ACM Transactions on Computer Human Interaction*, *4*, 137-168.