Vrije Universiteit Brussel

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

Towards Advanced Task Management

Master thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in de Toegepaste Informatica

Joachim Michiels

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

Academic year 2015-2016

Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN
VAKGROEP COMPUTERWETENSCHAPPEN

Towards Advanced Task Management

Masterproef ingediend in gedeeltelijke vervulling van de eisen voor het behalen van de graad Master of Science in de Toegepaste Informatica

Joachim Michiels

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

Academiejaar 2015-2016

Abstract

Keeping track of the things we need to do is a common human activity. It is something we do on a daily basis and enables us to manage our life. We are first confronted with this in our childhood years when we have to use paper agendas to write down homework assignments. This extends further in our adolescent lives where we have to combine work-related and personal tasks on a daily basis.

We are faced with multiple tasks everyday. Sometimes we experience difficulties deciding which tasks we are going to do first, especially if we cannot remember the task at the right place or moment. In our private lives this can cause frustrations to friends or family and can end up with some heated discussions. In the workplace, this becomes an even bigger problem because we have tasks to do and deadlines to keep. If we are unable to finish our tasks on time, we could be wasting a lot of company resources.

This thesis represents a new way into handling tasks and the way we are reminded of todos and tasks. Instead of being reminded in a time-based manner. We will look into finding a way of being reminded when we are in the right context. We will be searching for a way to remind the user of the right todo at the right time with the right information.

Personal Information Management is introduced for the improvement of the organisation and re-finding of personal information. Multiple researchers introduced different systems to support their view on how to make the management of tasks easier or better. We studied their implementations but nevertheless we found out that only partial implementations of context reminding was being used. Since only partial implementations of our view on how to improve todo management was introduced we decided to design and implement our own todo management framework.

We take a look at the Resource-Selector-Link (RSL) model and its implementation in the form of the iServer. We study the design and implementation and the changes that were introduced to this model by the Information Linking and Interaction (ILI) framework. We study all the different design and implementation choices that were made and further build on these with our own todo management framework. After the adaptation to the model and the iServer we build our own TodoServer that utilises the iServer to implement the core functionality of the todo management framework.

We end by implementing a simple GUI to demonstrate the core components of the todo management framework and we introduce a use case to demonstrate a more lifelike example of utilising the todo management framework.

Declaration of Originality

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

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

Acknowledgements

First of all, I want to thank my promoter Prof. Beat Signer for his guidance in the thesis. He helped me when I needed it the most. Secondly, I would like to thank my advisor Sandra Trullemans. She helped me better understand the Information Linking and Interaction framework and helped me with the structure of this thesis.

Furthermore I would like to thank my children, Fiebe and Ben and another special person who I will not mention by name. They were my reason and motivation to finish this work. Many times did they have to wait for my attention but in the end they always stood by me.

Contents

1 Introduction

1.1 Time Management and Task Management

Time management is a term that is widely used in the academic world, the professional world and in everyday life, but it is something that is difficult to define. According to Lakein [42] time management involves the process of determining what you need, defining the goals that you need to achieve and prioritising and planning tasks required to achieve these goals. The work of Kaufman-Scarborough and Lindquist [37] explains time management as ways to assess the relative importance of activities through the development of a prioritisation plan. These two definitions emphasise prioritising activities as a way to effectively manage time.

But based on research on the literature, Brigitte J.C. Claessens, Wendelien van Eerde and Christel G. Rutte [17] suggest a definition of time management as "behaviours that aim at achieving an effective use of time while performing certain goal-directed activities".

They define that the use of time is not an aim in itself and cannot be achieved without looking at other factors. The focus of time management is on a goal-directed activity, such as completing a task, which is carried out in a way that implies an effective use of time. According to a review of time management literature [18] these behaviors are made of:

- Time assessment behaviours, which aim at awareness of here and now or past, present, and future [37] and self-awareness of one's time use (attitudes, cognition, e.g. Wratcher and Jones [69]), which help to accept tasks and responsibilities that fit within the limit of one's capabilities.

- Planning behaviours, such as setting goals, planning tasks, prioritising, making to-do lists, grouping tasks (e.g. Britton and Tesser [13]; Macan [46, 47]) which aim at an effective use of time.

- Monitoring behaviours, which aim at observing one's use of time while performing activities, generating a feedback loop that allows a limit to the influence of interruptions by others (e.g. Fox and Dwyer [27]; Zijlstra et al. [70]).

When they studied the impact of these behaviours, they found that planning showed the most significant results. Bond and Feather [11] for instance, found that the factor "sense of purpose" was the most important factor. Macan [46] found that the sub-scale "goal setting and prioritising" was significantly related to outcomes such as perceived control of time and job satisfaction. Britton and Tesser [13] found a positive relation between short-range planning and grade point average of students, whereas long-range planning was unrelated. They stated that short-range planning was a more effective time management technique than long-range planning because plans could be adjusted to immediate changes or unpredictable situations, which allowed for flexibility. three studies (Hall and Hursch [31]; King et al. [40]; Orpen [53]) showed a positive relation between training time management techniques and performance meaning that performing effective time management strategies could have a positive impact on what we want to accomplish.

Time management strategies are often associated with the recommendation to set goals or tasks. But then the question still remains as to which tasks will have to be completed first? We will have to prioritise these tasks. This can be done in various ways, a few examples include:

- ABC analysis [42]: This is a technique that focuses on the categorisation of large data into groups. The technique uses A, B and C priority groups. The A, B and C are being used for:

  - A: Tasks that are urgent and important
  - B: Tasks that are important but not urgent
  - C: Tasks that are unimportant.

  With this method we can then order the tasks by priority group.

- Pareto analysis: This is the idea that 80 percent of the tasks can be completed in 20 percent of the disposable time. The remaining 20 percent of the tasks will take up the remaining 80 percent of the disposable time. This can be used to split the tasks into two groups. The first group should then get the highest priority [26].
- Eisenhower Method: This method comes from a quote attributed to Dwight D. Eisenhower: "*I have two kinds of problems, the urgent and the important. The urgent are not important, and the important are never urgent*". It is a method that places tasks in a decision matrix as shown in Figure 1.1 (also known as an "Eisenhower Box" or "Eisenhower Decision Matrix" [49]) The matrix uses the following 4 quadrants:
  - Important/Urgent quadrant
  - Important/Not Urgent quadrant
  - Unimportant/Urgent quadrant
  - Unimportant/Not Urgent quadrant

Even though there exist many other methods to prioritise tasks, we can conclude that the selection and completion of tasks forms an important factor when someone wants to manage their time in a productive manner. Because short-range planning seems to be a more effective time management technique than long-range planning it seems to be more optimal to focus our research on a way to manage todos and short tasks.

1.2 Personal Information Management

Information through time has always been used to accomplish daily goals. The five senses are the main input mechanisms for us humans to capture information [12]. The brain then processes these input signals and can store them for later use. It then organises and manages information by creating associations between information items [65]. The main problem with the human brain is that it has a limited storage capacity and information can be lost because people forget things. Throughout time mankind has found multiple

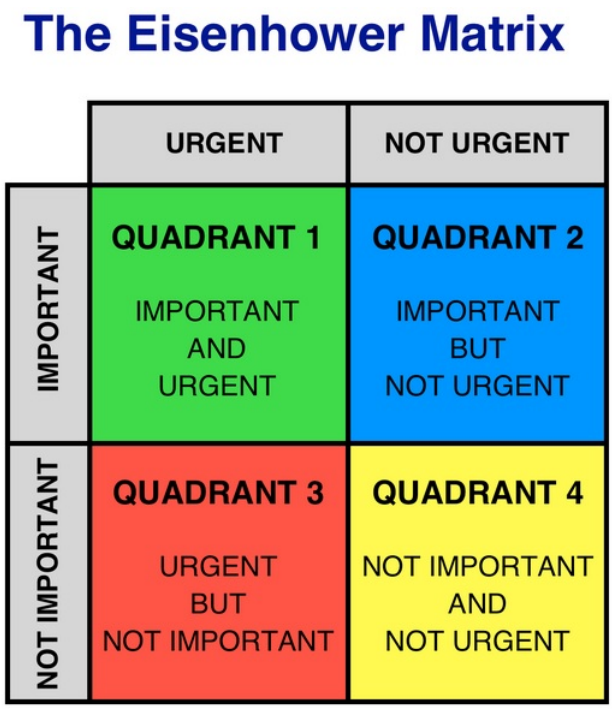

Figure 1.1: Representation of the Eisenhower matrix found on askbutwhy.com

ways to help itself store information. It all started in ancient times where they used writings on cave walls to store information. Later in the fifteenth century, mankind had invented paper and used this as the preferred medium to store information. Paper has not only been used for information storage but also for moving information, placing information in context or reminding the owner of a paper document [57]. Over the last decades, technology has improved and personal computers, tablets, smartphones, harddisks, flash drives and other technological options are being used to store information. The introduction of these new technologies simplifies the storage of information and ensures that there is virtually no limit on the amount of information we can store. With this comes a new problem: The overabundance of information. Because there is so much information, it is difficult to find the right information we seek. The overabundance of information does not only appear in finding information in publicly available data such as on the Internet. It may also occur in our own personal information space. This space contains all the information we keep after finding it; all the information we receive from other people or instances and all information we produce ourselves [63]. This personal information space does not only include digital data

but also physical data (post-its, notes written on paper, books or letters). For example, a user may want to retrieve information about the object of a project meeting. In their search process, they encounter an email with the meeting minutes of that particular project meeting. This report discusses the scope of the project meeting. Nevertheless, another received email from one of the attendees adds remarks to this report and adds extra crucial information to the proposed scope of the project. For the initial search problem the second email would better fit than the first one where the user stopped the search process. If one would have received less emails, they would have been reminded of the second email and stop the search there instead of being satisfied with the first one [63]. Personal information management has a classification problem and a fragmentation problem. According to Dumais and Landauer, the classification problem comes from 2 major problems [23]:

- The first problem comes with the labeling of information. People want to use labels that contain lots of information because that can help them later on when they need to retrieve the information. This introduces the problem that time must be spent when an item needs to be classified. As Cole [19] concludes, users do not want to spend this time and only classify if the benefit goes beyond the initial effort.

- The second problem is introduced with the fact that information can only exist in one category in the physical space without duplicating it. But what should we do when we also need the same information in another category? Already in 1945, Vannevar Bush envisioned the use of selection by association instead of the categorisation of information. He also envisioned the Memex, a mechanical device which would make it possible to define links between information items [15].

The fragmentation problem is based on the diffusion of information in both digital and physical environments. Some information might be written down in a notebook while other information might be stored on a laptop, all in the same personal information space. Not only can some information be duplicated in the digital and the physical environment, another problem lies in the fact that the person will need to remember where which information is stored in order to be able to retrieve it later on.

If we want to address the fragmentation problem we also need to find a way to augment the human memory so that people do not need to lose time with managing information but more enjoying life itself. How would it be if all information is taken care by a system disappearing in our daily life? Users would not be aware of the presence of technological support but they would just use it as we now use a pencil to write on paper. Weiser described this

view in 1991 in his publication entitled *The Computer for the 21st Century* [67] under the term Ubiquitous Computing [63]. Most important, ubiquitous computing will help overcome the problem of information overload. "*There is more information available at our fingertips during a walk in the woods [67]than in any computer system, yet people find a walk among trees relaxing and computers frustrating. Machines that fit the human environment instead of forcing humans to enter theirs will make using a computer as refreshing as taking a walk in the woods*". [67]

The work of Trullemans [63] aims to approach the personal information space as a Personal Cross-Media Information Space covering the digital as well as the physical space. For this she has designed a Personal Cross-Media Information Management System.

1.3 Contribution of This Thesis

Sandra Trullemans has created the Information Linking and Interaction (ILI) framework. It is based on the Resource-Selector-Link (RSL) hypermedia metamodel and provides the necessary functionality for organising personal information, including digital media and physical artefacts, by linking objects to each other or to semantically defined concepts. A major contribution is the ability to define the relevance of an information piece in a given context. She also provides users the possibility to express how relevant several information pieces are for each other. In addition to the linking functionality, an interaction layer was introduced to serve as a basis for different user interfaces, finally leading to an extensible user-centric framework [63].

This thesis builds further on the design and the principles of the Information Linking and Interaction framework to implement a new way of interacting and organising Reminders and todos. As we already discussed, short-range planning seems to be a more effective time management technique than long-range planning. Therefore we focus our research to find a way to manage todos and small tasks instead of large planning artefacts. One problem with current todos is that they are everywhere and nowhere. Some todos are written on a piece of paper, some are being stored in a digital mail, and others are kept on a post-it note. But because they are all kept in different places they loose their purpose. How can they remind the person on the right moment of the right todo? This seems to be related to the fragmentation problem that we discussed in the field of personal information management. But it does not stop there. Because people use different methods and tools to make todos (post-it notes, digitally entered todos) our extension of the information linking and interaction framework

does not only need to resolve the fragmentation problem but also needs to support low extensibility so that it can easily incorporate different ways of making todos.

Most current todo handling systems have limited ways of handling repetitive tasks. Therefore our extension needs to incorporate a way of making and using templates to create todos.

Most todos are being used to handle a specific task for a specific moment, which introduces another shortcoming of current todos. They only remind the person in a time-based manner. The users are reminded only on a specific time to take a specific action. We would like to change this, it seems optimal to remind the user not only in a time-based manner but rather in a context-based manner.

Another shortcoming of current todos that we like to address in this thesis is that there is limited capability to link documents or other information to specific todos. For example, it would be useful that when you are reminded to do some shopping that you automatically are redirected to the latest shopping list that you have stored digitally. And last but not least, our extension needs to be ubiquitous.

The bottom line is that we want the right todo at the right situation with the right information. In this thesis we build further on the work of Trullemans and build further on her Information Linking and Interaction framework to make a todo management framework to unify the input of tasks, support the usage of templates, treat todos and send reminders in a context-based manner and have bidirectional links between tasks and other information items (which includes todos themselves).

1.4 Thesis Structure

The thesis is structured in four parts. The first part provides the necessary introduction of several concepts to be able to follow the further contributions. It gives an important understanding of time management, task management and personal information management. Therefore, the second chapter provides a literature review on the different functionalities that our implementation of a todo management framework should support.

In the second part, we discuss the model interpretations and extensions. We explain the Resource-Selector-Link (RSL) model and explain the adaptations that were made by the Information Linking and Interaction (ILI) framework. We then explain the adaptations that were necessary for our todo management framework to operate and end by discussing the GUI that we made and the use case that was implemented.

In the third part we discuss the implementation. We begin by giving an introduction to the workings of the iServer and the way the original RSL model was implemented in the architecture of the iServer. We then discuss the adaptations that were made to the iServer by the Information Linking and Interaction framework.

We then discuss our extensions to the iServer and introduce and discuss the architecture and implementation of the todoserver which utilises the iServer. We end by discussing the implementation of the GUI applications that we made.

In the fourth and final part, we discuss our solutions and give our conclusions about our findings. We end by giving our vision about possible future work.

2 Literature Review

2.1 Context-Based Task Management

Studies have shown that people organise their work in units of work grouped in a theme, often referred to in literature as tasks or activities [3, 21, 28].

There already exist many books and papers on how to organise these tasks or activities and your time (e.g., [1, 20]). They all explain in one way or another on how to prioritise an amount of tasks effectively [5].

An important factor in prioritising these tasks is giving them a priority based on the moment in time they need to be completed. Tasks that need to be completed more quickly are given a higher priority. This also leads to the way the user is reminded of a specific task. This is something that is almost always done in a time-driven manner.

Most systems remind the user of a tasks based on how much time there is left to complete this task. For example, the user has introduced a task that needs to be completed Saturday at 9.00 PM. The user has selected the option that he wants to be reminded one hour in advance. The system then reminds the user of this particular task Saturday at 8.00 PM.

But reminding the user of a particular task that needs to be executed is not always done in a time-driven manner. As we can see in the research of Ludford et al. [45] previous research has already discussed and designed location-based systems. Early proof-of-concept designs include Cybre-

Minder [22] and comMotion [48]. Their work defined the basic idea: virtual reminders associated with physical locations.

The E-Graffiti [14] and the GeoNotes [25] systems were similar, but were used in the field of social messaging. Other reminder systems are location based in a different way. Gate Reminder [39] is an example of this because it is installed at the doorway of a house.

Researchers have begun implementing location-based systems on cell phones because of the portability and the usage of GPS. For instance, DeDe supports location- and time-based social messages. A user specifies a place or time when the other person will receive a text message [35]. Place-Its is another example which runs on a cell phone and supports location-based reminders [62].

In the research of Kessell and Chan [38] that involved the ceation of castaway, a context-aware task management system. They start by telling that prior work from Bellotti and Ducheneaut [4], Anhalt and Jennings [2] and Rhodes [54] has already explored task management and the delivery of context relevant information. Their study shows that once users got the ability to make location-based reminders 33 % of all items required location-based reminders.

The research we have just discussed [38] shows that there is a need for more than only time-based reminders and once users get the opportunity to use other options they actually use it. This already shows that only using time-based reminders hinders the effectiveness of reminders and incorporating location in the way reminders are triggered can augment the effectiveness.

The research of Ludford [45] explains another location-based reminder system where their studies showed that 84 % of all messages used a location-based trigger to remind themselves of the message. This again shows that once people have the opportunity to use location-based reminders, they actually use it. They also showed that typically, users record information at a base location such as home or work and refer to it at the place where they carry out their task. This indicates that the actual location and context of the user when he records a task is not the same location or context they want to be reminded of that particular task.

But why stop at only time and location? Besides time and location, priority is often a context associated with managing tasks. These are also known as retrieval contexts. They are stored together with the task in order to provide information about when a todo can be carried out [24].

According to Bellotti one of the key features of a good reminder is: “Many to-dos are prompts placed in-the-way in anticipation of a routine practice that will occur at the right moment for the to-do to be discovered” [4]. The right

moment does not only mean the right time but rather the right context. For example, the moment you are in conferencing room A, you meet person A in a meeting that involves project A you want to be reminded of a particular question you want to ask person A about project A. She gives the example: "*when I go to grab my bag to go home, I'll go, I mus take that [object next to the bag] home*". This means that reminders should be triggered in a specific context instead of only on a particular time or only in a particular location.

Prospective memory, the kind of memory that involves remembering to perform a certain todo, is subject to failure [56]. Failures in this memory which leads to failures in task management makes it that tasks aren't completed, tasks are repeated or that the correct tasks are uncompleted. Therefore it is important to have the correct reminder in the correct context.

2.2 Low-Level Extensibility

The way that we record tasks or todos is another thing we need to better understand before we can begin the design of our todo management framework.

Beside the problems with the prospective memory that we already discussed there is also a problem with the fragmentation of task information on different platforms. We are living in the digital age and task information is stored in different programs or applications (email, calendars and other applications). This leads to difficulties in finding, managing, remembering and executing tasks.

That is why it is important that we build a framework that is capable of supporting multiple different input clients. So that all the necessary task information is stored and used in one central place and can be used by every connected client.

In the work of Bernstein [9] they show that information workers still like to write information from their memory to paper. But in the research of Lin [44] they say that digital tools are better suited for processing the captured information in the post-capture phase. This shows that paper should remain to be a possible capture tool for us but the processing of the captured information should be done digitally. Not only is the processing of the captured information better done digitally but also the storage of the captured information should be done digitally because digital storage has virtual no storage limits.

In the paper of Dey and Abowd [22] they provide a list of the following reminder tools:

- Paper To-Do lists
- E-Mail Mailbox
- Post-It Notes
- Personal Information Management Tool
- Human Assistant

This indicates that in order for our todo management framework to be utilised it should support multiple input devices each suited to the specific needs of their specific users.

In the work of Bellotti [4] they conducted a pilot study to study todos and how they are represented. Their study showed that out of all the recorded todos:

- 35,8 % were kept in email
- 11,6 % were kept in online calendar
- 9,7 % were kept in paper list or paper notepad
- 4,7 % were kept in online folders
- 4 % were kept in online special purpose to-do list
- 2,7 % were kept in pda calendar and list combined
- 2,2 % were kept in daytimer/bound notebook/planner

This again indicates that people use different capture tools to capture their todos. It shows the need for our todo management framework to support multiple ways of creating tasks or todos.

But besides the need to support multiple ways of creating tasks or todos this also shows that the fragmentation problem from personal information management exists in task information management. Even when we look at only digital information this problem exists.

Each program that the user uses to introduce information items stores these items in their specific collection [7]. But information items from different programs can be used outside the scope of their programs. For example an email is the primary source of a new event or appointment that is entered

in a calendar. But this calendar is not only used for planning, but also for reporting purposes and recording memorable events [66, 33].

Studies have shown that information is often organised within hierarchies reflecting the users' projects or tasks [7, 10, 32]. The low level extensibility of our todo management framework could resolve these issues.

Besides the digital information fragmentation there is also the logical information fragmentation. This type of fragmentation involves the user's mental model of their personal task information. In the book *Getting thing's done* [1] the author describes the six-level model for reviewing someones work. This also includes the concept of "areas of responsibility". These areas of responsibility are areas where a person wants to achieve results. If a particular area of responsibility that a user needs to accomplish tasks in is not currently in his or hers focus this can result in errors.

In the work of Gonzalez and Mark [29] they discuss that "technological support should be oriented towards helping individuals maintain both local and global perspectives of their working spheres". These working spheres can be linked to the areas of responsibility that we just discussed. They also say that a system should also support "the ability to represent information in portable devices that can be located on their desks or hung on walls, and be connected and synchronised with other tools such as email, electronic calendars, or other systems". Other work has also discussed the support to allow users to organise and group based on responsibilities [4, 20]. This also leads to the need to develop a framework that could link all the task information from different clients together and treat and diffuse this information to the different connected clients. The low extensibility of our framework could help us accomplish this.

2.3 Linking of Information in Tasks

In [45] they state that their research has discovered a common pattern for doing everyday tasks: "*people pre-plan at a base (typically home or work), create information resources (frequently lists), and take these resources with them to refer to at the place where the task is performed (e.g. a grocery store)*".

This actually means that information that could help to perform the task should be linked to the task, so that once the user is ready to perform the task, he has all the information that he needs at his disposal. In this example this means that once the user gets the reminder (possibly triggered because he enters the grocery store and he is in the context of making a purchase) he has access to the grocery list where he lists all the groceries he has to

buy. The bidirectional linking in the Information Linking and Interaction framework could help us achieve this.

In the paper from Bellotti [4] their pilot study shows that "*Todos are used in multiple ways. Sometimes they are part of a list that provides a sense of the amount of work to do. Sometimes they are resources supporting consultation, linking to work objects, or are work objects themselves, displaying state as well as to-do-ness*". This again shows that tasks should be able to have links to other information but it also introduces the idea that tasks should be able to be linked to other tasks.

These two papers show us that tasks or todos should be able to link to information that could be useful to complete the task or todo. But we should also look further into treating todos and tasks themselves as a resource that can be linked to other tasks or todos.

2.4 Templates

Another interesting feature that we would like to research is the usage of templates. In the work of Lepouras and Ioannidis [43] they use templates in their OntoPim system. They give an example of a Task Information Management system where a user wants to book a flight and the Task Information Management system already fills in certain fields with his personal, context-specific information. The usage of templates could be a useful feature in our extension of the Information Linking and Interaction framework.

Todo lists have also proven to be effective reminders of critical tasks [4], with appropriate cues effectively limiting errors [16].

2.5 What Does Already Exist?

While we have looked at some specific points that our Todo Management framework should incorporate it is also helpful to look at some attempts that have already been made:

- The first one is Cybreminder. Cybreminder [22] is an early design that actually incorporates the notion of context for the use of reminders. While this is a good start, it's only a start. They incorporate the notion of context but only at the part of delivery of the reminders. The tasks themselves do not use context. There is no support for multiple input devices or multiple delivery mechanisms. The task themselves are not stored (only the reminder is stored) so there is no possibility to link to extra information or other tasks. There is no support for templates.

- Commotion [48] Early proof-of-concept design of location-based reminders. It has an auto-learning mechanisms for locations and uses a to-do list that can be linked to a certain location. While this supports more than only time it is still limited to time and location and doesn't incorporate the full notion of context. There is also no support for multiple input devices, linking of information in tasks or the use of templates.

- Not only specific systems are being used for task management. Folder hierarchies are sometimes used as kind of project plans [34]. Each folder then contains specific tasks for that specific project. Action-oriented items like emails or temporary files can be used as reminders [68]. Another common file management practice is structuring files within folders representing tasks [32].

3

Model Interpretation And Extensions

In this chapter we are first going to take a look at the Resource-Selector-Link (RSL)metamodel by Signer and Norrie [60] and the extensions made on this model by Trullemans [64]. Once we understand this we can delve deeper into making our own adaptations to support our own todo-management functionalities. We end by discussing the clients that we made to test our todo management framework and a reminder plug-in that simulates the workings of the reminders in our todo management framework.

3.1 The Resource-Selector-Link Model

As already mentioned we are going to make modifications to the Resource-Selector-Link Model (which is basically a general metamodel for hypermedia systems dealing with data, structure and navigation information based on a core set of link concepts) to support our todo-management functionalities. But before we are going to do this we need to take a closer look and explain the Resource-Selector-Link (RSL) Model and the modifications made by Trullemans.

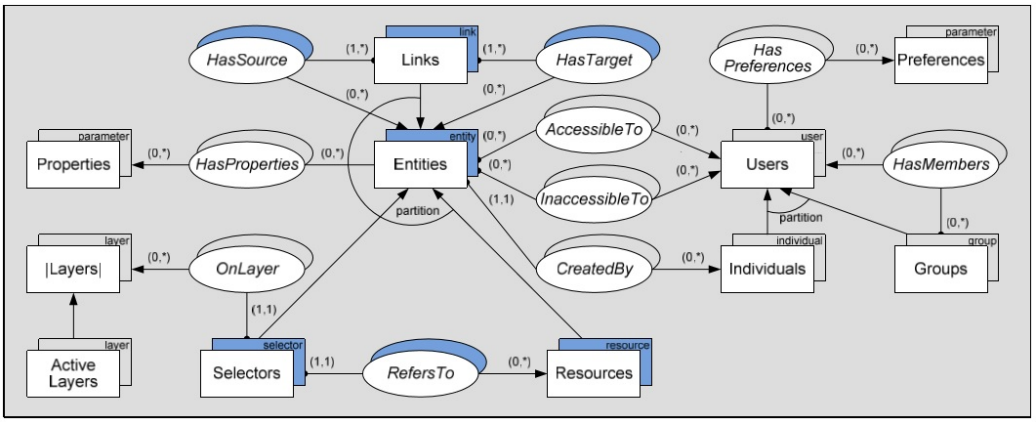

Figure 3.1: the RSL metamodel as cited in [60]

3.1.1 What is The RSL Model?

In 1945 Vannevar Bush [15] gave his vision to the world and since then many hypermedia models and systems have been developed based on his vision. He envisioned information spaces as interlinked collections of resources.

While there exist already some attempts to provide reference models such as Dexter [30] and the Fundamental Open Hypertext Model (FOHM) [50], most hypermedia models and systems are isolated solutions for specific domains or even specific applications. The major problem with these hypermedia models according to Signer and Norrie [60] is the "lack of well-defined conceptual models on which implementations are based. Often models are presented as a mix of architectural, technical and conceptual features. As a result the concepts become obfuscated and restrictions are introduced unnecessarily due to technicalities of the envisaged implementation".

The Resource-Selector-Link Model (RSL) was created as an answer to this problem and several successive projects have been realised based on the RSL metamodel for interactive paper [58]. PaperPoint [61] is an example where the user can navigate and add comments to a PowerPoint presentation by using a printed version of the presentation. Another example is EdFest [6] where the technology was used to add extra functionality to the Edinburgh Festivals Guide.

The RSL Model was defined using the semantic, object-oriented data model OM [52]. The OM model is used as an operational model for data management as well as for system design.

In the RSL model as shown in Figure 3.1, collections are represented by white rectangles where in each collection the shaded rectangle shows the type of the objects. Oval shapes represent associations and are defined be-

tween elements of collections. Cardinality constraints can be set for each participating collection in the association.

The most generic collection in the RSL model is the `Entities` collection. This collection is extended by the `Links` collection, the `Resources` collection and the `Selectors` collection. A partition constraint is used to ensure that an `entity` can only be a member of one of these three collections.

The `Resources` collection only contains objects of the type `resource` and each `resource` represents an information unit.

The `Selectors` collection contains objects of the type `selector` and is being used when we want to address specific parts of a `resource`. An association `RefersTo` represents the fact that a `selector` is always associated with exactly one `resource`, whereas each `resource` can have more than one referencing `selector`.

The third collection is the `Links` collection which contains objects of the type `link`. These are directed many-to-many links between `entities`. A source may be an entire `resource`, parts of a `resource` addressed by a selector or even another `link`. This is being represented in the RSL model because the collection `Entities` is the target collection of both the `HasSource` and the `HasTarget` associations.

A `link` can have more than one source as well as more than one target. But each `link` needs to have at least one source and one target. Because the underlying OM model [41] provides bidirectional associations. All the associations in the RSL model are also bidirectional.

The `HasProperties` association in the RSL model is being used to associate properties to an `entity`. These properties are stored as a set of string tuples in an entity's property attribute. These properties are not predefined by a system implementing the model. This leads to the possibility to customise the behaviour for a specific entity object.

The RSL model features the use of `Layers`. We have already explained the usage of `selectors` to select a part of a `resource`. But what happens when the parts of a `resource` defined by different `selectors` overlap? For example, what happens when Selector A refers to a piece of text on Paper A and Selector B refers to a particular word within the piece of text A on the same paper A? When the user now points to the overlapping part, both `selectors` will be activated. To avoid this the RSL model introduces the notion of `Layers`. The association `OnLayer` is used to relate each `selector` to exactly one `layer`. The association is defined so that each `layer` may contain multiple `selectors` but overlapping `selectors` may not appear on the same `layer`.

Another feature of the RSL model is user management. The user man-

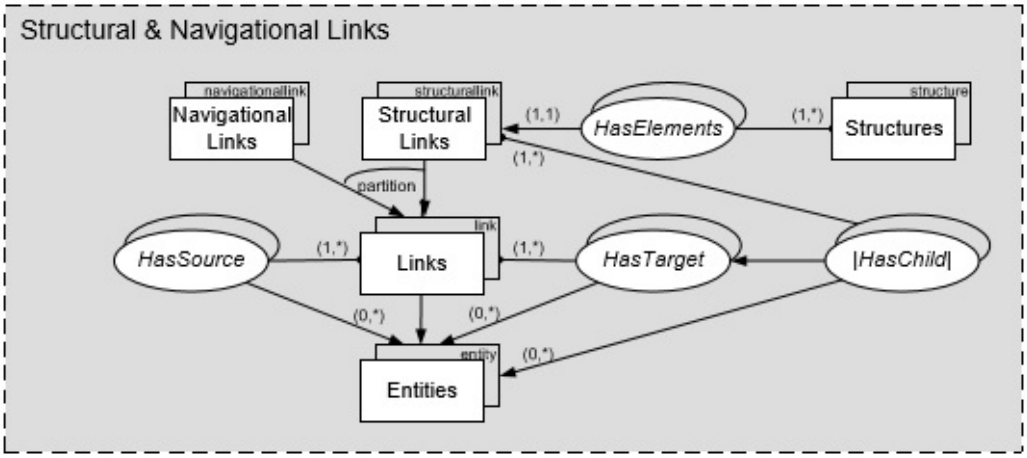

Figure 3.2: Links as cited in [60]

agement part provides the functionality of defining specific access rights to individuals or groups of individuals to implement the notion of data ownership. Most early hypermedia systems did not implement a representation of users as part of the model, whereas some adaptive hypermedia models (e.g. AHAM [8]) did introduce the concept of user models.

In the RSL model, each `entity` is created by one `individual` and an `individual` may create multiple `entities`. Each `individual` is a `user` whereas `users` can also be groups of users by using the association `HasMembers`. This also means that a group of users can consist of different `individuals` but also other groups of users. The associations `AccessibleTo` and `InaccessibleTo` are used to manage different access-rights to different `users` or groups of users.

The last feature that was introduced in the RSL model was the introduction of navigational and structural links as shown in Figure 3.2. A partition constraint is used to ensure that a `link` is either a navigational link or a structural link.

The collection `Structural Links` is a subcollection from the collection `Links`. A `structure` is a member of the collection `Structures`. The `HasElements` association is being used to ensure that a `structure` can contain multiple structural links but a structural link can only be contained within one `structure`. For example, we can look at how a book can use this model. A book has different chapters and each chapter can consist of multiple sections. A chapter is a structure which consists of multiple structural links to different sections. Each section is also a structure and consists of structural links between resources. A resource may of course be the target or source of

several structural links which leads to the reuse of data in a document. This is also known as transclusion as mentioned by Nelson [51].

3.1.2 ILI Framework, an Extension to The RSL Model

Trullemans and Signer [64] have made adaptations to the RSL model to match with the conceptual human memory model.

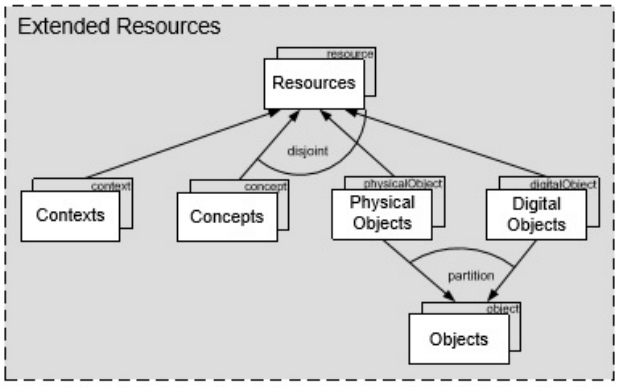

Figure 3.3: Objects as cited in [64]

The conceptual human memory model makes a distinction between concepts which are internal to the memory and objects which are external to the memory. So they introduced the collection `Contexts`, the collection `Objects` and the collection `Concepts` in the RSL model as shown in Figure 3.3. The collection `Objects` uses a partition constraint to ensure that an `object` is either a physical object or a digital object. The collection `Resources` has been extended with the collections `Contexts`, `Concepts`, `Physical Objects` and `Digital Objects`. A disjoint constraint is used to ensure that a `resource` can only be a `concept`, a `physical object` or a `digital object`.

Secondly, the conceptual human memory model imposes a distinction between associative links and extent links. `Extent Links` are a subcollection of `Links` because they have the semantics of linking objects as shown in Figure 3.4. Whereas associative links are more seen as a specific type of navigational links since they represent the navigation functionality between concepts. Therefore, they are a subcollection of the collection `Navigational Links` instead of the collection `Links` [63] .

The third adaptation they have made was the introduction of a new type of association namely the weighted association. This is an association that adds the functionality to add a weight within a range from 0 to 1 to the association itself. Therefore, she has introduced the weighted associations

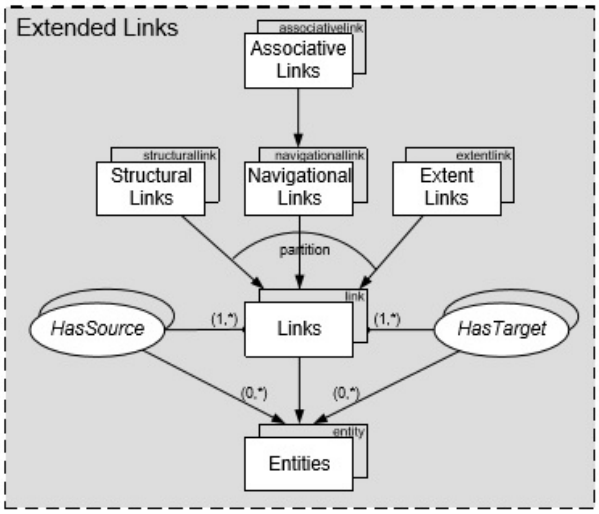

Figure 3.4: Links as cited in [63]

`HasSourceRelevance` which has its startpoint at the association `HasSource` and the `HasTargetRelevance` weighted association which has its startpoint at the association `HasTarget`. Both weighted associations have a `Context` instance as targetpoint as shown in Figure 3.5. In this way, each source and target of a link may have their own relevancy to a context.

3.1.3 Extension of The ILI Framework

After the introduction and discussion of the RSL metamodel and the adaptations made by the ILI framework as shown in Figure 4.1, we may elaborate on the extensions made in the context of this thesis subject matter as shown in Figure 3.7.

The first extension of the ILI model is the introduction of the collection `Tasks`. The collection `Tasks` is a subcollection of the collection `Resources`. We add this new collection in the disjoint constraint so that a `resource` can only be a `concept`, a `physical object`, a `digital object`, a `template` or a `task`. We also introduce the collection `Todos`. In our framework we implement `tasks` as `todos` that are either completed or as `todos` that are not yet started. Therefore the collection `Todos` is a subcollection of the collection `Tasks`.

Both the collection `Todos` and the collection `Tasks` have an association `OnArtefact` to model that `todos` and `tasks` can be found on both `physical objects` or `digital objects`. The association `OnArtefact` uses cardinality constraints to implement that `tasks` or `todos` can reference to multiple

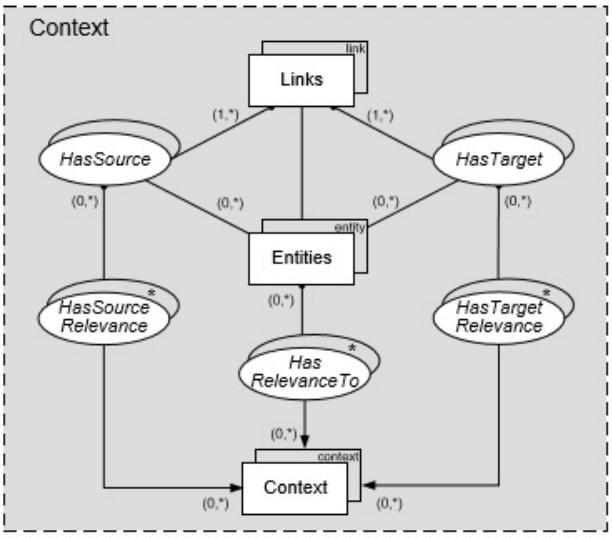

Figure 3.5: Integration of context as cited in [63]

`objects` and `objects` can reference to multiple `todos` or `tasks`.

Another collection that we introduce is the collection `Templates`. This collection is also a subcollection of the collection `Resources` and also utilises the disjoint constraint. It has a many to many association `HasContent` to model that a `template` can consist of multiple `tasks` and a `task` can be part of multiple `templates`.

The fourth new collection that we introduce is the collection `Reminders`. `Reminders` are the way that we want to remind a user of a particular `todo`. We implement the weighted association `HasReminder` to model that we want an ordered list of `reminders` for a particular `task`. The association `HasReminder` has also an association `HasReminderRelevance` with the collection `Contexts` to model that we want an ordered list of `reminders` for a particular `task` but also for a particular `context`. For example, User A wants to be reminded on his tablet of todo A when he is in the context A. He gives this a priority of 0.8 . He also wants to be reminded on his phone of todo A when he is in the context A but he gives this a lower priority of 0.6 . The moment he is in the context A he gets reminded on his tablet of todo A because he gave this a higher priority. If his tablet had run out of power and was no longer available he would have been reminded on his phone because this had the second highest priority.

The fifth new collection that we introduce is the collection `TodoStructures`. It is a subcollection of the collection `Structures` and it has a one to many association `Represents` with the collection `Tasks`. We implement this to

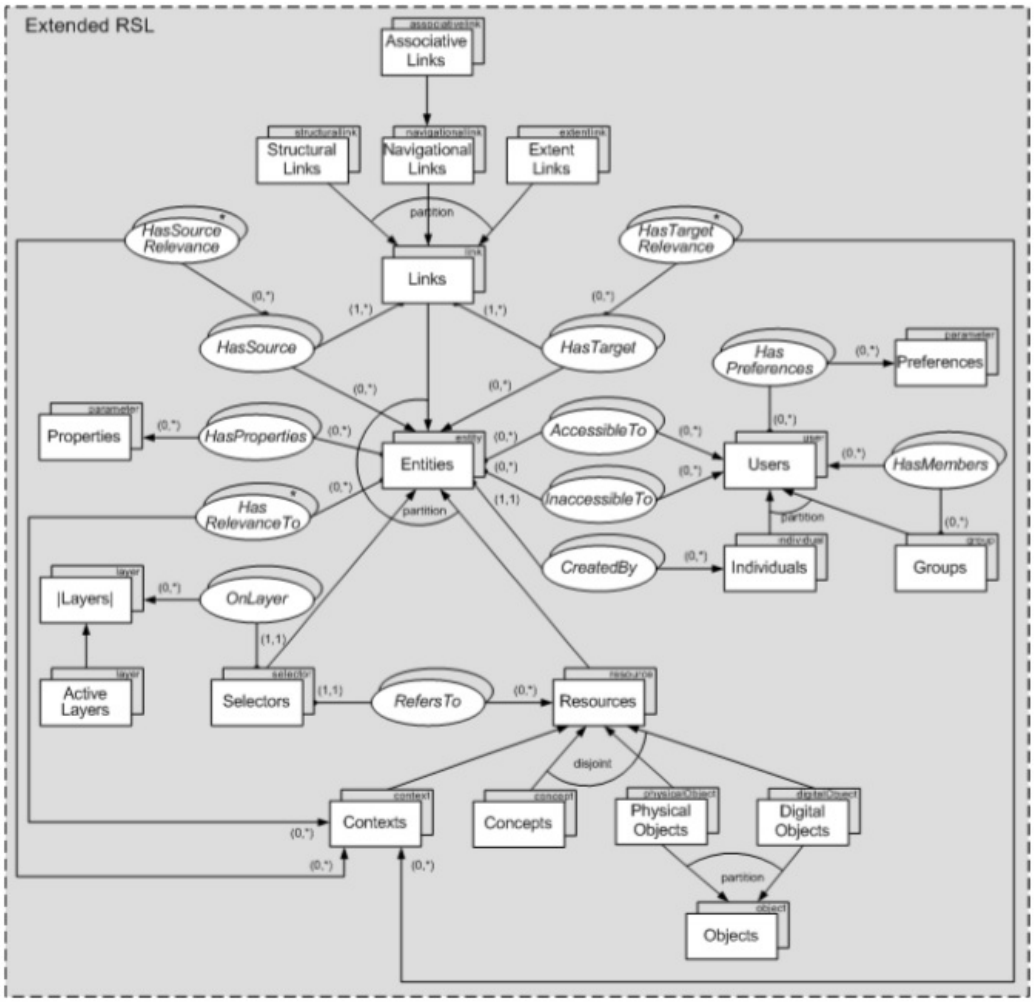

Figure 3.6: the ILI framework model as cited in [64]

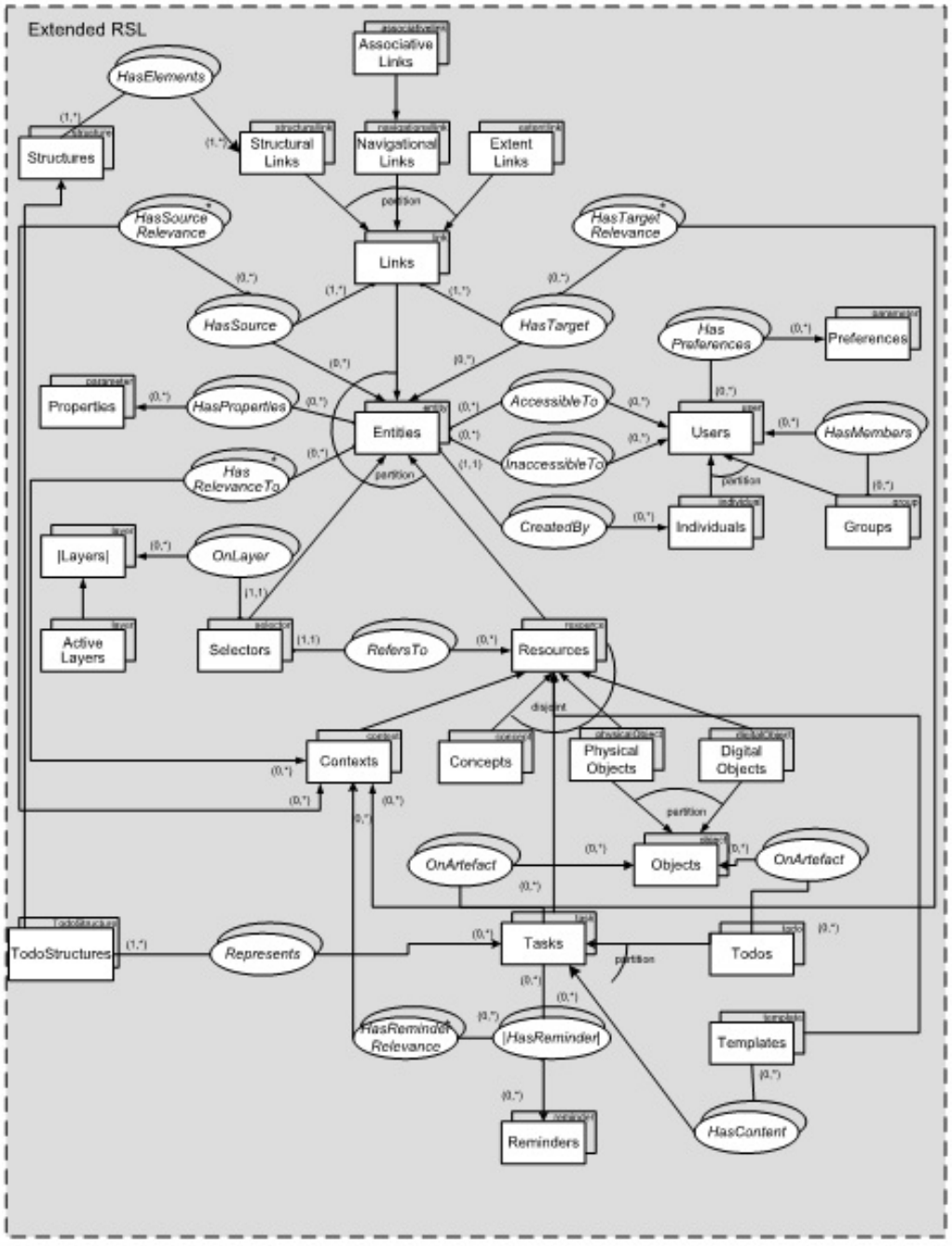

Figure 3.7: Integration of Todo functionalities

model the use of todo lists. Each `todoStructure` has at least one `task` and a `task` can be part of multiple `todostructures`.

3.2 GUI and Reminder Plug-in

As shown in Figure 4.3, the todo framework consists of a todoserver that is incorporated in the task framework that utilises the Information Linking and Interaction framework. Different clients that facilitate the input and manipulation of todos, todolists, templates and reminders utilise the todoserver to do this.

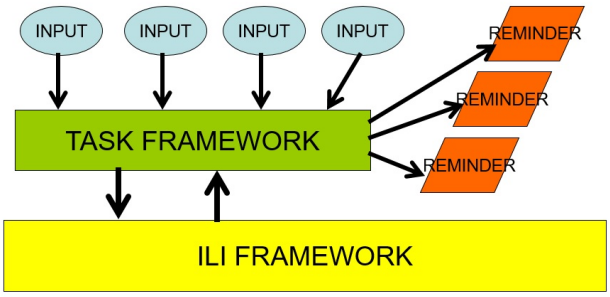

Figure 3.8: Visual representation of the todo framework

The reminder plug-ins register themselves in the todoserver and the todoserver facilitates the reminder procedures and makes it possible to remind the user on the correct reminder plug-in.

The first GUI that we created was a windows tray application that is capable to call all the different functions of the todoserver as shown in Figure 3.9. This GUI was created for basic testing purposes and to demonstrate the functionality of the todoserver.

When you right click on the icon in the windows task tray a menu opens where you can select all the basic functions of the todoserver.

When you select the option Add Task from the main menu a new box opens where you can input a name and a description for the new task as shown in Figure 3.10. Clicking on ok will let the todoserver create the task.

Selecting the option Manage Tasks from the main menu will open a box where you can select a created task and change its name or description as shown in Figure 3.11. In this box it is also possible to delete an already created task.

The screenshots above are given to reflect the basic look and feel of the application. The two next functions are related to the linking of reminders to todos. This functionality is split in two functions: `Add reminder with`

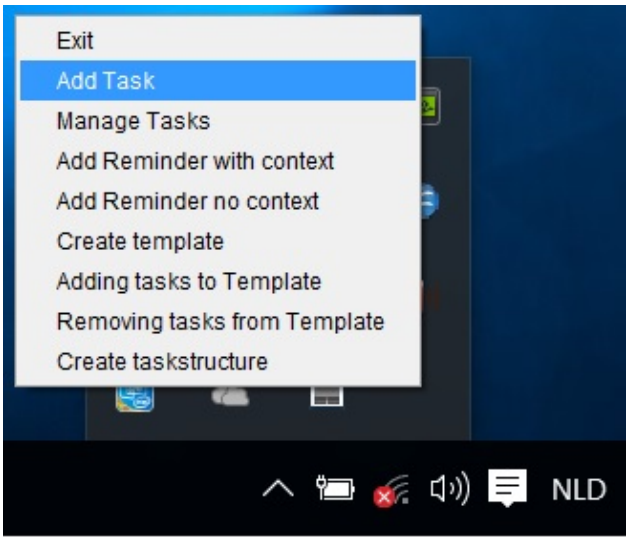

Figure 3.9: windows tray application menu

`context` and `Add reminder no context`. When you select you want to add a reminder to a task with context, the application opens a box where you can select a Task from a list. When you select a task you get the option to select a reminder, a context and the priority for this reminder. The program also shows all the already linked reminders to this task. Adding a reminder without context works the same way but without the option to select a context.

When one clicks on `Create template` this opens up a box where you can enter a name for the template and a description for this template.

You can then select `Adding tasks to template` in the main menu to manage your newly created template. This opens a box where you can select a template and a task to add to this template. It also shows the already linked tasks to this template.

The option `Removing tasks from template` in the main menu opens up a box where you can select a template and a task to remove from this template.

Another option in the main menu is the option `Create taskstructure`. This opens a box where you can enter a name for the taskstructure and a description for the new taskstructure. You also have to select a task since there is no reason to make a taskstructure without at least linking one task to this structure.

Another little tray application that we created is the implementation of a reminder plug-in. This little tray application only has the option to change its state to online or offline. This way we can simulate the available

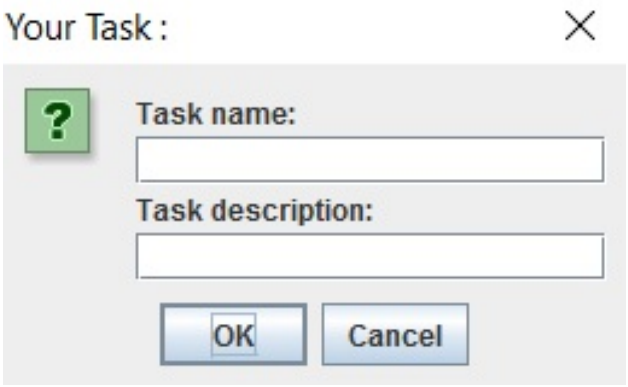

Figure 3.10: Creating a task

and unavailable behaviour of a reminder plug-in. By instantiating multiple instances of this application we can simulate the use of multiple reminder plug-ins and monitor their behaviour in our todo management framework.

Another tray application we implemented was the `context changer`. It is another small tray application that only has the option to change the current context of the todoserver. Together with the other two tray applications we have all the means necessary to simulate all the different functionalities of our todo management framework.

The last tray application was created to simulate a use case where we wanted to give a more lifelike example. The idea behind this application was to simulate the use of the todoserver for a more practical use. The idea was to implement the system of templates so that templates are used as a blueprint for todolists. A template can only have a name, a description and a list of linked digital objects. But without todos that can be linked to this template.

A todolist on the other hand is an implementation of a specific template. When you want to create a todolist you have to select a template to start from, together with the digital objects that are linked to the template that you want to use. Once the todolist is created you can then add or remove tasks. The tasks themselves can be linked to a reminder and a context together with a priority. Reminders are plug-ins that need to register themselves to the todoserver. A reminder has the status available or not available. Once the user's current context changes the todoserver needs to check all the tasks that this user has made and check which of these tasks are linked to the current context with a reminder. For each of these tasks the todoserver then has to check which registered reminder is available and has the highest priority for this task. The found reminder is then triggered by the todoserver

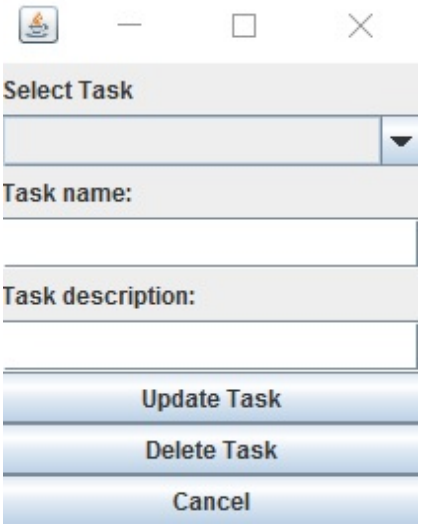

Figure 3.11: Example of editing a task

to remind the user of this specific task.

4 Implementation

After the introduction of the RSL metamodel and our extensions to it in the previous chapter, a discussion of the implementation is given. Initially, the iServer implementation [59] was done in OMS Java [41]. In this chapter we are first going to explore the Information Linking and Interaction Framework. We will then discuss how the Information Linking and Interaction Framework is implemented and end with an explanation about our extensions to this framework to implement our todo management framework.

4.1 The Information Linking and Interaction Framework

Researchers are faced with a big challenge in designing and implementing entire PIM systems for user interface evaluations. Their work consists of organising the personal information on the one hand but also on the visualisation and interaction with personal information on the other hand. The organizing part is done by associations at the lower system level but the visualisation and interaction is done at a higher level.

The Information Linking and Interaction (ILI) Framework tries to solve this problem by separating concerns of the associations at lower level and the user interface design at higher level [63] as shown in Figure 4.1.

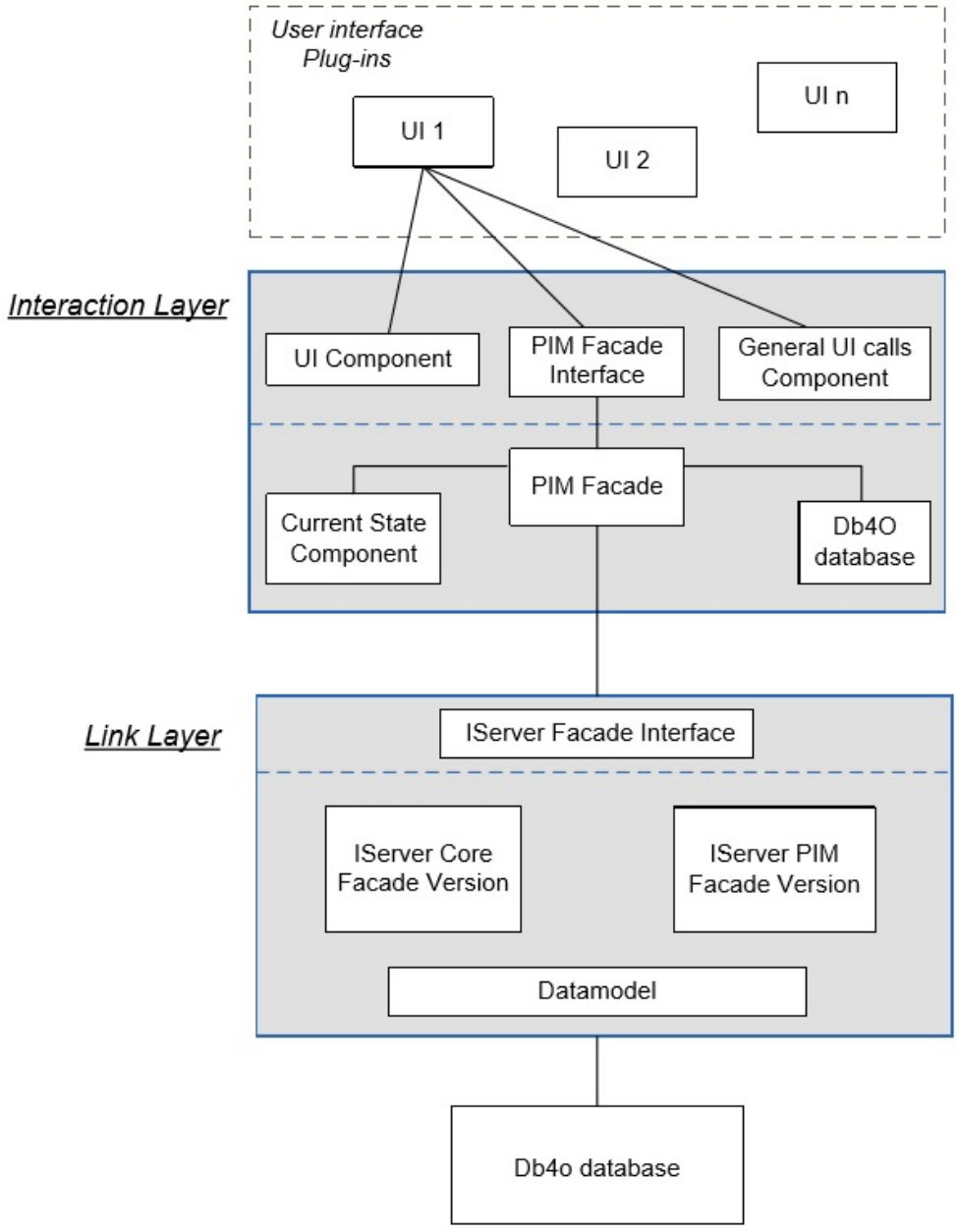

Figure 4.1: the ILI framework as cited in [63]

The ILI framework uses a layered design to separate the concerns of information linking and information interaction. Figure 4.1 shows the ILI architecture. The first thing we notice in this figure is that the framework consists of four parts (db4o database, Link layer, Interaction layer and User interface Plug-ins). The db4o database is used for persistence and the User interface Plug-ins are used for fast prototyping. But the separation of concerns takes place at the Link layer and Interaction layer.

The link layer focuses on the information linking and uses associations at the system level. The functionality of this layer can be compared to systems such as HayStack [36] and Gnowsis [55]. The Link layer provides much more functionality than only associative linking. It also includes the contextual relevancy of these associative links and the time-based organisation of information items. To facilitate all of the above this layer uses a data model which is based on an extension of the Resource-Selector-Link metamodel (which we already discussed). To incorporate the Resource-Selector-Link metamodel this layer was based on the The iServer by Signer and Norrie [59]. The original iServer was made in OMS Java [41] but was later also remade in Java and improved by Trullemans to support the needed functionality given in the conceptual model of the human memory [63]. A Facade and Strategy design pattern is used to enhance the layered architecture and the reusability of the iServer in other applications. The iServer Facade Interface provides the interface for client applications. This interface is then provided in three implementation versions. The first version includes the original iServer with the implementation of the core RSL metamodel elements. The second version includes the extended iServer implementation and is called the iServerCCO version. The third version includes our extension to the iServer implementation to support our todo-management and is called the iServerTODO version.

Secondly, the interaction layer concerns the needed functionality for PIM user interface design and focuses on the information interaction. This layer is formed of three components:

- a PIM component
- a User Interface (UI) component
- a General UI calls component

The UI component provides the functionality to register a user interface which can then be used by other user interfaces. A plug-in mechanism is also provided for these user interfaces.

The General UI calls component provides commonly used calls by the user interfaces.

But the most important component of the Interaction layer is the PIM component. This component controls the access to the Link layer by having the functionality to access the underlying iServer. Besides this, it also provides an abstraction to link elements [63].

Also the current state of a user's personal information organisation is kept and a database for PIM-specific elements is provided by the PIM component.

4.2 The Implementation of The Information Linking and Interaction Framework

The core of the iServer as shown in Figure 4.2 is the implementation of an adapted strategy pattern to support the Create, Read, Update and Delete (CRUD) functionalities.

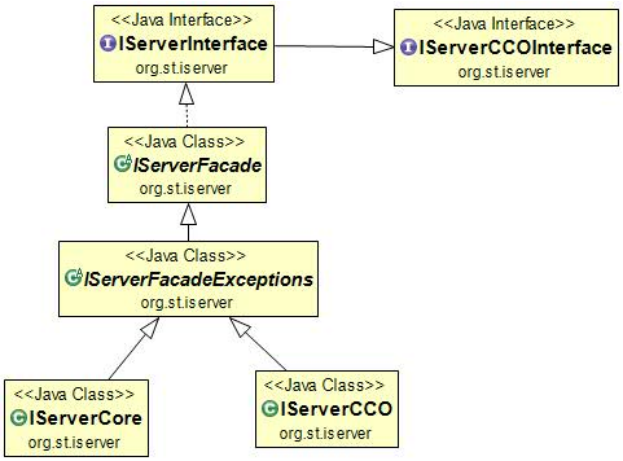

Figure 4.2: Implementation of the core classes of iServer

The purpose of a Strategy pattern is to decouple the client from the actual implementation of the iServer. A second feature of the pattern is the ability to switch between iServer versions without any additional effort required by the client. In the Information Linking and Interaction framework they provide two versions: the core iServer version and a Context-Concept-Object iServer version (iServerCCO). The client can switch between the two implementations by only instantiating the other version. This gives the client the ability to dynamically switch between the two versions and this gives the iServer the ability to change its implementation without affecting the client.

Only the CRUD operations of the core RSL metamodel are provided by the iServerInterface. The iServerInterface is supertyped with an additional

interface for each extension on the RSL metamodel. This is done to assure the client that there will be no changes in this interface. The iServerInterface inherits these additional methods but the client only has to be aware of these inherited methods if they want to actually use them.

The abstract class iServerFacade is introduced to overcome the problem that if we change the implementation of one method concerning a CRUD operation, this needs to be done in every single extended iServer version. The abstract class iServerFacade only implements the CRUD operations of the core RSL metamodel. All extensions then have to inherit from this abstract class and have to provide their implementation of the CRUD operations of their own extensions of the core RSL metamodel. If there is a change needed in one of the core CRUD operations, we only have to change the abstract class iServerFacade and by means of inheritance it will be changed in all implemented iServer versions.

Every extension of the core RSL metamodel has to expand the iServerFacadeException class with its own methods throwing an unsupported exception.

The ILI framework also contains a general abstract AbstractRSLElement class where each stored data object need to be subtyped from. This is done so that a unique identifier is automatically set by every creation of each RSL metamodel object and the objects of its extensions. Therefore two or more iServer versions would always provide a unique name for each object.

The iServer also uses a system of listeners for client applications. All methods concerning CRUD operations in the iServer fire notifications to the registered listeners. A client can then listen to any specific operation or to operations on the whole iServer. These listeners are also implemented for the iServerCCO version.

The Information Linking and Interaction framework also introduced explicit associations and weighted associations in the new RSL metamodel implementation. Since associations are bidirectional, the Java implementation contains two hashtables: the domainCollection hashstable and the rangeCollection hashtable. An association has a startpoint and a target.

When you make an association, the domainCollection hashtable as well as the rangeCollection hashtable are used with this new association to make sure that the association is bidirectional. The domainCollection hashtable adds the pair (domain object, rangeobject) and the rangeCollection hashtable adds the pair (range object, domain object). When the domainCollection hashtable or the rangeCollection hashtable already contain the key object, the value object is added to this key.

A different kind of association that was also implemented was the weighted

association. A weighted association defines a weight between 0 and 1 to a relation between a domain object and a range object (like the normal association). But in a weighted association the domain object can be another association. To implement this the Information Linking and Interaction framework defines a type AssociationInstance. This type has two attributes, a domain and a range (like a normal association). In a weighted association, the domainCollection is defined as a hashtable where the key is an AssociationInstance and the value is another hashtable where the key is an object (the range object) and the value is a double (the weight). The rangeCollection in a weighted association is defines as another hashtable where the key is an object (the range object) and the value is another hashtable where the key is of the type AssociationInstance and the value is a double (the weight). This is done so that the bidirectionality of the weighted association is guaranteed.

4.3 Changes Made to The iServer

Multiple changes were made to the Information Linking and Interaction framework to implement the todo management framework as shwon in Figure 4.3. The first thing that we will discuss are the changes that were made to the iServer as shown in Figure 4.4.

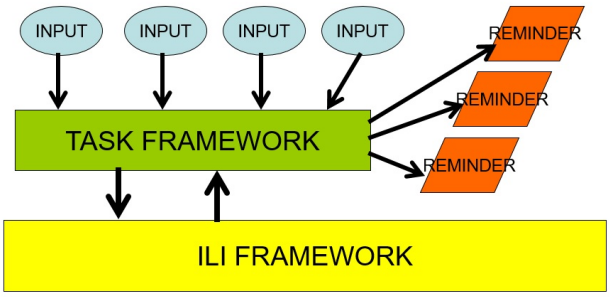

Figure 4.3: visual representation of the Todo framework

First of all we have made changes to the core of the iServer that we discussed in the previous section. We have added the interface IServerTODOInterface which defines all the necessary collections and methods needed in the Todo management framework. The new version of the iServer which implemments all the todo management framework functionalities can be found in the IServerTODO class. A client needs to instantiate this class to have access to the TODO iServer functionalities. The IServerTODO class implements the same system of listeners and action-firing as the Information Linking and In-

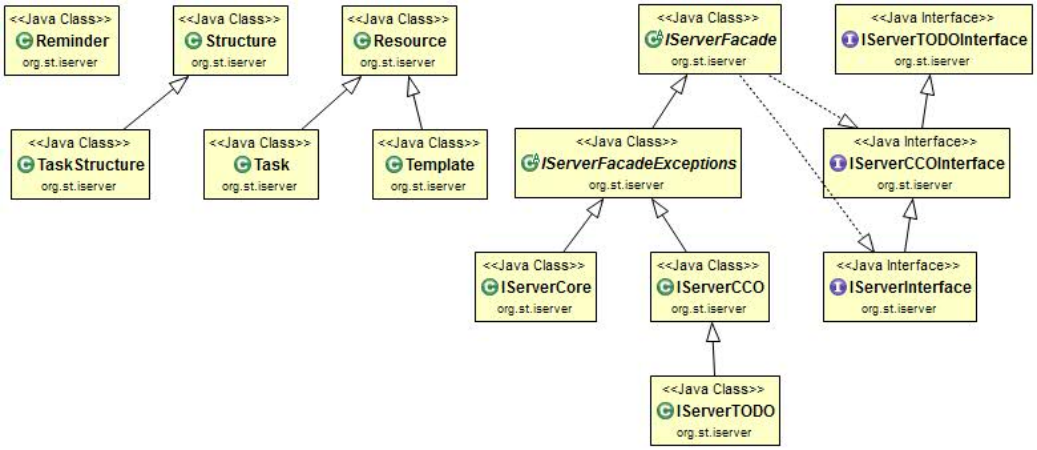

Figure 4.4: Changes made to the iServer

teraction framework. Therefore we also introduced the necessary event and listener classes in the IServer.

We will look in deatil at the adaptations that were necessary to implement the method createTask in IServerTODO:

```
private static HashSet<TaskListener> taskListeners = new HashSet<
    TaskListener>();

HashSet<Collection> collections = new HashSet<Collection>();
Collection resources = (Collection)getDatabaseManager().read(Collection.
    class, "type == Resources").getFirst();
Collection tasks = new Collection("Tasks", resources);
collections.add(tasks);

for(Collection col : collections){
  if(getDatabaseManager().read(org.sigtec.odiJava.Collection.class, "type ==
      \""+col.getTypeName()+"\"").getFirst() == null){
    getDatabaseManager().create(col);
  }
}
```

This code initialises a HashSet TaskListeners that will contain all the listeners of the type TaskListener. The other code is used to populate the HashSet collections with the tasks that are found in the Database.

```
HashSet<BidirectionalAssociation> associations = new HashSet<
    BidirectionalAssociation>();
  associations.add(new BidirectionalAssociation(org.sigtec.odi.Constant.
      INFINITY,
    org.sigtec.odi.Constant.INFINITY, "hasRepresentation"));
  associations.add(new BidirectionalAssociation(org.sigtec.odi.Constant.
      INFINITY,
    org.sigtec.odi.Constant.INFINITY, "hasContent"));
  associations.add(new BidirectionalAssociation(org.sigtec.odi.Constant.
      INFINITY,
```

```
    org.sigtec.odi.Constant.INFINITY, "onArtefact"));
  OrderedBidirectionalAssociation displayedBy = new
      OrderedBidirectionalAssociation(org.sigtec.odi.Constant.INFINITY,
    org.sigtec.odi.Constant.INFINITY,"hasReminder");
  BidirectionalWeightedAssociation HasReminderRelevance = new
      BidirectionalWeightedAssociation(org.sigtec.odi.Constant.INFINITY, org
      .sigtec.odi.Constant.INFINITY, "hasReminderRelevance");


  for(BidirectionalAssociation ass : associations){
    if(getDatabaseManager().read(BidirectionalAssociation.class, "type == \"
        "+ass.getType()+"\"").getFirst() == null){
      getDatabaseManager().create(ass);
    }
  }

  if(getDatabaseManager().read(OrderedBidirectionalAssociation.class, "type
      == \""+displayedBy.getType()+"\"").getFirst() == null){
    getDatabaseManager().create(displayedBy);
 }

 if(getDatabaseManager().read(BidirectionalWeightedAssociation.class, "type
     == \""+HasReminderRelevance.getType()+"\"").getFirst() == null){
  getDatabaseManager().create(HasReminderRelevance);
 }

 getDatabaseManager().commit();
```

The above code is used to initialise our HashSet associations and populate this HashSet with all the necessary associations needed by tasks.

```
public synchronized void addTaskListener (TaskListener listener) {
 if (listener == null) {
  return;
 }
  else {
   taskListeners.add(listener);
 }
}

public synchronized void removeTaskListener(TaskListener listener) {
  taskListeners.remove(listener);
}
protected synchronized void fireTaskInitialised(TaskEvent e) {
  for (IServerListener listener : listeners) {
    listener.taskInitialised(e);
  }
  for (TaskListener listener : taskListeners) {
    listener.taskInitialised(e);
  }
}

protected synchronized void fireTaskUpdated(TaskEvent e) {
  for (IServerListener listener : listeners) {
    listener.taskUpdated(e);
  }
  for (TaskListener listener : taskListeners) {
    listener.taskUpdated(e);
  }
}
```

```
protected synchronized void fireTaskDeleted(TaskEvent e) {
  for (IServerListener listener : listeners) {
    listener.taskDeleted(e);
 }
  for (TaskListener listener : taskListeners) {
    listener.taskDeleted(e);
  }
}
```

With the above code we implement the methods to add and remove tasklisteners (addTaskListener and removeTaskListener) and we implement the methods to fire TaskEvents to notify our iserver when a client creates, updates or deletes a task (fireTaskInitialised, fireTaskUpdated and fireTaskDeleted).

```
private void setObjectAssociations(Object object){
  BidirectionalAssociation ass = (BidirectionalAssociation)
      getDatabaseManager().read(BidirectionalAssociation.class, "type == \""
       + "onArtefact"+"\"").getFirst();
  object.setOnArtefactAssociation(ass);
  getDatabaseManager().update(object);
 }

private void updateObjectAssociations(Object object){
  getDatabaseManager().update(object.getOnArtefactAssociation());
}

private void deleteObjectAssociations(Object object){
  BidirectionalAssociation ass = (BidirectionalAssociation)
      getDatabaseManager().read(BidirectionalAssociation.class, "type == \""
       + "onArtefact"+"\"").getFirst();
 ass.removeRangeObject(object);
 getDatabaseManager().update(ass);

}

public DigitalObject createDigitalObject(String name, String uri, Individual
     creator) throws CardinalityConstraintException {
  DigitalObject object = new DigitalObject(uri);
  setResourceAssociations(object);
 object.setName(name);
 object.setLabel(new File(uri).getName());
 object.setCreator(creator);
 getDatabaseManager().create(object);
 setObjectAssociations(object);
 Collection col = (Collection)getDatabaseManager().read(Collection.class, "
     type == DigitalObjects").getFirst();
 col.add(object);
 getDatabaseManager().update(col);
 fireDigitalObjectInitialised(new DigitalObjectEvent(object));
 return object;
} // createObject

public void updateDigitalObject(DigitalObject object){
 updateResourceAssociations(object);
 updateObjectAssociations(object);
 getDatabaseManager().update(object);
 fireDigitalObjectUpdated(new DigitalObjectEvent(object));
```

```
} // updateObject

public void deleteDigitalObject(DigitalObject object) {
 Collection col = (Collection)getDatabaseManager().read(Collection.class, "
     type == DigitalObjects").getFirst();
 col.removeAllPropagated(object);
 removeResourceAssociations(object);
 deleteObjectAssociations(object);
 getDatabaseManager().update(col);
 getDatabaseManager().delete(object);
 fireDigitalObjectDeleted(new DigitalObjectEvent(object));
} // deleteObject


public PhysicalObject createPhysicalObject(String name, Individual creator)
    throws CardinalityConstraintException {
 PhysicalObject object = new PhysicalObject();
 setResourceAssociations(object);
 setObjectAssociations(object);
 object.setName(name);
 object.setLabel(name);
 object.setCreator(creator);
 getDatabaseManager().create(object);
 Collection col = (Collection)getDatabaseManager().read(Collection.class, "
     type == PhysicalObjects").getFirst();
 col.add(object);
 getDatabaseManager().update(col);
 firePhysicalObjectInitialised(new PhysicalObjectEvent(object));
 return object;
} // createObject

public void updatePhysicalObject(PhysicalObject object){
 updateResourceAssociations(object);
 updateObjectAssociations(object);
 getDatabaseManager().update(object);
 firePhysicalObjectUpdated(new PhysicalObjectEvent(object));
} // updateObject

public void deletePhysicalObject(PhysicalObject object) {
 removeResourceAssociations(object);
 Collection col = (Collection)getDatabaseManager().read(Collection.class, "
     type == PhysicalObjects").getFirst();
 col.removeAllPropagated(object);
 getDatabaseManager().update(col);
 deleteObjectAssociations(object);
 getDatabaseManager().delete(object);
 firePhysicalObjectDeleted(new PhysicalObjectEvent(object));
} // deleteObject
```

We also had to change the implementation that the Information Linking and Interaction framework made for digital and physical objects because our adaptations to the RSL model gave a new OnArtefact association to these objects.

```
public Task createTask(String name, String description, Individual creator){
  Task task = new Task();
 task.setName(name);
 task.setDescription(description);
 getDatabaseManager().create(task);
```

```
 Collection col = (Collection)getDatabaseManager().read(Collection.class, "
     type == Tasks").getFirst();
 col.add(task);
  getDatabaseManager().update(col);
 this.setResourceAssociations(task);
 OrderedBidirectionalAssociation ass = (OrderedBidirectionalAssociation)
     getDatabaseManager().read(OrderedBidirectionalAssociation.class, "type
     == hasReminder").getFirst();
 task.setHasReminderAssociation(ass);

 BidirectionalAssociation ass2 = (BidirectionalAssociation)
     getDatabaseManager().read(BidirectionalAssociation.class, "type ==
     hasContent").getFirst();
 task.setHasContentAssociation(ass2);

 BidirectionalAssociation ass3 = (BidirectionalAssociation)
     getDatabaseManager().read(BidirectionalAssociation.class, "type ==
     hasRepresentation").getFirst();
 task.setHasRepresentationAssociation(ass3);

 BidirectionalAssociation ass4 = (BidirectionalAssociation)
     getDatabaseManager().read(BidirectionalAssociation.class, "type ==
     onArtefact").getFirst();
 task.setOnArtefactAssociation(ass4);

 BidirectionalWeightedAssociation ass5 = (BidirectionalWeightedAssociation)
     getDatabaseManager().read(BidirectionalWeightedAssociation.class, "type
     == hasReminderRelevance").getFirst();
 task.setHasReminderRelevanceAssociation(ass5);

 getDatabaseManager().update(task);
 fireTaskInitialised(new TaskEvent(task));
 return task;
}
```

And last but not least, we implemented the `createTask` method. In this method we begin by initialising the task object. We set the name and the description of this task. We create the task in the db4o database. We add this task object to the tasks collection. We then tell the database to update this changed collection. Since the Task object inherits from the Resource class we use the method `setResourceAssociations` to initiate all the associations available to an object of the type Resource.

We then initialise every association related to the Task object, tell the database to update our newly created task and last but not least we fire a `TaskEvent` to tell the system that a new task has been created. We end by giving back the newly created `Task` object.

The IServerFacadeExceptions class was adapted to support the new todo exceptions.

We also added the classes Reminder, Task, TaskStructure and Template to introduce our RSL model into the iServer implementation. As we discussed in our RSL model, the classes Task and Template inherit from the class Resource. The class TaskStructure inherits from the class Structure and

the class Reminder inherits from the class AbstractRslElement. You may wonder why we did not create a Todo class. This was done because the Todo class exhibits the same behavior as the Task class therefore we choose to implement this in our IServerTODO class. The IServerTODO implements todos as a collection and the manipulation and persistence of todos is handeld by the IServerTODO class. The decision to implement it this way was done in accordance with Sandra Trullemans the creator of the Information Linking and Interaction framework.

4.4 Implementation of TodoServer

While we made the adaptations to the iServer necessary to implement our adapted RSL model. It is now time to look at how we implemented the TodoServer as shown in Figure 4.5. The Todoserver is the core of the todo management framework. It utilises the iServer to implement the RSL model and communicate with the data objects. When we take a look at the architecture of the Todoserver we can see that it utilises the same principle as the iServer to implement its main server component. We have implemented an interface TodoServerInterface where we define all the methods that need to be implemented by the client.

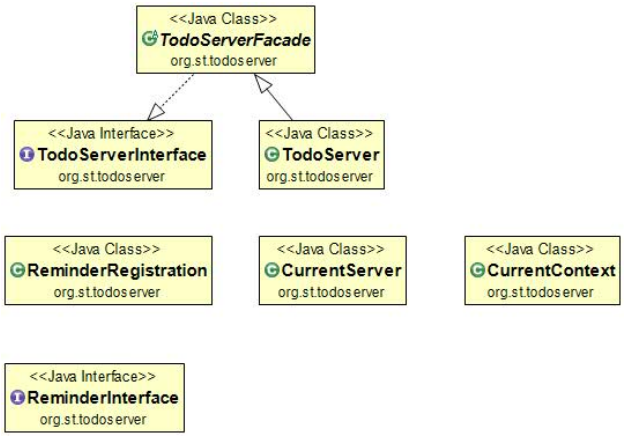

Figure 4.5: Architecture of the TodoServer

The abstract class TodoServerFacade implements the TodoServerInterface. If in the future someone wants to add functionalities to the Todoserver they only need to create an interface themselves and let the TodoServerFacade extend it as well.

The class TodoServer extends the TodoServerFacade and implements all

the methods that we defined in the TodoServerInterface. It also initialises the iServerTODO as an iServerInterface.

We created the class CurrentServer to store a reference to our TodoServer. This way we can make sure that multiple input clients and multiple reminder plugins all make use of the same TodoServer. To instantiate a new TodoServer it suffices to use:

```
TodoServer tServer = CurrentServer.getInstance().getServer();
```

The class CurrentContext is a class that is used and manipulated in TodoServer. It is used to track changes to the current context.

The interface ReminderInterface is implemented to facilitate the use of different reminder plug-ins. When someone wants to make a reminder plug-in it suffices to implement the ReminderInterface and implement the necessary methods.

Another class that is used and manipulated by the TodoServer is the class ReminderRegistration. This class is used to keep track of the reminder plug-ins that are registered and still available. It maintains a HashMap where the key is of the type Reminder and the value is of the type ReminderInterface. Because each reminder plug-in has to implement the ReminderInterface we can search the available reminders in the ReminderRegistration and use the found ReminderInterface to call methods of the reminder-plugin.

```
private HashMap<Reminder,ReminderInterface> map = new HashMap();

public void addReminder(Reminder rem, ReminderInterface remi){
  map.put(rem, remi);
}
```

When we look at the code in the class TodoServer we can see how the ReminderRegistration is used:

```
private ReminderRegistration remreg;

@Override
 public void registerReminder(ReminderInterface remi) {
  Reminder rem = searchReminder(remi);
  if(rem == null){
   rem = iServer.createReminder(remi.getName(), "automatically created",
        remi.getIndividual());
  }
  remreg.addReminder(rem, remi);

 }

 @Override
 public void unregisterReminder(ReminderInterface remi) {
  // TODO Auto-generated method stub
  remreg.removeReminder(remi);
 }
```

We will now look in the code of the TodoServer how this all works together:

```
cContext = new CurrentContext();

@Override
public void changeCurrentcontext(Context cont) {
  cContext.setCont(cont);
  sendReminders();
}
```

When the user enters a new context the client calls the method `changeCurrentContext` from the TodoServer. CurrentContext changes the context and the todoserver then execute the method `sendReminders`.

```
private void sendReminders(){
  Collection<Object> col= iServer.collectionTasks();
 Task task = null;
 LinkedHashMap<java.lang.Object,Double> hmap = null;
 Set<Object> keys = null;
 Boolean send = false;

 for(Object element : col){
  if(element instanceof Task){
   send = false;
   hmap = ((Task) element).getOrderedReminders(cContext.getCont());
   keys = hmap.keySet();
   for(Object element2 : keys ){
    if(element2 instanceof Reminder && send == false){
     if(remreg.isReminderRegistered((Reminder)element2)){
      remreg.giveReminderInterface((Reminder)element2).Remind((Task)element)
          ;
      send = true;
     }
    }
    }

   }
  }
}
```

The method `sendReminders` starts by creating a collection and populating this collection with all the tasks from the iServer. For each object of the type Task in the collection a HashMap gets populated with an ordered list of reminders that are linked with this task and the current context. For each key of this HashMap the todoserver checks if this is of the type Reminder. If this is true then the todoserver checks by using the methods of the ReminderRegistration if the reminder is still available. If all of this is true then the ReminderInterface of this reminder is used to call the method `Remind` of this reminder plug-in. A boolean send is used to only send a reminder to the reminder-plug in that is available and has the highest priority (because the association `HasReminder` is a weighted association).

4.5 Implementation of GUI and Reminder Plug-in

For the implementation of a GUI to demonstrate all the different functions of the todo management framework we opted to use tray applications. We chose this implementation because a tray application is not as intrusive as a normal desktop application. We implemented our tray applications by having one main class to configure the tray application and its main components. For each different panel that's displayed when you select a menu option a new class was made. This way it is easier to make changes to one particular panel without effecting the main tray application or other panels.

The basic implementation of the tray app:

```
if (SystemTray.isSupported()) {

  SystemTray tray = SystemTray.getSystemTray();
 Image image = Toolkit.getDefaultToolkit().getImage("tray.gif");

 //section with all the actionlisteners
 ActionListener exitListener = new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    System.out.println("Exiting...");
    System.exit(0);
  }
 };

  PopupMenu popup = new PopupMenu();
 MenuItem exitItem = new MenuItem("Exit");
 exitItem.addActionListener(exitListener);

 popup.add(exitItem);

 trayIcon = new TrayIcon(image, "Tray Demo", popup);
 trayIcon.setImageAutoSize(true);
 try {
  tray.add(trayIcon);
 } catch (AWTException e) {
  System.err.println("TrayIcon could not be added.");
 }

}
```

All the basic Create, Read, Update and Delete operations of our todo management framework are implemented by our todoserver. Each client (for example this tray aplication) can then use these operations to implement their own way of handling all the different objects we have implemented. This implementation makes it possible to implement the use case that we have discussed previously only by affecting the code of the tray application without touching the code of the todoserver.

An example of the code of the tray application where a panel is instan-

tiated to create a todolist and the todoserver is used to perform the create operation of this todolist:

```
CreateTodolist panel = new CreateTodolist();

int a=JOptionPane.showConfirmDialog(new JFrame(),panel,"create todolist :",
    JOptionPane.OK_CANCEL_OPTION);
if(a == JOptionPane.OK_OPTION){
 todoServer.createTaskstructure(panel.getListName(), panel.
     getListDescription(), panel.getTaskList(), panel.getTemplate(),
     getCurrentUser());
 JOptionPane.showMessageDialog(null, "Todolist was created");
}
```

The reminder plug-in that we made Implements our ReminderInterface and implements the methods that were defined by the interface. This makes it so that our todoserver can call these methods from each reminder plug-in.

```
public class ReminderTray implements ReminderInterface {

@Override
 public void Remind(Task taak) {
  System.out.println("REMIND TODO: " + taak.getName() + " BY REMINDER " +
      name);
 }
}
```

5 Conclusions

We started with a brief introduction to time management and task management followed by an introduction to personal information management. We concluded that the selection and completion of tasks forms an important factor when someone wants to manage their time in a productive manner. Because we saw that short-range planning seemed to be a more effective time management technique than long-range planning we decided to focus our research to todos and short tasks instead of long-range planning.

We saw that the fragmentation problem from personal information management also had an impact in our todo management research. Some todos are written on paper while others are stored in a digital mail. Todos are everywhere and nowhere. But because they are all kept in different places, how can they remind the person on the right moment of the right todo? We concluded that it would be important for our todo management framework to support multiple ways of creating todos but our framework should also be capable of supporting multiple reminder clients.

We also discussed that most current todo handling systems have limited ways of handling repetitive tasks. Therefore our todo management framework needs to incorporate a manner of making and using templates to create todos.

The next requirement of our todo management framework was based on the fact that most todos are being used to handle a specific task for a specific

moment. They only remind the person in a time-based manner. The users are reminded only on a specific time to take a specific action. We would like to change this, it seems optimal to remind the user not only in a time-based manner but rather in a context-based manner.

We also saw that another shortcoming of current todos that we like to address in this thesis was that there is limited capability to link documents or other information to specific todos. Therefore it was necessary for our todo management framework to support this functionality. And last but not least, our framework needed to be ubiquitous.

The bottom line was that we want the right todo at the right situation with the right information. To accomplish this we worked further on the work of Trullemans and extended her Information Linking and Interaction framework to make a todo management framework to unify the input of tasks, support the usage of templates, treat todos and send reminders in a context-based manner and have bidirectional links between tasks and other information items.

5.1 Conclusion

In this thesis we found new insights in the management of todos and a way to augment task management with personal information management. To address the fragmentation problem in task management we optimised the design of our framework.

As shown in Figure 4.3 we have build our todo management framework as a standalone framework that is build on top of the information linking and interaction framework. We have extended the ILI framework to integrate our adapted RSL model and our todo management framework utilises the ILI framework to communicate with the data model. Our framework can easily be used by different clients because we have designed our todoserver to accommodate this. The todoserver also implemented a reminder interface that each reminder plug-in needs to implement as a mechanism to communicate with different reminder plug-ins. This design helps resolve the aforementioned fragmentation problem.

To help resolve the limited capabilities of handling repetitive tasks in most current todo handling systems we introduced templates. These templates were introduced in the RSL model and were implemented in the iServer extension. The todoserver implemented the necessary methods to make and manage templates. The template serves as a way to handle repetitive tasks by letting the user define a template and later reuse this multiple times.

We saw that multiple attempts were already made to manage tasks and

reminders in not only a time-based manner. But we extended further on this idea by not letting our framework only use time or place but instead letting it use context. This way the user is not only confined to place or time but can now freely define his todos and reminders in a context-based manner. To Accommodate this we have build further on the principles of the RSL model and the ILI framework to build further on their idea of utilising context. This made it possible to let the user define their todo in a current context and let the user be reminded of this todo in another context.

Because we have build further on the architecture of the ILI framework we could also build further on their implementation of information linking. We extended further on their RSL model to introduce the information linking in our todo management framework. We designed our own extension of the RSL model to incorporate this and also made it so that it is possible to let todos link to other todos. Because of this the user is capable of being reminded of a particular todo with also all the information necessary to accomplish this todo.

We feel that this thesis can be an important first step into merging task management with personal information management and that the implementation of the todo management framework can be extended to further help the productive management of time and tasks.

5.2 Future Work and Vision

The presented work raises a lot of new opportunities in the fields of task management, time management and personal information management. Future work may build further on our research to look further into long range planning and project management. Project management is an important factor in today's professional life. It would be interesting to see how someone could build further on this work to research how project management could benefit from the findings of this work and the usage of personal information management.

On the implementation field we think it could be useful to look for a way to build further on the todoserver. While we already discussed and found a solution for the fragmentation problem, it still is only a beginning. An interesting idea is to redesign the todoserver to an even more centralised design where the main idea is to use a system of web services to communicate with the different clients. With this design the clients can be programmed in the programming language of choice as long as they respect the wsdl-rules and communicate by xml. The same can also be done for the reminder plug-ins.

A
Bibliography

Bibliography

[1] Allen, David. *Getting Things Done*. Penguin, 2001.

[2] Anhalt, Joshua and Smailagic, Asim and Siewiorek, Daniel P and Gemperle, Francine and Salber, Daniel and Weber, Sam and Beck, Jim and Jennings, James. Toward Context-aware Computing: Experiences and Lessons. *IEEE Intelligent Systems*, 16(3):38–46, May 2001.

[3] Bannon, Liam and Cypher, Allen and Greenspan, Steven and Monty, Melissa L. Evaluation and Analysis of Users' Activity Organization. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 54–57, Boston, United States, December 1983.

[4] Bellotti, Victoria and Dalal, Brinda and Good, Nathaniel and Flynn, Peter and Bobrow, Daniel G and Ducheneaut, Nicolas. What a To-do: Studies of Task Management Towards The Design of A Personal Task List Manager. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 735–742, Vienna, Austria, April 2004.

[5] Bellotti, Victoria and Smith, Ian. Informing the Design of an Information Management System with Iterative Fieldwork. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 227–237, New York City, United States, August 2000.

[6] Belotti, Rudi and Decurtins, Corsin and Norrie, Moira C and Signer, Beat and Vukelja, Ljiljana. Experimental Platform for Mobile Information Systems. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 258–269, Cologne, Germany, August 2005.

[7] Bergman, Ofer and Beyth-Marom, Ruth and Nachmias, Rafi. The Project Fragmentation Problem in Personal Information Management.

In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 271–274, Montréal, Canada, April 2006.

[8] Bergman, Ofer and Beyth-Marom, Ruth and Nachmias, Rafi. The Project Fragmentation Problem in Personal Information Management. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 271–274, Montreal, Canada, April 2006.

[9] Bernstein, Michael and Van Kleek, Max and Karger, David and Schraefel, MC. Information Scraps: How and Why Information Eludes Our Personal Information Management Tools. *ACM Transactions on Information Systems (TOIS)*, 26(4):24, September 2008.

[10] Boardman, Richard and Sasse, M Angela. Stuff Goes Into The Computer and Doesn't Come Out: a Cross-tool Study of Personal Information Management. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 583–590, New York, United States, April 2004.

[11] Bond, Michael J and Feather, NT. Some Correlates of Structure and Purpose in the Use of Time. *Journal of Personality and Social Psychology*, 55(2):321, August 1988.

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

[13] Britton, Bruce K and Tesser, Abraham. Effects of Time-management Practices on College Grades. *Journal of educational psychology*, 83(3):405, September 1991.

[14] Burrell, Jenna and Gay, Geri K. E-graffiti: Evaluating Real-world Use of a Context-aware System. *Interacting with Computers*, 14(4):301–312, July 2002.

[15] Bush, Vannevar. As We May Think. *Atlantic Monthly*, 176(1):101–108, July 1945.

[16] Chung, Phillip H and Byrne, Michael D. Cue Effectiveness in Mitigating Postcompletion Errors in a Routine Procedural Task. *International Journal of Human-Computer Studies*, 66(4):217–232, April 2008.

[17] Claessens, Brigitte JC and Van Eerde, Wendelien and Rutte, Christel G and Roe, Robert A. Planning Behavior and Perceived Control of Time at Work. *Journal of Organizational Behavior*, 25(8):937–50, November 2004.

[18] Claessens, Brigitte JC and Van Eerde, Wendelien and Rutte, Christel G and Roe, Robert A. A Review of the Time Management Literature. *Personnel review*, 36(2):255–276, 2007.

[19] Cole, Irene. Human Aspects of Office Filing: Implications for the Electronic Office. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pages 59–63, Seattle, USA, October 1982.

[20] Covey, Stephen R. *The 7 Habits of Highly Effective People*. Simon & Schuster New York, NY., 1989.

[21] Czerwinski, Mary and Horvitz, Eric and Wilhite, Susan. A Diary Study of Task Switching and Interruptions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 175–182, New York, USA, April 2004.

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

[23] Dumais, Susan T and Landauer, Thomas K. Using Examples to Describe Categories. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 112–115, Boston, USA, December 1983.

[24] Ellis, Judi. Prospective Memory or the Realization of Delayed Intentions: A Conceptual Framework for Research. *Prospective memory: Theory and applications*, pages 1–22, 1996.

[25] Espinoza, Fredrik and Persson, Per and Sandin, Anna and Nyström, Hanna and Cacciatore, Elenor and Bylund, Markus. Geonotes: Social and Navigational Aspects of Location-based Information Systems. In *Proceedings of International Conference on Ubiquitous Computing*, pages 2–17, Atlanta, USA, October 2001.

[26] Ferriss, Tim . *The 4-Hour Workweek: Escape 9-5, Live Anywhere, and Join the New Rich*. Crown Publishing Group, 2007.

[27] Fox, Marilyn L and Dwyer, Deborah J. Stressful Job Demands and Worker Health: An Investigation of the Effects of Self-Monitoring1. *Journal of Applied Social Psychology*, 25(22):1973–1995, November 1995.

[28] González, Victor M and Mark, Gloria. Constant, Constant, Multi-tasking Craziness: Managing Multiple Working Spheres. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120, Vienna, Austria, April 2004.

[29] González, Victor M and Mark, Gloria. Managing Currents of Work: Multi-tasking Among Multiple Collaborations. In *Proceedings of ECSCW 2005*, pages 143–162, Paris, France, September 2005.

[30] Halasz, Frank and Schwartz, Mayer and Grønbæk, Kaj and Trigg, Randall H. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, February 1994.

[31] Hall, Brandon L and Hursch, Daniel E. An Evaluation of the Effects of a Time Management Training Program on Work Efficiency. *Journal of Organizational Behavior Management*, 3(4):73–96, October 1982.

[32] Henderson, Sarah. Genre, Task, Topic and Time: Facets of Personal Digital Document Management. In *Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: making CHI natural*, pages 75–82, Auckland, New Zealand, July 2005.

[33] Jeuris, Steven and Houben, Steven and Bardram, Jakob. Laevo: a Temporal Desktop Interface for Integrated Knowledge work. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 679–688, Honolulu, USA, October 2014.

[34] Jones, William and Phuwanartnurak, Ammy Jiranida and Gill, Rajdeep and Bruce, Harry. Don't Take My Folders Away!: Organizing Personal Information To Get Things Done. In *Procedings of CHI 2005 extended abstracts on Human factors in computing systems*, page 1505Ű1508, Portland, USA, April 2005.

[35] Jung, Younghee and Persson, Per and Blom, Jan. DeDe: Design and Evaluation of a Context-enhanced Mobile Messaging System. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 351–360, Portland, USA, April 2005.

[36] Karger, David and Bakshi, Karun and Huynh, David and Quan, Dennis and Sinha, Vineet. 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, January 2003.

[37] Kaufman, Carol Felker and Lane, Paul M and Lindquist, Jay D. Time Congruity in the Organization: A Proposed Quality-of-life Framework. *Journal of Business and Psychology*, 6(1):79–106, September 1991.

[38] Kessell, Angela and Chan, Christopher. Castaway: a cContext-aware Task Management System. In *Proceedings of CHI 2006 ACM Conference On Human Factors in Computing Systems*, pages 941–946, Montréal, Canada, April 2006.

[39] Kim, Sung Woo and Kim, Min Chul and Park, Sang Hyun and Jin, Young Kyu and Choi, Woo Sik. Gate Reminder: a Design Case of a Smart Reminder. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 81–90, Cambridge, USA, August 2004.

[40] King, Abby C and Winett, Richard A and Lovett, Steven B. Enhancing Coping Behaviors in At-risk Populations: The Effects of Time-management Instruction and Social Support in Women from Dual-earner Families. *Behavior Therapy*, 17(1):57–66, January 1986.

[41] Kobler, Adrian and Norrie, Moira C. OMS Java: A Persistent Object Management Framework. *Java and Databases. Hermes Penton Science*, 6(3):46–62, June 2000.

[42] Lakein, Alan and Leake, Pip. *How To Get Control of Your Time and Your Life.* PH Wyden New York, 1973.

[43] Lepouras, George and Dix, Alan and Katifori, Akrivi and Catarci, Titziana and Habegger, Benjamin and Poggi, Antonella and Ioannidis, Yannis. Ontopim: From Personal Information Management to Task Information Management. *Personal Information Management: Now That We are Talking, What Are We Learning*, 1(1):78, August 2006.

[44] Lin, Min and Lutters, Wayne G and Kim, Tina S. Understanding the Micronote Lifecycle: Improving Mobile Support for Informal Note Taking. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 687–694, Vienna, Austria, April 2004.

[45] Ludford, Pamela J and Frankowski, Dan and Reily, Ken and Wilms, Kurt and Terveen, Loren. Because I Carry my Cell Phone Anyway: Functional Location-based Reminder Applications. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 889–898, Montréal, Canada, April 2006.

[46] Macan, Therese Hoff. Time Management: Test of a Process Model. *Journal of applied psychology*, 79(3):381–391, 1994.

[47] Macan, Therese Hoff. Time-management Training: Effects on Time Behaviors, Attitudes, and Job Performance. *The Journal of psychology*, 130(3):229–236, 1996.

[48] Marmasse, Natalia and Schmandt, Chris. Location-aware Information Delivery with Commotion. In *International Symposium on Handheld and Ubiquitous Computing*, pages 157–171, Bristol, UK, September 2000.

[49] McKay, B and McKay, K. The Eisenhower Decision Matrix: How to Distinguish Between Urgent and Important Tasks and Make Real Progress in Your Life. *A Man's Life, Personal Development*, 2013.

[50] Millard, Dave E and Moreau, Luc and Davis, Hugh C and Reich, Siegfried. FOHM: a Fundamental Open Hypertext Model for Investigating Interoperability Between Hypertext Domains. In *Proceedings of the eleventh ACM on Hypertext and hypermedia*, pages 93–102, San Antonio, USA, May 2000.

[51] Nelson, Theodor H. Complex Information Processing: a File Structure for The Complex, The Changing and The Indeterminate. In *Proceedings of the 1965 20th national conference*, pages 84–100, Cleveland, USA, January 1965.

[52] Norrie, Moira C. An Extended Entity-relationship Approach to Data Management in Object-oriented Systems. In *Proceedings of International Conference on Conceptual Modeling*, pages 390–401, Arlington, USA, December 1993.

[53] Orpen, Christopher. The Effect of Time-management Training on Employee Attitudes and Behavior: A Field Experiment. *The Journal of psychology*, 128(4):393–396, 1994.

[54] Rhodes, Bradley J. The Wearable Remembrance Agent: A System for Augmented Memory. *Personal Technologies*, 1(4):218–224, December 1997.

[55] Sauermann, Leo and Bernardi, Ansgar and Dengel, Andreas. Overview and Outlook on The Semantic Desktop. In *Proceedings of the 2005*

International Conference on Semantic Desktop Workshop: Next Generation Information Management D Collaboration Infrastructure-Volume 175, Galway, Ireland, November 2005.

[56] Schacter, Daniel L. *The Seven Sins of Memory: How The Mind Forgets and Remembers*. Houghton Mifflin Harcourt, 2002.

[57] Sellen, Abigail J and Harper, Richard HR. *The Myth of The Paperless Office*. MIT press, 2003.

[58] Signer, Beat. *Fundamental Concepts for Interactive Paper and Cross-media Information Spaces*. PhD thesis, SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH, Zurich, Switzerland, 2005.

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

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

[61] Signer, Beat and Norrie, Moira C. PaperPoint: a Paper-based Presentation and Interactive Paper Prototyping Tool. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 57–64, Baton Rouge, USA, February 2007.

[62] Sohn, Timothy and Li, Kevin A and Lee, Gunny and Smith, Ian and Scott, James and Griswold, William G. Place-its: A Study of Location-based Reminders on Mobile Phones. In *International Conference on Ubiquitous Computing*, pages 232–250, Tokyo, Japan, September 2005.

[63] Trullemans, Sandra. Personal Cross-Media Information Management. In *Graduation thesis*, Brussels, Belgium, 2013.

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

[65] Tulving, Endel and Thomson, Donald M. Encoding Specificity and Retrieval Processes in Episodic Memory. *Psychological review*, 80(5):352–373, September 1973.

[66] Tungare, Manas and Perez-Quinones, Manuel and Sams, Alyssa. An Exploratory Study of Calendar Use. *arXiv preprint arXiv:0809.3447*, September 2008.

[67] Weiser, Mark. The Computer for The 21st Century. *Scientific american*, 265(3):94–104, September 1991.

[68] Whittaker, Steve. Personal Information Management: from Information Consumption to Curation. *Annual review of information science and technology*, 45(1):1–62, 2011.

[69] Wratcher, MA and Jones, RO. A Time Management Workshop for Adult Learners. In *Journal of College Student Personnel*, volume 27, pages 566–567, Missouri, USA, 1986.

[70] Zijlstra, Fred RH and Roe, Robert A and Leonora, Anna B and Krediet, Irene. Temporal Factors in Mental Work: Effects of Interrupted Activities. *Journal of Occupational and Organizational Psychology*, 72(2):163–185, June 1999.