Designing a User Interface for the Innovative E-mail Client
Semester Thesis

Student: Alexandra Burns
Supervising Professor: Prof. Bertrand Meyer
Supervising Assistants: Stephanie Balzer, Joseph N. Ruskiewicz

December 2005 - April 2006

Abstract

Email Clients have become a crucial application, both in business and for personal use. The term *information overload* refers to the time consuming issue of keeping up with large amounts of incoming and stored email. Users face this problem on a daily basis and therefore benefit from an email client that allows them to efficiently search, display and store their email. The goal of this thesis is to build a graphical user interface for the innovative email client developed in a previous master thesis. It also explores the possibilities of designing a user interface outside of the business rules that apply for commercial solutions.

Contents

Chapter 1

Introduction

This semester project aims at developing the graphical user interface of an email client offering a solution to the information overload problem. The term *information overload* refers to the time consuming issue of keeping up with large amounts of incoming and stored email. Users often have to deal with this problem on a daily basis and therefore benefit from an email client that allows them to efficiently search, display and store their email. Many email clients are designed according to the unofficial standard, i.e. they provide the functions that the other email clients offer. One of the reasons may be that users are reluctant to adapt to a completely new way of doing things, especially with applications like email clients that are crucial to everyday life. The aim of this project is to explore what is possible outside this standard, without having to think of business rules.

The project builds upon an object-oriented framework by Andrea Rezzonico [24]. A first part of this project focuses on identifying the problems that users encounter when working with existing email clients. This information is then be compiled into use cases and scenarios. The *use cases* summarize the experience of a user with a specific functionality of the existing solution - for instance, there is a use case focusing on search mechanisms. The *use scenarios* state what the user should experience with the specific functionality. While the *use case* describes the situation as it is, the *use scenario* focuses on the situation as it should be. The second part of the project the implementation of the graphical user interface that satisfies the use scenarios.

We first identify and study information overload problems of existing email clients. For instance, is it possible to find an email with a specific date and if yes, how easily and how fast is it retrieved? What kinds of viewing options does the user have? We then gather the results to form use cases that are problematic when applied to existing email clients. The graphical interface reuses the elements of the framework [24], in a way that provides the users with easy, concise, fast and flexible access to their email. The resulting system is not evaluated as a prototype, but as an extensible basis for a new email client solution. Upon completion of the development the system is tested using an 4GB email archive.

Chapter 2

Existing Work

This chapter provides an overview of the research related to this project. We emphasize in particular on the research projects where an email client prototype is implemented. We analyze the work based on the following structure: We focus on (1) the methods applied, (2) the problems encountered, and (3) the solutions proposed. We then give a short assessment.

2.1 ReMail

ReMail is a research project of the Collaborative User Experience (CUE) team at IBM. They developed ReMail as an email client prototype to provide users with a "'more integrated email experience"'[6].

2.1.1 Methods

The team used the following methods in their research:

- Collections of email histories
  The client was instrumented to log the actions performed by the users. The details of the messages and threads in an were stored in an encrypted format. The encryption was to ensure that the email histories could be analyzed without knowing the details of the message. The email structure was also statistically analyzed.
- **Traditional usability tests**
  Test users with varying email client experience checked the ReMail prototype for usability of the overall layout and the different functionalities.
- **Prototypes**
  A group of 8 business email users at IBM tested the ReMail prototype for 5 weeks together with their usual email client. The users kept a log book that was discussed with the project team after the trial.

The focus is on building prototypes that incorporate new features and then to observe how users use it on their own email.

2.1.2 Problems Identified

- **Overwhelming volumes of email**
  To keep up with ever increasing volumes of email, user often have to frequently check their inboxes. This increases the disruption that email creates.

- **Losing track of email**
  High volumes of email lead to incoming email quickly moving out of view. Users have to hunt down specific email, this often involves scrolling through the inbox. This problem is aggravated as email comes in as a single stream, and there is no differentiation between spam, informal mail and mail requiring action.

- **Pressure to respond**
  Users feel pressure to respond quickly to incoming email, with many email requiring immediate attention.

- **Different types of email**
  New kinds of email have emerged. While email still includes traditional "'letter"'messages, there are also invitations, receipts, transactions, discussions, conversations, tasks and newsletters. Many email clients do not provide the mechanisms to efficiently handle these new kinds of email.

2.1.3 Proposed Solutions

As part of the larger project "'Reinventing email"'the team developed an email client prototype Remail. The solution is based on three principles: (1) Visualizations to aid navigation and understanding (2) Advanced Text Analysis on the content of emails and (3) Features to help users manage their attention, i.e. ways to differentiate between important and non-important messages. Remail also includes search mechanisms.

- **Visualization**
  The remail prototypes provides different types of visualizations, so called *Maps* . The *Thread Map* shows the thread tree for a message selected. The current message is highlighted and other messages, such as unread ones, are visualized using different colors and fonts. The *Correspondent Map* groups messages in a folder by sender. Senders are grouped by domain (e.g. inf.ethz.ch) and are ordered by number of messages sent. Different shades of color are used to indicate the age of the email. The *Message Map* is used to show the user the relationship of the messages in a folder. For instance, the relationship between two messages can be: in the same thread, one message is a reply to the other, etc. Different colors are used to show messages in the same thread and by the same sender.

- **Text Analysis**
  Date extraction [26] is a mechanism to extract informally written dates from an email text. For instance, "'lets meet next Monday"' is identified as date and time phrase, parsed using the context of the message and then highlighted and displayed to the user. The user can click on the highlighted dates to generate a calendar entry. A thread is summarized using the

Thread Summarization method. This simple method consists of extracting the first line from each of the messages in the thread and combining this information. The foldering system used in Remail is called *Collections* . Like in regular foldering systems filters can be defined according to which email is rerouted. The difference is that as opposed to folders messages can belong to more than one collection.

- **Attention Management**
  One of the mechanisms to manage the users attention is to differentiate between *In-sight* and *Out-of-sight* collections. The not so important messages in the out-of-sight collection are removed from the inbox view so that the user can focus on the more important messages. The in-sight collections remain in the inbox view. The user can choose between the two options when creating a collection. Users often need information from *Multiple Data sources* to do their job. The Remail prototype supports messages from different sources such as Lotus Notes Email, POP3 and IMAP among others.

- **Search Mechanisms**
  The search mechanisms include an instant search (a character-by-character filtering of the current message list based on the information in the message headers), full-text search and a date search using date extraction.

2.1.4 Assessment

The ReMail research group has more than twenty members, so it looks like IBM is putting an emphasis on the research in this area. There are also numerous publications of the group, including a paper on the experiences of testing the ReMail prototype [10]. The ReMail prototype itself focuses on solving the information overload problem with visualization. While the correspondent map is visualized in a very intuitive way, the thread and message map are less so to a new user. The principles of what to visualize could probably be visualized in a more user friendly way.

2.2 Inner Circle

Inner Circle is a research project of the Microsoft Research Lab and the University of Washington. The Inner Circle email prototype presents an automatic people-centered organization of emails. messages are associated with the people involved [28].

2.2.1 Methods

The team used the following methods for their research:

- **Comparison with Microsoft Outlook**
  The graphic interface of Inner Circle was designed to resemble the Microsoft Outlook GUI. This made a direct comparison of the user experiences easier.

- **User Study**
  Six participants from Microsoft tested the Inner Circle application in a lab session. They were given tasks to solve and were interviewed about their experiences after the session.

2.2.2 Problems Identified

- **Manual Message Organisation in Folders**
  The problem is how to find a way of organizing the messages that is easier to understand and maintain than the traditional folder hierarchy. The goal is to relieve the users of the burden of manually organising their email.

- **Old and Irrelevant Information in the Inbox**
  The Inbox is often cluttered with old and irrelevant messages. This makes the task of retrieving the important emails more difficult and time consuming.

- **Collaboration not Supported**
  Current solutions do not reflect the fact that email is often used as a collaboration tool. Documents are exchanged between a group of people using email, leading to different versions in various locations. A shared document repository is required to solve this problem.

- ***Nuggets* not in Direct View**
  Nuggets are elements such as attachments, file links and URL s embedded in email messages. It is often to find this information by navigating the messages. Nuggets should therefore be part of the view.

2.2.3 Proposed Solutions

- **People List**
  People List is a list that includes entries for all contacts that appeared as a sender or recipient in any message. The list is ordered according to how frequent and how recent the interaction with the contact was. Due to this ordering People List can accommodate large numbers of contacts. When the application is started, only the most recent contacts, i.e. the "'Top Ten" is shown - however the list can be expanded upon request. To avoid spammers being listed at the top, sent messages are counted with a higher weight than received ones.

- **Conversational View of Messages**
  The Conversational View idea is based on the observation that emails exchanged within a group often form a conversation. Messages are associated with the group of people involved and the Conversational View displays these messages in a uniform way. For each message, time, sender, subject and the full message body are displayed. From reply or forward messages only the new information is included.

- **Mailed Items**
  Mailed Items is a concept to view *nuggets* that have been exchanged by email within a group. It is a virtual shared space that users can browse.

Mailed Items display all attachments and all identified file links and URL s that were exchanged between the group. The shared space is accessible for all users that are part of the group and only shows the items that were exchanged within this group. Documents exchanged with other parties are not shown. It also includes calender events involving the members of the group. The nuggets are associated with the emails that were used to exchange them. This allows the users to view the context.

- **Search Mechanisms**
  Inner circle offers search of the content across the whole message store. It also supports nested searches within the current view or the previous search results. For example, a nested search of previous search results means that the new search is applied to the results of the previous search only. This way, a search can be refined step-by-step, until the user has found what he was looking for.

2.2.4 Assessment

One of the strong points of this solution (and also something not provided in the other prototypes) is the concept of the Mailed Items. As a user I often have searched for attachments in my email archive. The problem is especially annoying if there are several versions of the same document. Having the option of using a shared virtual repository within a certain group is certainly something I, as a user, would be interested in.

2.3 TaskMaster

The team at the Palo Alto Research Center presents a task-centered approach to email design [4].

2.3.1 Methods

The team conducted their user research using the following methods:

- **Study of Outlook and Eudora Users**
  This study was conducted in various phases. For the user the team recruited 11 people. During the testing, a filter was installed to save all in- and outgoing email. The research team interviewed the users about their experiences using Outlook and Fedora.

- **User study with Prototype**
  A group of 8 users were trained on the features of the TaskMaster prototype. They were then tested and interviewed on the subject

- **Prototype for users to use for real work with their own mail**
  The same group then used TaskMaster exclusively for their real work for two weeks

2.3.2 Problems Identified

The TaskMaster solution focuses on Task Management with the email client. The main problem identified is that existing email clients do not provide a satisfactory solution to task management. In many working environments email is used to manage tasks within a team.

- **Keeping Track of many Concurrent Actions**
  Users find it hard to keep track of their own tasks (to do's) and the tasks they expect others to do (expected to do's). Items drift out of sight and are forgotten.
- **Separating Important from not Important**
  The inbox quickly becomes cluttered with non-important messages.
- **Managing Activity Extending over Time**
  While some to do items can be immediately resolved others require time, e.g. while the user needs to wait for a reply from someone else first.
- **Managing Deadlines and Reminders**
  With current email clients it is hard for the users to manage the deadlines of their tasks.
- **Collating Related Messages, Attachments and Links**
  The problem is how to group messages that belong together. Automatic filtering can partially solve this problem. However, the research conducted by the team showed that users like to see the incoming email before it is stored somewhere else.
- **No Task Oriented Overview**
  Users would like a overview of their tasks without having to search through different folders.

2.3.3 Proposed Solution

The TaskMaster prototype focuses on solving the problem of managing tasks in email clients.

- **Thrasks: Threaded Task-Centric Collections**
  The main principle is that the key element is not the message but the task. *Thrasks* are semi-automatic collections of task items that are interdependent. Sometimes a single message corresponds directly to a single task. Mostly however a task is generated from a thread of emails, including links etc.
- **Incoming and Outgoing Messages Viewed Together**
  In a task related context, users need to keep track not only of their to do's but also of to do's assigned to others. For that reason TaskMaster displays ingoing and outgoing messages together.
- **Equality for All Content**
  TaskMaster does not give messages more importance than attachments

or links. While messages are often deleted, the attachments need to be filed and links bookmarked. Attachments and links are extracted from the messages and stored in their own right.

- **Task-centric Meta-information for All Items**
  Any item in TaskMaster can have meta data such as a deadline, reminder, action or a color code associated with it. It is no longer necessary for users to copy the task information into a separate calendar or task tracking tool. Color codes help the user to visually distinguish between the different types of tasks.

- **Aggregations of Information for an Overview**
  These mechanisms permit users to get a sense of their obligations and upcoming deadlines and to be able to contact relevant collaborators without spending time searching through thrasks and inspecting individual items. *Warning Bars* remind the user of the nearest upcoming task deadline. *Action clusters* represent clusters of actions associated with a thrask. *Task-Specific Contact lists* are task-centric popup lists. They contain the names and email addresses of all senders and recipients associated with items in the thrask.

2.3.4 Assessment

The paper was published in 2003 and stated that the authors planned to "'push some of our concepts further"'. So far there have been no new publications on Taskmaster and it would be interesting to see if and how the prototype has been extended. The TaskMaster solution focuses on providing task management functionalities. The key principle is the concept of *Thrasks* , where many messages in an email conversation can constitute a task. By moving away from the restriction of one message equals one task, TaskMaster gives the user a more powerful mechanism to manage his tasks. However, TaskMaster does not offer the mechanism of breaking tasks down into subtasks. When each of the subtasks is done, the original task could be marked as complete and removed from the todo list. Tasks would be much easier to manage that way.

2.4 Email Overload

The research into Email Overload [31] was conducted at the Lotus Research Center by Steve Whittaker and Candace Sidner. The term *Email Overload* refers to the situation that email clients are used for many additional functions such as task management and personal archiving. The problem arises from the fact that email clients were not designed to be used in this way.

2.4.1 Methods

The team conducted their user research using the following methods:

- **Study of NotesMail Users**
  NotesMail is the email component of Lotus Notes. The 20 participants were Lotus staff representing the different levels at work: high-level managers, first level managers, professional workers and administrative staff.

The researchers collected quantitive data about the mailbox of the users and conducted interviews with the participants.

- **User study with Prototype**
  A group of 8 users were trained on the features of the TaskMaster prototype. They were then tested and interviewed on the subject

- **Prototype**
  A group of users used the prototype for real work with their own email. The same group then used TaskMaster exclusively for their real work for two weeks.

2.4.2 Problems Identified

The researchers identified the following problems as symptoms of the email overload problem

- **Full Inbox**
  Many users experience a backlog of unanswered email - even though most dedicate a large proportion of their working hours to keep on top of their email. However, unanswered messages are not the only cause for the full inbox. The research team identifies the main reasons to be: (a) the Inbox is also a task manager, with users being reminded of their tasks and (b) users find it hard to file their messages into folders. Many users find that the folder structure is not very helpful when they later try to retrieve a message.

- **Retrieving Information is Difficult**
  Finding a specific email or email conversation takes time and is error prone. Users often do not remember in which folder they stored the particular email. This makes them more reluctant to use the folder structure, with most email ending up in the Inbox (see above).

2.4.3 Proposed Solutions

- **Communication Management**
  The authors suggest marking messages from the same conversation with a thread ID. This allows users to view related messages together and also to trace back through conversations. Essentially the user can then view messages by thread. The authors see several benefits to this approach: (a) the user can determine the status of the communication: i.e. does he "'owe"' someone an answer or does someone else owe him one? (b) It can help declutter the inbox. The user can keep the most recent message of the thread in the inbox, as a reminder of the ongoing conversation. The other messages can be archived in a folder and retrieved quickly when necessary.

- **Filing based on Context**
  The researchers suggest using information retrieval techniques to cluster

semantically related messages (i.e. about a common subject) automatically. It would be implemented analogously to the thread ID. However, they also stress that the automatic filing must not be done without the user having seen the message first.

- **Task Management Support**
  One of the key points to task management is clearing the Inbox of unnecessary items, so that only the important messages remain. This helps users focus on the tasks. Additionally, email clients need the following functionalities to support task management: (a) a user must be able to mark messages as "'requiring action"', with such items easily visible. (b) *Program reminders* provide a solution to the problem of tasks that cannot be done immediately. With program reminders it would be possible to program the item to re-appear as a task as the deadline approaches.

2.4.4 Assessment

All of the papers reviewed in this section cite the "'Email Overload: exploring personal information management of email"' paper. It seems that the principles suggested by Whittaker et al. have been taken up by the email client research community. However, most email client prototypes focus on one of the aspects.

Chapter 3

Existing Solutions

This section analyzes the existing solutions for email clients. They can be grouped into three categories:

- email clients with graphical interfaces
- email clients that are console-based
- email clients that are part of web browsers.

The focus on studying the existing solutions will be on the graphical interface based email clients. These also are the largest group of the three categories.

3.1 Existing Email Clients

Based on the list compiled in Wikipedia [32], the email clients to be analysed were selected by applying the following criteria: The email client must

- have a current stable version that is less than 3 years old.
- not have ceased to be developed
- run on one of the operating systems: Windows, Mac OS, UNIX/Linux.
- provide a graphical user interface.

These criteria are to ensure that the email clients to be analysed qualify as existing solutions. The first two state that the implementations reflect the current work in the field of email clients. As part of the research into the usability of the email clients, user experiences will be taken into account. The third criteria is to make sure that the email clients have a reasonably large user base. For instance the Acme email client runs only on the operating system Plan 9, an operating system that is not widely used.

The following email clients resulted from this process:

Email client	**Developed by**	**Current version**	**Operating system**
Apple Mail	Apple	2.0.5 / October 2005	Mac OS X
Microsoft Outlook	Microsoft	as part of Office 2003	Windows, OS X
Lotus Notes	IBM	7.0 / August 2005	Windows, OS X
Mozilla Thunderbird	Mozilla	1.0.7 / September 2005	Windows, OS X, Linux, UNIX
Eudora	Qualcomm	7.0.0.16 / November 2005	Windows, OS X
KMail	KDE	1.8.2 / July 2005	Linux, UNIX, Mac OS X
Balsa	gnome	2.3.2 / May 2005	Linux
Groupwise	Novell	7 / September 2005	Windows, Linux, Mac OS X
Novell Evolution	Ximian	2.2.3 / July 2005	Linux, UNIX
The Bat!	Ritlabs	3.64 / December 2005	Windows
Pegasus	David Harris	4.3 / December 2005	Windows

Table 3.1: Existing Email Client Solutions

Apple Mail [18]
Apple Mail (also known as just Mail) is an email client developed by Apple. It is included in Mac OS X. Apple Mail implements SMTP, POP3, and IMAP protocols, and supports .Mac and Exchange via IMAP. It is released under a proprietary license.

Microsoft Outlook [22]
Microsoft Outlook is the email client included in Microsoft's Office package. Outlook implements SMTP, POP3, and IMAP protocols. It is released under a proprietary license.

Lotus Notes [21]
Lotus Notes is a client-server collaborative software system and e-mail email client developed by IBM. It implements SMTP, POP3, and IMAP protocols. Lotus Notes is released under a proprietary license.

Mozilla Thunderbird [27]
Mozilla Thunderbird is an email client developed by the Mozilla Foundation. It implements SMTP, POP3, and IMAP protocols. Mozilla Thunderbird is available under the MPL/GPL licenses.

Eudora [7]
Eudora is an email client that runs on Windows and Mac OS. It implements SMTP, POP3, and IMAP protocols. Eudora is released under a proprietary

license.

KMail [14]
KMail is the e-mail client of the KDE Desktop Environment. It implements SMTP, POP3, and IMAP protocols. KMail is available under the GPL license.

Balsa [2]
Balsa is an email client that runs under the GNOME Linux user interface. It builds upon other open source packages. Balsa implements SMTP, POP3, and IMAP protocols. It is available under the GPL license.

Groupwise [9]
Groupwise is a collaborative software product released by Novell. Apart from email and calendaring, Groupwise also offers instant messaging and document management. It implements SMTP, POP3, and IMAP protocols. It is available under proprietary licenses.

Novell Evolution [8]
Novell Evolution is a personal information management system and is included in the GNOME user interface. It implements SMTP, POP3, and IMAP protocols. Novell Evolution development is primarily sponsored by Novell Evolution and is available under the GPL license.

The Bat! [3]
The Bat! is an email client for the Windows operating system. It implements SMTP, POP3, and IMAP protocols. The Bat! is shareware.

Pegasus [23]
The Pegasus email client is an email client developed by David Harris. It implements SMTP, POP3, and IMAP protocols. Pegasus is available as freeware.

3.2 Email Client Features

We then compared the solutions the email client offers for:

1. Visualization Techniques
2. Task Management
3. Search Mechanisms
4. Special Features.

The Visualization Techniques include mechanisms to visualize threads, contacts and messages. They should provide the user with an intuitive "at-a-glance" view. An example for a thread visualization technique is to display a message thread as a tree.
The Task Management consists of techniques for the users to manage their todo list. Examples are calendars, tasks, reminders and meeting schedules.
The search mechanisms show what possibilities users have to efficiently search their Inbox. The focus is on the granularity, i.e. in how much detail a query be can specified.
The final section Special Features lists features that are not among the standard features offered by all of the email clients analysed. The standard features that apply to all email clients are listed in figure 3.1. Although Search is also a standard feature, the different email clients offer different search mechanisms. They are analysed separately in (3). For the email client feature comparison, please refer to the table in the Appendix.

Standard Feature	**Description**
Spam Filtering	Bayesian filtering mechanisms
User-defined Filters	Messages are rerouted to folders based user-defined rules
Address Book	Manage contacts
Attachments	Send, receive and forward attachments
Encryption	PGP and/or SSL
Spell Checker	Check message for spelling

Table 3.2: Standard Features for Email Clients

3.2.1 Assessment

In general, while the email clients boast a vast collection of features, there is no visible strategy. There appears to be no focus on key concepts (such as task management or visualization techniques) and on building the features around these key areas. Instead there seems to be a "war of the features" between the different email clients. Most of the product websites include a long list of features, with the emphasis on: the more, the better. The Visualization section is still pretty empty. Some email clients such as KMail, GroupWise and Pegasus offer color schemes to mark messages. So far there are no implementations of the concepts suggested by the Remail research group [25]. Not many email clients offer task management functionalities. Those that do provide todo and task checklists. However, there is no solution where a task can be directly extracted and generated from a message. In our experience, a tool that would provide such a functionality would be extremely helpful. If an email includes a task that cannot be immediately dealt with, it can be hard not to lose track of it. The Search functionalities are very similar across the different email clients. Surprisingly however, only Pegasus offers the search by date and size. Among the Special Features, some interesting functionalities are offered. Microsoft Outlook contains many different features, but most of those require additional server software. In the comparison of the email clients some elements we find particularly interesting are marked:

- **BossWatch:**
  Eudora offers a solution to the *reply-to-all* problem. This term refers to the problem that users inadvertently use the reply to all button. I have often received such emails, even when the original author had added "please do not reply to all". Another reason for using the reply to all button could be that it is often fussy to select the addresses for sendto list. For example in Outlook, this requires clicking on every address and selecting delete or add. So often users just reply to all. The BossWatch solution focuses on warning the user about special email addresses, e.g. the boss's (hence the name). This principle could also be applied to large mailing lists (such as all@adomain.com).

- **Smart Sorting Office:**
  The basic idea of this solution, filtering email messages and rerouting them to folders, is not new. The Smart Sorting Office by TheBat! however offers this concept in a more comprehensive way than other solutions. It filters mail according to rules and reroutes the messages to the folders. The Smart Sorting Office can also handle replies by templates, auto replies, automatic forwarding and printing. What we like about the concept is that these features are consolidated into a single module.

- **Circulation Messages:**
  The circulation messages idea by Pegasus is similar to the concept of office mail. In office mail (for instance, when there is a procedure of which department must review which kind of document) there are often envelopes with a list of receivers. Each person on the list passes it on to the next person on the list, when they have finished with the document.

I really like this idea. At university my group of friends usually meets up a few times a month. When it comes to organising a place and time, someone sends out a message with a suggestion. What usually follows is chaos, i.e. everyone replies based on different emails of the conversation. With this solution, the original sender defines a list of receivers and the messages is then passed on one by one.

3.3 Email Client Survey

This section describes the results of the email client survey. 144 users at the insurance companies SwissRe and Winterthur participated. First, the results of the answers to the questions are given. These results are then analyzed in a qualitative manner. Finally, we show how these answers are integrated in the use scenario strategies.

3.3.1 Email Client Survey Results

General Questions

- 1. Which email client do you use at work?

	Response Percent	Response Total
Microsoft Outlook	83.1%	118
Lotus Notes	13.4%	19
Apple Mail	0.7%	1
Mozilla Thunderbird	5.6%	8
Other	2.9%	4
	Total Respondents	144
	skipped this question)	2

Table 3.3: Results of question 1

Of those 4 users who chose "Other", the following answers were given:

  - Web-Based (via Browser)
  - mutt
  - pine
  - kmail

- 2. Which business area do you work in?

	Response Percent	Response Total
IT	42.9%	60
Management	2.1%	3
Sales	24.3%	34
Administration	20.7%	29
Other	14.3%	20
	Total Respondents	140
	skipped this question)	4

Table 3.4: Results of question 2

The areas of work given by the users that chose "Other" are mainly insurance related.

Attention Management

- 3. Do you always read every email you receive? If not, please specify the type of messages you do not read (e.g. cc messages).

	Response Percent	Response Total
Yes	41.4%	55
No	58.6%	78
	Total Respondents	133
	skipped this question)	11

Table 3.5: Results of question 3

The 78 users that selected "No" in the question above gave the following answers (multiple answers are possible):

Response	Response Percent	Response Total
Spam emails	69.2%	54
Emails where I am addressed as cc	7.8%	10
I decide based on the subject	14.1%	11
General information (about the company, Newsletters, Mailing lists)	15.4%	12

Table 3.6: Specific answers to question 3

- 4. When do you delete messages from your Inbox?

	Response Percent	Response Total
After answering the message	35.3%	47
After filing the message	33.8%	45
Every once in a while	49.6%	66
When the Inbox is full	12%	16
Other	20.3%	27
	Total Respondents	133
	skipped this question)	11

Table 3.7: Results of question 4

The majority of the 27 users that chose "Other" answered that they use a combination of the strategies stated above. Three users stated that their emails are automatically archived from their Inbox.

Task Management

- 5. Has it ever happened that you forgot a task that you received per email? If yes, what do you think the reasons are for this happening?

	Response Percent	Response Total
No	56.3%%	71
Yes	43.7%	55
	Total Respondents	126
	skipped this question)	18

Table 3.8: Results of question 5

The 55 users that selected "Yes" saw the reason for the problem of forgetting tasks in the following areas (multiple answers are possible):

Response	Response Percent	Response Total
Too many emails	45.5%	25
Too many applications handling tasks	1.8%	1
Emails move out of sight to quickly	16.4%	9
Too many tasks	5.5%	3
Failure to add task to task list	5.5%	10
No follow up by the sender	3.6%	2

Table 3.9: Specific answers to question 5

- 6. How do you manage your tasks?

	Response Percent	Response Total
Email client todo list	32.8%	42
As messages in the email client	21.9%	28
List on paper	41.4%	53
Other	28.9%	37
	Total Respondents	128
	skipped this question)	16

Table 3.10: Results of question 6

The 37 users that selected "Other" in the question above, use the following strategies to manage their tasks:

- Excel
- Planning tools (MS Project)

- Calendar
- Brain
- Palm
- Wiki

Search

- 7. How do you search your Inbox?

	Response Percent	Response Total
By navigating the folders	23.3%	30
Using the search functionality	17.1%	22
Combination of the above	26.4%	34
By sorting the Inbox	55.8%	72
Other	10.9%	14
	Total Respondents	129
	skipped this question)	15

Table 3.11: Results of question 7

Useful features

- 8. Which features would you find useful in an email client?

	very useful	useful	not useful	Respondent Total
Separate Inboxes (e.g. private/work)	34%	41%	25%	117
Todo list with priorities	30%	46%	24%	119
Task with subtasks	8%	48%	44%	111
Generate task from message	27%	42%	31%	109
Automatic archiving after certain time	24%	36%	41%	118
Colours to mark messages	40%	44%	20%	121
Total Respondents				122
skipped this question)				22

Table 3.12: Results of question 8

Comments

- 9. What is the worst part about email clients?

Total Respondents	79
skipped this question	65

Table 3.13: Results of question 9

The top ten of the worst part about using email clients are:

1. Nothing
2. Too many functions. They are too hard to master and use on a regular basis.
3. The email client is too slow
4. The stability of the email client
5. Insufficient search functionality
6. The archiving functionality

- 10. Additional comments and remarks

Total Respondents	19
skipped this question	125

Table 3.14: Results of question 10

- i don't understand why tasks are so important in the survey. i'm perfectly happy with having different applications for my email client and for my task management. however, if the email client were able to automatically extract tasks from email messages and it could only do it for a proprietary format, then it would make sense

- A task management tool is different to email client. ToDo lists do only work if they are continuously managed which cost a lot effort.

3.3.2 Analysis of the results

The survey aimed at gathering information about how users handle their email. Users from different business areas participated in this survey. While users working in IT constitute the majority, there are also many others from non-technical business areas. As a result, the answers to the survey are a good mixture of technical and usability ideas and feedback.

The key problem identified by the users in terms of email is the large amount of messages received on a daily basis. As a consequence of this problem, 58.6% of all users do not read all of the messages they receive. It is not surprising that most users do not read spam - it does however point to the fact that SPAM filtering at the companies has room for improvement. A substantial part of the users decides whether to read a message or not based on the subject.

Another key problem seems to be how to keep the state of the Inbox consistent. Many users find adding emails to folders tedious and time consuming and that some messages cannot be easily categorised (i.e. should belong to more than one email folder). As a consequence, the Inbox is often used as a repository: anything to be done is left in the Inbox, the rest is either deleted, filed or archived. The answers to question 4 however point to the fact that almost half of the users do not handle their Inbox in a systematic way: 49.6% of the users delete emails from their Inbox every once in a while. 35.3% delete the message after answering it, another 33.8% after filing it. It appears that the key to keeping track of messages, i.e. the solution to "the email moving out of sight", lies in organizing the Inbox.

Question 6 led to surprising results: 41.4% of the users manage their tasks on a list of paper. Only 32.8% of the users manage their tasks using the email client task list. These approaches entail copying the information from the messages containing a task to the task or paper list. A further 21.9% of the users does not copy the information to somewhere else but stores the tasks as messages.

The search functionality does not seem to be widely supported: only 17.1% use the search functionality to search their inbox, while 55.8% search their inbox by sorting the Inbox. This may be due to the fact that often a user does not have precise information about the email he is searching for. For instance, the subject and the sender may be unknown (especially if emails were exchanged between a group of people: the user would have to search for all senders). So a strategy could be not to focus on improving the search functionality by adding new features, but to look for other ways of supporting the users.

Of all the features given in question 8, all were deemed useful, except for automatic archiving after a certain time. "Separate Inboxes" and "colours to mark messages" missed the mark for very useful by only a few votes.

Chapter 4

Use Cases

This chapter contains the use cases. The aim of these use cases is to identify the situations that are problematic when applied to existing email clients. The email clients reviewed are taken from the chapter "Existing Solutions".

First, we propose a template for the use cases. It contains the elements summary, frequency of use, actors, use case steps, and illustrations, and follows the concept of *contracts* with pre- and postcondition elements [19]. The use case is represented in tabular form and is supported by an illustration where applicable. We introduce a categorization of the use cases and provide the deferred - descendant use case relationships.

As a next step, the common behaviour of a user for a certain functionality is combined into a deferred use case. The functionalities are grouped into Task Management, Visualization, Search, and Attention Management. The use case steps described in the deferred use case are common to all descendant use cases, that inherit from the deferred use case. Finally, we introduce a set of descendant use cases for each of the functionalities and email clients. These use cases describe the specific behaviour of an email client for a given functionality. For example, the descendant use case "search by date with Pegasus" describes the use case of searching by date using the email client Pegasus.

4.1 Use Case Design

The template for the use cases is defined as follows:

Element	Description
Summary	Brief summary of the use case
Frequency of Use	How often is the use case executed?
Actors	Parties outside the system that interact with the system
Preconditions	State the necessary conditions that must be fulfilled before the user can start.This precondition applies to the whole use case (not for the single steps).
Use Case Steps	Steps an actor takes when interacting with the email client
Exceptions	Exceptional cases in the interaction
Postconditions	Situation after the use case steps were executed. This post-conditions applies to the whole use case (not for the single steps).
Illustrations	UML diagrams that describe the interactions of the system with its Actors. The illustrations are omitted where there is only a single Actor involved.

Table 4.1: Use Case Template

4.1.1 Use Case Categorization

Categorizing the use cases ensures a systematic approach to modeling use cases and provides the reuse mechanism, as known from object-oriented design. The Springer *Expert's Voice* paper [13] focuses on modeling use cases for object-oriented design. The author suggests categorising use cases as follows:

- **Concrete use cases** that can be instantiated (as opposed to abstract ones)
- **Generalization use cases** that have generic sequence of actions that can be specialized by other use cases.
- **Extension use cases** that add behavior to an existing (or presumed) use case, without changing the original use case. The extension use case is the use case that extends the behavior of an generalization use case. This corresponds to sub-/superclass behavior in object-oriented design.
- **Inclusion use cases** that describe behavior that can be used by other use cases. The inclusion use case uses the behavior of the use case it includes. In UML, this behavior is known as *association* , in object-oriented design as client-supplier relationship.

For our work, our categorisation is based on the categorisation by I. Jacobson. We use Eiffel terminology to illustrate the mapping between use cases and

the implementation.

We will be using the following categorisation:

- **Effective use cases** that can be instantiated (as opposed to abstract ones). This category corresponds to the concrete use cases in the categorisation by I. Jacobsen.
- **Deferred use cases** that have generic sequence of actions that can be specialized by other use cases. This category corresponds to the generalization use cases in the categorisation by I. Jacobsen.
- **Descendant use cases** that add behavior to an existing (or presumed) use case, without changing the original use case. The extension use case is the use case that inherits the behavior of a deferred use case. This category corresponds to the extension use cases in the categorisation by I. Jacobsen.
- **Client use cases** that describe behavior that can be used by other use cases. The client use case uses the behavior of the use case it includes. In UML, this behavior is known as *association* , in object-oriented design as client-supplier relationship. This category corresponds to inclusion use cases in the categorisation by I. Jacobsen.

The goal of our design is to keep the inheritance relationship between the use cases as simple as possible (but as complex as necessary). Use cases can have multiple inheritance relationships where required.

Our design includes object-oriented design concepts and advanced features in the use cases. These concepts are pre- and postconditions, and the inheritance between use cases. The use cases will be grouped by functionality: task management, visualization, search, attention management, and special features. The idea is to have a deferred use case for each of the functionalities. It defines the basic common behavior of a use case. For each of the email clients of the existing solutions list, a use case will inherit from this deferred use case, the resulting use case is a descendant use case. It can have additional elements and/or overwrite some of the implementations of the deferred use case.

As a guideline, deferred use case steps define *what* the user does. The descendant use cases then define *how* these steps are executed. For example, the deferred use case states as use case step: the user opens the search menu. The descendant use cases then define for each of the email client, how the search menu is accessed.

Our approach is to define a few (min 1, max 5) deferred use cases for each of the functionality groups: task management, visualization, search, attention management and special features. These use cases define the basic user interaction for this functionality. They are then inherited by descendant use cases for each of the email clients.

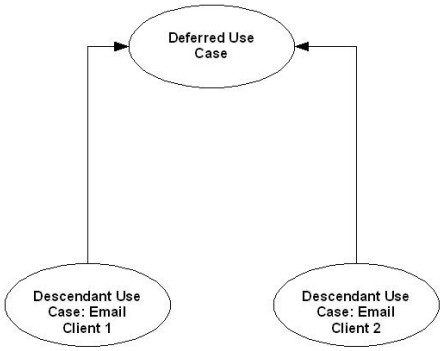

Figure 4.1: Use Case Inheritance

4.1.2 UML Diagrams

The use cases are described using the template defined above, i.e. in textual representation. Additionally, the use case is supported by UML use case diagrams. The use case diagrams are part of the behavior diagram category, describing what must happen in the system modeled. In our representation, the goal is to visualizing the interaction between the Actors and the system. The UML use case representation used is based on the representation suggested in [12] and was slightly modified to suit our case. For this purpose, the following elements are used:

Oval : represents a use case

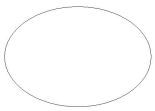

Figure 4.2: Use Case

Stick figure : represents an Actor

Box : represents the boundaries of the system.

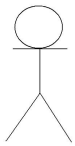

Figure 4.3: Actor

Figure 4.4: System Boundaries

Connecting Arrow : represents the interaction between an Actor and a use case.

Figure 4.5: Interaction between Actor and Use Case

Connecting dotted line Arrow : represents the relationship between the use cases. This can be *includes* or *inherits* . A use case can have several relationships and can inherit multiple classes.

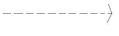

Figure 4.6: Relationship between Use Cases

4.2 Use Cases

This section describes the use cases we derived from the existing email client solutions. First, we define deferred use cases that describe the overall behaviour of a user interacting with a particular functionality. The descendant use cases then inherit from the deferred use cases. For each of the email client solutions described in the previous section a descendant use case is defined.

The descendant use cases in this section describe the behaviour of the existing email client solutions: Apple Mail, Microsoft Outlook, Lotus Notes, Mozilla Thunderbird, Eudora, KMail, Balsa, Groupwise, Novell Evolution, The Bat!, and Pegasus. Each of these descendant use cases inherits from a deferred use case that describes the behaviour that is common to all of the email clients.

At the beginning of each section the inheritance relationship with the deferred use cases is described. If not further specified, this relationship is valid for all descendant use cases in the same section. For instance, all descendant use cases in the section "create task from email received" inherit from the deferred use case "create task from email received". For each group of descendant use cases, an assessment is provided.

Not all of the email clients provide the functionalities specified in the deferred use case. For instance, certain email clients do not offer task management functionalities to the user. If this is the case, then the descendant use case is omitted.

4.2.1 Use Cases: Task Management

This section contains the Use Cases that are associated with task management. Task management is concerned with providing the user with mechanisms to manage his tasks within the email client. Tasks are often sent and received per email, especially in the working environment. The following use cases are described:

- **A. Create task from email received**
  This use case describes the steps a user takes upon receiving a task per email. The focus is on how the user can generate a task as conveniently as possible.

- **B. Manage tasks**
  The Manage tasks use case focuses on how a user can gain an overview of his tasks. For instance, what different views of the pending tasks are offered by the email client etc.

- **C. Complete tasks**
  This use case shows the steps a user takes upon completing a use case.

A. Create Task from Email Received

All descendant use cases in this section A inherit from the deferred use case: Create Task from Email Received.
This deferred use case defines the common behaviour of a user to create a task from an email received. This common behaviour includes steps that the user takes that are independent of the particular email client.

Deferred Use Case: Create Task from Email Received

Element	**Description**
Summary	The use case describes the steps the *Receiver* takes to create a task from an email. This refers to tasks that are not immediately done by the user, but that need to be stored for later reference and reminders.
Frequency of Use	The use case is executed daily
Actors	The *Sender* is the user that sends the email containing the task. The *Receiver* is the user that receives this email.
Preconditions	The email client must be running. The *Receiver* must receive an email containing a task.
Use Case Steps	The Receiver: **1.** Opens the email message. **2.** Reads the email and encounters a task. This task is either explicitly requested by the Sender (e.g. "'please send me the report x by deadline y."') or implicitly to the Receiver (e.g. "'I am currently receiving http errors when accessing the intranet"')
Exceptions	Exceptions apply where the email client does not offer task management. In this case, the inheriting use case is marked invalid.
Postconditions	The task is stored and the Receiver will be reminded of it before the deadline is reached.

Table 4.2: Deferred Use Case: Create Task from Email Received

Illustrations

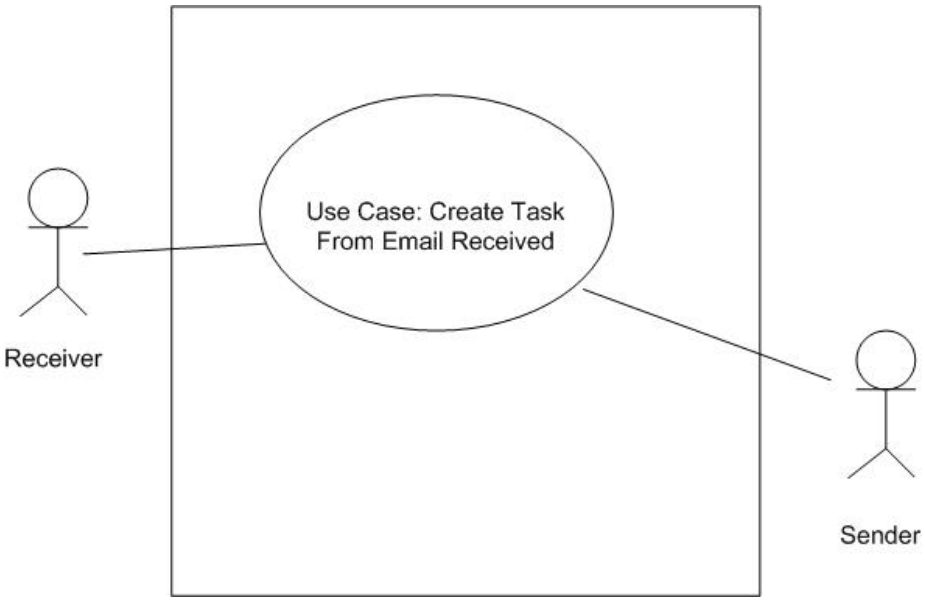

Figure 4.7: Create Task from Email Received

The descendant use cases inherit all fields from the deferred use case.

1. **Descendant Use Case: Create Task from Email Received with Microsoft Outlook**

Element	**Description**
Use Case Steps	The Receiver: **3.** Clicks on the Tasks icon **4.** Creates a new task. **5.** Fills in the details (subject, status, due date, completeness, categories.) by copying from the email message. **6.** Stores the task.

2. **Descendant Use Case: Create Task from Email Received with Lotus Notes**

Element	**Description**
Use Case Steps	The Receiver: **3.** Clicks on the task item. **4.** Fills in the details (subject, due date, status, and assigned to) by copying from the email message. **5.** Stores the task.

3. **Descendant Use Case: Create Task from Email Received with GroupWise**

Element	**Description**
Use Case Steps	The Receiver: **3.** Clicks on the message containing the task. **4.** Drag and drop the message to the CheckList folder. **5.** Click on the CheckList item. **6.** Open the message in the CheckList menu. **7.** Add the details such as due date and the order. **8.** Saves the adapted message.

4. **Descendant Use Case: Create Task from Email Received with Novell Evolution**

Element	**Description**
Use Case Steps	The Receiver: **3.** Selects the task menu. **4.** Fills in the details (Summary, description, due dates, priorities, and categories) from the email message. **5.** Stores the task.

B. Manage Tasks

All descendant use cases in this section B inherit from the deferred use case: Manage Tasks.
The Manage tasks use case defines the common behaviour of a user to manage his tasks. For instance, how the user can gain an overview of all pending tasks. This common behaviour includes steps that the user takes that are independent of the particular email client.

Deferred Use Case: Manage Tasks

Element	**Description**
Summary	The use case describes the steps the actor takes to manage his tasks. This entails task reminders and gaining an overview of pending tasks.
Frequency of Use	The use case is executed daily, on average once or twice daily
Actors	The principle Actor is the user handling the email client.
Preconditions	The email client must be running. The *user* has pending tasks to manage.
Use Case Steps	The User: **1.** Clicks on the location of the tasks.
Exceptions	Exceptions apply where the email client does not offer task management. In this case, the inheriting use case is marked invalid.
Postconditions	The user has gained an overview of the pending tasks.

Table 4.3: Deferred Use Case: Manage Tasks

1. **Descendant Use Case: Manage Tasks with Microsoft Outlook**

Element	**Description**
Use Case Steps	The User: **2.** Chooses between the different views: - Simple list: shows subject and due dates - Detailed list: shows all details - Active tasks: shows active tasks - Next seven days: shows all tasks with the due dates in the next seven days - Overdue tasks: shows all tasks with due date earlier than current date - Assignment: shows the owners of the tasks - By person responsible: shows tasks sorted by person responsible - Completed tasks: shows completed tasks - Task time line: shows graphical representation of time line

2. **Descendant Use Case: Manage Tasks with Lotus Notes**

Element	**Description**
Use Case Steps	The User: **2.** Chooses between the different views: - All To Dos: show all to dos - Personal: show all tasks that are assigned to the user - Group: show all tasks that are assigned to someone in the group - Status: show all tasks according to status

3. **Descendant Use Case: Manage Tasks with GroupWise**

Element	**Description**
Use Case Steps	The User: **2.** Sorts the CheckList view by sorting the columns.

4. Descendant Use Case: Manage Tasks with Novell Evolution

Element	**Description**
Use Case Steps	The User: **2.** Chooses between the following views: **-** date **-** priority **-** deadlines

C. Complete Tasks

All descendant use cases in this section C inherit from the deferred use case: Complete Tasks.
This use case defines the common behaviour of a user to complete tasks. This common behaviour includes steps that the user takes that are independent of the particular email client.

Deferred Use Case: Complete Tasks

Element	**Description**
Summary	The use case describes the steps the user takes upon completion of a task. This includes removing the task from the todo list.
Frequency of Use	The use case is executed daily upon completion of a task.
Actors	The principle Actor is the user handling the email client.
Preconditions	The email client must be running. The *user* has completed a task.
Use Case Steps	The User: **1.** Clicks on the location of the tasks.
Exceptions	Exceptions apply where the email client does not offer task management. In this case, the inheriting use case is marked invalid.
Postconditions	The user has removed a completed task from the todo list.

Table 4.4: Deferred Use Case: Complete Tasks

1. **Descendant Use Case: Complete Tasks with Microsoft Outlook**

Element	**Description**
Use Case Steps	The User: **2.** Ticks the box next to the task.

2. **Descendant Use Case: Complete Tasks with Lotus Notes**

Element	**Description**
Use Case Steps	The User: **2.** Clicks on the Action Bar **3.** Selects Save and Close

3. **Descendant Use Case: Complete Tasks with GroupWise**

Element	**Description**
Use Case Steps	The User: **2.** Clicks on the CheckList. **3.** Checks the box next to the task to be deleted.

4. **Descendant Use Case: Complete Tasks with Novell Evolution**

Element	**Description**
Use Case Steps	The User: **2.** Removes the task from the list

4.2.2 Assessment: Task Management

Many of the email clients analysed do not offer task management at all. Task management is not a core concept of email and can be done with tools external to the email clients. However, email messages and tasks are usually connected, especially in work environments. For instance, meeting requests or task requests arrive per email. In those email clients that do not offer task management functionalities, a separate application needs to be opened and the user has to copy the details from the email message by hand. Where the email client offers these functionalities, a user still has to open a new menu and fill in the task details by hand. In our opinion, the connection email message - task entry should be more closely and conveniently coupled.

4.2.3 Use Cases: Visualization

This section contains the use cases associated with visualization. The visualization techniques should help the email client user identify the structure of his email. The visualisation techniques go beyond the usual graphical user interface presentation. For example, a tree view of a conversation thread is such a visualization. The following use cases are described:

- **A. Thread view**
  The descendant use cases in this section describe how the email clients analysed provide a view of the message threads. The focus is on how the message thread is visualized and on how easy it is for the user to choose this view.

A. Thread View

All descendant use cases in this section A inherit from the deferred use case: Thread View.
This deferred use case describes the common behaviour of a user to gain an overview of a conversation thread. This common behaviour includes steps that the user takes that are independent of the particular email client.

Deferred Use Case: Thread View

Element	**Description**
Summary	The use case describes the steps the user takes to gain an overview of a conversation thread. He would like to see how the messages are interconnected.
Frequency of Use	The use case is executed on a daily to weekly basis.
Actors	The principle Actor is the user handling the email client.
Preconditions	The user has a set of messages in his inbox that form a conversation thread.
Use Case Steps	The User: **1.** The user selects a conversation thread to be visualized.
Exceptions	Exceptions apply where the email client does not offer visualization. In this case, the inheriting use case is marked invalid.
Postconditions	The user has gained an overview of the conversation thread.

1. **Descendant Use Case: Thread View with Microsoft Outlook**

Element	**Description**
Use Case Steps	The User: **1.** Selects the View menu **2.** Selects the Arrange By submenu **3.** Selects Conversation from the submenu.

2. **Descendant Use Case: Thread View with Mozilla Thunderbird**

Element	**Description**
Use Case Steps	The User: **1.** Selects the View menu **2.** Selects the Sort By submenu **3.** Selects Threaded from the submenu

3. **Descendant Use Case: Thread View with KMail**

Element	**Description**
Use Case Steps	The User: **1.** Selects a message of the conversation to be visualized. **2.** Selects the View Menu **3.** Selects Expand Thread

4. **Descendant Use Case: Thread View with The Bat!**

Element	**Description**
Use Case Steps	The User: **1.** Selects the View menu. **2.** Selects the View threads by menu **3.** Selects References (standard)

4.2.4 Assessment: Visualization

Again, many email clients do not offer thread visualization at all. Those email clients that do, provide a textual representation of the message thread. However, this does not really qualify as a visualization technique such as suggested by [25]. Representation in email clients is still implemented on a textual level. We believe that visualization techniques such as graphs, color schemes, charts, etc. could provide means to gain a quick and intuitive overview over the mailbox. These techniques could also be combined with attention management schemes. For the participants in the email client survey, a key problem they face with email clients is the overwhelming amount of information in their mailbox. Visualization techniques could solve a part of this problem.

4.2.5 Use Cases: Search

This section contains the use cases associated with the search functionality. The review of the email client features (weekly report 4th February) showed that all email clients offer search by subject, to, and from fields and search by content. The following use cases focus on features that are not (or not widely) covered in existing solutions. The following use cases are described:

- **A. Search by date**
  This section describes the use cases for searching by date.

A. Search by Date

All descendant use cases in this section A inherit from the deferred use case: Search by Date.
This deferred use case describes the common behaviour of a user when searching for an email message by date. This common behaviour includes steps that the user takes that are independent of the particular email client.

Deferred Use Case: Search by Date

Element	**Description**
Summary	This use case is about finding an email message based on the date.
Frequency of Use	It is executed on a regular basis, daily to weekly.
Actors	The user handling the email client is the principal actor.
Preconditions	The email client is running. The user has a date and wants to search for messages associated with this date
Use Case Steps	The user: **1.** Calls the search facility. **2.** Enters the date that he wants to search for. **3.** If there are search results: the user selects the message he is searching for from the result set **4.** Else: the user ends the search.
Exceptions	Exceptions apply where the inheriting use case does not offer the search by date functionality. In this case the inheriting use case is marked invalid.
Postconditions	The user has found the messages associated with the date, if applicable.

1. **Descendant Use Case: Search by Date with Pegasus**

Element	**Description**
Use Case Steps	The user: **1.** Calls the search facility. **2.** Enters the date that he wants to search for. **3.** If there are search results: the user selects the message he is searching for from the result set **4.** Else: the user ends the search.

4.2.6 Assessment: Search

All of the email clients analysed provide a wide range of search functionalities and indexing structures. Of the four areas Task Management, Visualization, Search and Attention Management, Search is the area that is covered the most by the email clients we analysed. Only Pegasus, however, offers the search by date. The reasons for omitting this feature are not known. We believe that the date of a message is a core field and should be included in the search mechanisms.

4.2.7 Use Cases: Attention Management

This section contains the use cases associated with attention management. It focuses on the functionalities that should help the user manage his attention, for instance by separating important from unimportant information. The following use cases are described:

- **A. Separating private from work email**
  The descendant use cases in this section describe how a user can separate private from work email using the email clients analysed.
- **B. Out of office reply**
  The descendant use cases in this section describe how a user can manage out of office replies using the email clients analysed.
- **C. Archive messages**
  The descendant use cases in this section describe how a user can archive email messages using the email clients analysed.

A. Separating private from work email

All descendant use cases in this section A inherit from the deferred use case: Separating Private from Work Email. The actor receives a random mixture of private and work related emails. The steps chosen below are to reflect this situation.

This deferred use case describes the common behaviour of a user when separating private from work-related email. Where permitted by office policy, work email addresses are often used for private communication as well (especially where web mail such as hotmail, gmx etc. is blocked). This common behaviour includes steps that the user takes that are independent of the particular email client.

Deferred Use Case: Separating Private from Work Email

Element	**Description**
Summary	This use case describes the steps a user takes to manage private and work email.
Frequency of Use	It is executed daily, whenever the user receives a new message.
Actors	The *user* handling the email client, the *private sender* and the *work sender* . The private sender is a contact that communicates with the user about non work-related topics. The work sender communicates with the user about work-related topics.
Preconditions	The email client must be running. The user receives email messages from private and work senders.
Use Case Steps	The user: **1.** Receives a random mixture of private and work related email messages.
Exceptions	There are no exceptions for the abstract use case.
Postconditions	The user has an overview of the message received from private and work senders.

Illustrations

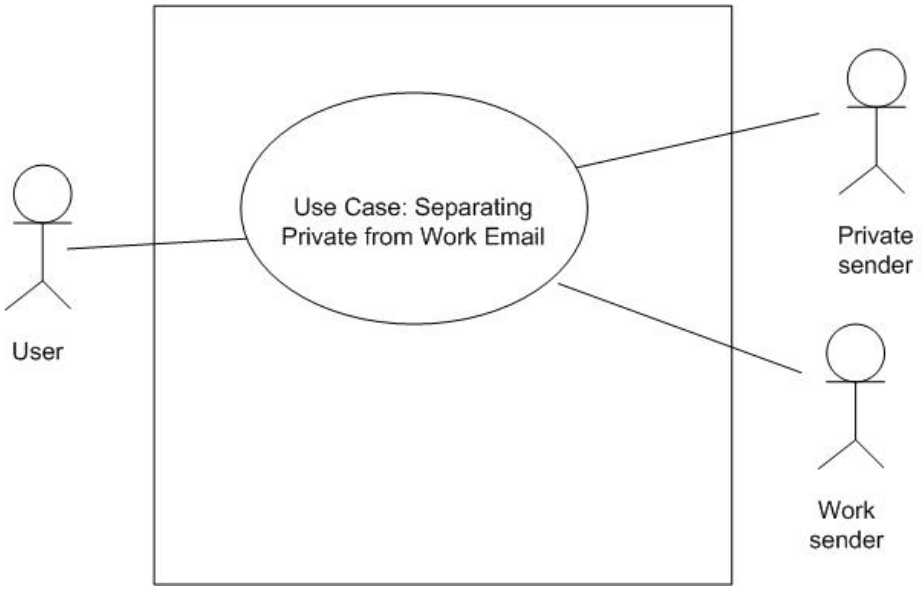

Figure 4.8: Separating Private from Work Email

The following descendant use case serves as an example for all the existing email clients analyzed. The same steps apply to Microsoft Outlook, Lotus Notes, Mozilla Thunderbird, Eudora, KMail, Balsa, GroupWise, Novell Evolution, The Bat!, and Pegasus.

1. **Descendant Use Case: Separating Private from Work Email with AppleMail**

Element	**Description**
Use Case Steps	The user: **2.** Has to manually separate the private from the work messages

B. Out of Office Reply

All descendant use cases in this section B inherit from the deferred use case: Out of Office Reply.
This deferred use case describes the common behaviour of a user to manage out of office replies. This common behaviour includes steps that the user takes that are independent of the particular email client.

Deferred Use Case: Out of Office Reply

Element	**Description**
Summary	This use case describes the steps a user takes upon receiving an out-of-office reply (OOOR).
Frequency of Use	It is on a daily to weekly basis, whenever the user receives a new out-of-office message.
Actors	The *receiver* that receives the out-of-office reply, the *sender* is out of office and has an automatic reply configured.
Preconditions	The email client must be running. The receiver gets an OOOR message from the sender.
Use Case Steps	The receiver: **1.** Receives an OOOR message from the sender. **2.** The receiver stores the OOOR message with the name of the sender and return date.
Exceptions	There are no exceptions for the generalization use case.
Postconditions	The receiver knows when the sender will return to the office

Illustrations

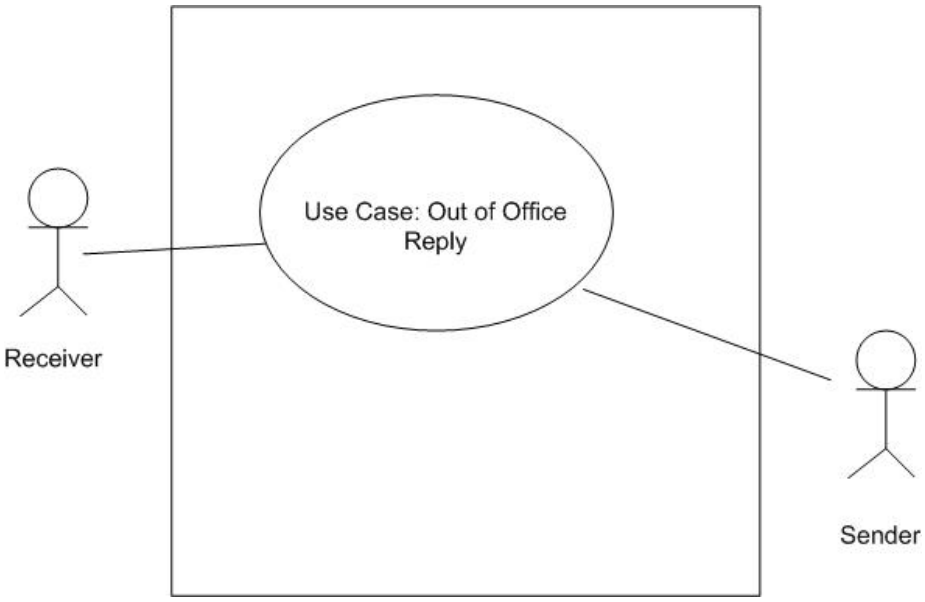

Figure 4.9: Out of Office Reply

The following descendant use case serves as an example for all the existing email clients analyzed. The same steps apply to Microsoft Outlook, Lotus Notes, Mozilla Thunderbird, Eudora, KMail, Balsa, GroupWise, Novell Evolution, The Bat!, and Pegasus.

1. **Descendant Use Case: Out of Office Reply with AppleMail**

Element	**Description**
Use Case Steps	The receiver: **3.** The user has no other possibility to keep track about who is away until when,than to keep notes on paper or in the calendar.

C. Archive Messages

All descendant use cases in this section C inherit from the deferred use case: Archive Messages.
This deferred use case describes the common behaviour of a user to archive email messages. This common behaviour includes steps that the user takes that are independent of the particular email client.

Deferred Use Case: Archive Messages

Element	**Description**
Summary	This use case describes the steps a user takes upon receiving an email to be archived. The user estimates that he will not need the message in the near future but wants to store it for reference.
Frequency of Use	It is on a daily to weekly basis, whenever the user receives such a message.
Actors	The *receiver* that receives the message.
Preconditions	The email client must be running. The receiver gets a message to be archived.
Use Case Steps	The receiver: **1.** Receives a message to be archived. **2.** The user stores the message at an archive location.
Exceptions	There are no exceptions for the generalization use case.
Postconditions	The receiver has stored the message where it is out of sight.

1. **Descendant Use Case: Archive Messages Microsoft Outlook**

Element	**Description**
Use Case Steps	The receiver: **2.** Selects the File menu. **3.** Chooses Archive from the menu. **4.** Selects the item to archive. **5.** Selects the storage where to archive.

2. **Descendant Use Case: Archive Messages Lotus Notes**

Element	**Description**
Use Case Steps	The receiver: **2.** Selects the message to be archived **3.** Selects Actions **4.** Selects Archive **5.** Selects Archive selected documents **6.** Chooses the archive location from the dialog **7.** Chooses ok

3. **Descendant Use Case: Archive Messages with Pegasus**

Element	**Description**
Use Case Steps	The receiver: **2.** Selects the message to be archived **3.** Selects the Folder menu **4.** Selects save messages to disk

4.2.8 Assessment: Attention Management

Attention Management is concerned with how a user can focus on the items in his mailbox that are important. That is, so the important messages receive the attention of the user, while the not so important items are concealed. The separation private - work related email is not offered by the email clients we analysed. The idea is to separate messages into several inboxes, such as private/work related - the concept can also be applied to other cases. The separation could happen based on rules or by categorizing the contacts in private/work contacts. The view of the mailbox could then be customized to show whichever separate inbox is relevant at the moment. Out of office replies are also not managed in the email clients reviewed. The user is not supported by the email client in managing the out of office replies. This would be a useful functionality, especially in large companies where such out of office replies are frequent. Often, the email client is configured that is only sends one out of office reply per contact, so that a sender to the absent receiver is notified only once. The archiving functionality is supported by some of the email clients, but the user needs to do much of the process "by hand". Archiving messages that are not needed at the moment is a core attention management functionality: it conceals the (currently) unimportant messages and therefore helps the user focus on what is important.

Chapter 5

Use Scenarios

The use scenarios describe the situation of how the interaction with the email client solution should be. It builds upon the problems identified in the use cases but there is no one-to-one correspondence between use cases and use scenarios. The use scenario will provide the basis for the implementation of the email client.

5.1 Ideas and Principles

- Focus on core concepts: instead of aiming to have as many features as possible, the email client should focus on the core problems of the users. These core areas are identified by the use cases and are: task management, visualization, attention management and search. The functionalities of the email client should be grouped into these categories and solve a part of one of the problems mentioned. Any other nice-to-have but not essential functionalities should be omitted for the sake of simplicity and compactness of the email client solution.
- The point is debatable: on one hand, an email client would benefit from concentrating on email and its core functions: sending, composing and receiving email messages. In my experience however, email clients are used for much more than those core functions. Task management is especially important in the working environment, where tasks are often explicitly or implicitly included in email messages.
- The use scenarios could also be based on the information gathered in the survey - particularly the section about what features are seen as helpful. All features but one are seen as helpful by the majority of the users that answered the survey.
- The implementation will focus on one of the areas described by the use scenarios.
- The use scenarios will contain technical aspects, but also aspects about handling the email clients (i.e. focusing on the user).

5.2 Use Scenario: Task Management

This section contains the use scenario focusing on how to help the user keep on top of tasks.

Problems to be solved

- It has to become easier for a user to create a task from a message:
  The user should have the option (on the display of the message) to generate a task. With the task window opening next to the message, it is easy and quick for the user to copy/paste the relevant information.
- It needs to be possible for users to create subtasks from tasks.
- It would be nice to have a certain form of messages to inform someone that a particular task has been completed (i.e. to the user that assigned the task). One of the problems stated in the survey was the lack of follow ups.

1. **Use Scenario: Create Task from Email Received**

Element	**Description**
Summary	The use scenario describes the steps the Receiver takes to create a task from an email. This refers to tasks that are not immediately done by the user, but that need to be stored for later reference and reminders.
Frequency of Use	The use scenario is executed daily, on average 5-10 times
Actors	The *Sender* is the user that sends the email containing the task. The *Receiver* is the user that receives this email.
Preconditions	The email client must be running. The *Receiver* must receive an email containing a task.
Use Case Steps	The Receiver: **1.** Opens the email message. **2.** Reads the email and encounters a task. This task is either explicitly requested by the Sender (e.g. "'please send me the report x by deadline y."') or is implicitly implied to the Receiver (e.g. "'I am currently receiving http errors when accessing the intranet"') **3.** Chooses the "create a task" button on the message window. **4.** Copies the details (subject, status, due date, completeness, categories) from the message window to the task window **5.** If the task needs to be broken down into subtasks: goto use scenario Create Subtasks for Tasks. Else: continue. **6.** Stores the task.
Exceptions	No exceptions apply.
Postconditions	The task is stored and the Receiver will be reminded about it before the deadline arises.

2. **Use Scenario: Create Subtasks for Tasks**

Element	**Description**
Summary	The use scenario describes the steps the Receiver takes to create subtasks for a task. This refers to tasks that are not immediately done by the user, but that need to be stored for later reference and reminders.
Frequency of Use	The use scenario is executed on a regular basis, depending on the type of tasks a user usually encounters.
Actors	The *Sender* is the user that sends the email containing the task. The *Receiver* is the user that receives this email.
Preconditions	The email client must be running. The *Receiver* must have a task that needs to be broken down into subtasks. The task menu is opened next to the message the Receiver received from the sender containing the original task. The details of the super task have been filled in by the Receiver.
Use Case Steps	The Receiver: **1.** Chooses the "create subtasks" menu. **2.** Add the details (subject, status, due date, completeness, categories) for each subtask. **3.** Stores the subtasks
Exceptions	No exceptions apply.
Postconditions	The task is stored and the Receiver will be reminded about it before the deadline arises.

3. **Use Scenario: Manage Tasks due Today**

Element	**Description**
Summary	The use scenario describes how the user gains an overview of tasks due today.
Frequency of Use	The use scenario is executed on a daily basis.
Actors	The principle Actor is the user handling the email client.
Preconditions	The email client must be running. The *user* must have one or more tasks stored.
Use Case Steps	The User: **1.** Chooses the view "Show tasks due today". **2.** Has a view of the tasks due today , including the subtasks and sorted by priority.
Exceptions	No exceptions apply.
Postconditions	The user has gained an overview. No tasks in the list are forgotten.

4. **Use Scenario: Complete Tasks assigned by Others**

Element	**Description**
Summary	The use scenario describes how the Receiver completes a task. This entails sending an automatic feedback message to the Sender.
Frequency of Use	The use scenario is executed whenever the Receiver completes a task that was assigned by the Sender.
Actors	The *Sender* is the user that sends the email containing the task. The *Receiver* is the user that receives this email.
Preconditions	The email client must be running. The *Receiver* has completed a task that was previously assigned to him by the Sender.
Use Case Steps	The User: **1.** Selects the relevant task. **2.** Selects the option "send task confirmation" **3.** Selects "complete task".
Exceptions	No exceptions apply.
Postconditions	The Sender receives an automatic email stating that the task he assigned was completed.

5.3 Use Scenario: Visualization

Problems to be solved

- There need to be colours to mark different types of email in the Inbox. For example, messages requiring answer could be marked as "require answer". These messages could then be viewed as a list and/or marked with a particular colour. One of the results of the survey is that most users do not use the search functionality to search their Inbox. Instead, they browse or sort the Inbox. A colour scheme would make scanning the Inbox much easier for the user.
- The idea described above could also be an approach to systematically handle the Inbox. Instead of filing every single message to the appropriate folder upon receiving it, categorize the messages and file them (e.g. once a week) at regular intervals.

1. **Use Scenario: Categorize Messages**

Element	**Description**
Summary	The use scenario describes how the user categorizes the messages.
Frequency of Use	The use scenario is executed on a daily basis.
Actors	The principle Actor is the user handling the email client.
Preconditions	The email client must be running.
Use Case Steps	The User: **1.** Receives a messages. **2.** Opens the message. **3.** Chooses the button "categorize messages". **4.** Chooses a category: requires answer, file for reference, requires task, private email. **5.** Stores the message.
Exceptions	No exceptions apply.
Postconditions	In the Inbox, the user sees the categories visualized as colors. For instance, "requires answer" equals red etc.

5.3.1 Use Scenario: Search

Problems to be solved

- The results of the survey show that the search functionality is barely used. As stated in the analysis of the survey, the reason may be that the users often do not precisely know the subject or the sender. I think the focus should be on arranging/displaying the messages in a helpful way.

1. **Use Scenario: Separate Messages in Inbox by Date**

Element	**Description**
Summary	The use scenario describes how the user separates the messages in the Inbox by date.
Frequency of Use	The use scenario is executed on a daily basis.
Actors	The principle Actor is the user handling the email client.
Preconditions	The email client must be running. There must be one or more messages in the Inbox.
Use Case Steps	The User: **1.** Chooses the "Sort by Date" icon.
Exceptions	No exceptions apply.
Postconditions	In the Inbox, the user sees the messages separated by date. The most current messages are shown at the top of the list.

2. **Use Scenario: Separate Messages in Inbox by Sender**

Element	**Description**
Summary	The use scenario describes how the user separates the messages in the Inbox by sender.
Frequency of Use	The use scenario is executed on a daily basis.
Actors	The principle Actor is the user handling the email client.
Preconditions	The email client must be running. There must be one or more messages in the Inbox.
Use Case Steps	The User: **1.** Chooses the "Sort by Sender" icon.
Exceptions	No exceptions apply.
Postconditions	In the Inbox, the user sees the messages grouped by the sender. The senders are sorted (by default) alphabetically.

3. **Use Scenario: Separate Messages in Inbox by Category**

Element	**Description**
Summary	The use scenario describes how the user separates the messages in the Inbox by category.
Frequency of Use	The use scenario is executed on a daily basis.
Actors	The principle Actor is the user handling the email client.
Preconditions	The email client must be running. There must be one or more messages in the Inbox. The use scenario "categorize messages" was executed previously.
Use Case Steps	The User: **1.** Chooses the "Sort by Category" icon.
Exceptions	No exceptions apply.
Postconditions	In the Inbox, the user sees the messages grouped by category. The categories are sorted (by default) alphabetically.

5.4 Use Scenario: Attention Management

Problems to be solved

- The problem with out of office replies needs to be solved. The user should have the possibility to create an out of office entry. This would be a list of: who is absent, from when until when. The email client then checks who is absent (i.e. which user on the "to" list of an email is absent) and then alerts the user, e.g. by marking this contact red.
- Many users stated in the survey that they decide whether to read a message or not based on the subject. It would therefore be useful to design a sort naming convention. It could be based on prefixes, as with the existing RE: and FW: prefixes. There could be one for general information, meetings, as a reference, including a task, etc.

1. **Use Scenario: Receive Out of Office Reply**

Element	**Description**
Summary	The use scenario describes the steps the Receiver takes upon receiving an out of office reply.
Frequency of Use	The use scenario is executed on a regular basis, whenever the Sender of the out of office reply is out of office.
Actors	The *Sender* is the user that sends out of office reply. The *Receiver* is the user that receives this email.
Preconditions	The email client must be running. The *Receiver* have received an out of office reply from the sender.
Use Case Steps	The Receiver: **1.** Opens the out of office reply message. **2.** Chooses the out of office icon on the message display **3.** Adds the details (Name, email address, absent from, absent until) in the out of office menu. **4.** Stores the out of office entry
Exceptions	No exceptions apply.
Postconditions	No postconditions apply

2. **Use Scenario: Compose Message to Contact Out of Office**

Element	**Description**
Summary	The use scenario describes the steps the Sender takes when composing a message to a contact who is out of office.
Frequency of Use	The use scenario is executed on a regular basis, whenever the Sender composes a message to a contact who is out of office.
Actors	The *Sender* is the user that is composing the message. The *Out of Office Contact* is the user that is out of office.
Preconditions	The email client must be running. The *Sender* must have executed the use scenario: Receive Out of Office Reply.
Use Case Steps	The Sender: **1.**. Adds the Out of Office Contact to the "to" list of the message. **2.** Is alerted by the email client that the Out of Office Contact is currently absent **3.** The Sender decides whether to send the message anyway or not. **-** if yes: Sender sends the message. **-** if no: the Sender deletes the Out of Office Contact from the "to" list.
Exceptions	No exceptions apply.
Postconditions	No postconditions apply

3. **Use Scenario: Send Task to Other User**

Element	**Description**
Summary	The use scenario describes how the user separates the messages in the Inbox by category.
Frequency of Use	The use scenario is executed on a daily basis.
Actors	The *Sender* is the user that sends the email containing the task. The *Receiver* is the user that receives this email.
Preconditions	The email client must be running. The Sender has composed a message to the receiver and the message is currently displayed.
Use Case Steps	The User: **1.** Chooses the "Task" icon from the categories available. **2.** The email client prefixes a TASK: to the subject.
Exceptions	No exceptions apply.
Postconditions	No postconditions apply

4. **Use Scenario: Send General Information to Other User**

Element	**Description**
Summary	The use scenario describes how the user separates the messages in the Inbox by category.
Frequency of Use	The use scenario is executed on a daily basis.
Actors	The *Sender* is the user that sends the email containing the general information. The *Receiver* is the user that receives this email.
Preconditions	The email client must be running. The Sender has composed a message to the receiver and the message is currently displayed.
Use Case Steps	The User: **1.** Chooses the "General Information" icon from the categories available. **2.** The email client prefixes a INFO: to the subject.
Exceptions	No exceptions apply.
Postconditions	No postconditions apply

5. **Use Scenario: Send Request for Reply to Other User**

Element	**Description**
Summary	The use scenario describes how the user separates the messages in the Inbox by category.
Frequency of Use	The use scenario is executed on a daily basis.
Actors	The *Sender* is the user that sends the email containing the request for reply. The *Receiver* is the user that receives this email.
Preconditions	The email client must be running. The Sender has composed a message to the receiver and the message is currently displayed.
Use Case Steps	The User: **1.** Chooses the "Request for Reply" icon from the categories available. **2.** The email client prefixes a REQUEST: to the subject.
Exceptions	No exceptions apply.
Postconditions	No postconditions apply

Chapter 6

TMail: Task Email Client

The previous chapters of this thesis have described the conceptual background of what to design in an email client. We propose a simple email client "TMail", that is centered around task management. More functionalities, as described in the use scenarios of this thesis, can be added at a later stage.

TMail is implemented in Eiffel and reuses the EiffelVision library clusters. It is based on the framework developed by Andrea Rezzonico [24]. The goal was to reuse the framework in an innovative way. Instead of implementing the standard email client functionalities that are offered by the commercial solutions, TMail focuses on the results of the use scenarios of this thesis.

6.1 Principles

Our design is based on the following principles:

In TMail, there is no concept of folders. We believe that the foldering structure is not very useful in organising email. It is time consuming to (a) find the appropriate folder and (b) to locate email messages in the folders. As stated in the survey results, this often results in users having one large Inbox repository. Instead, our approach to organise email messages consists of the following strategies:

- **Fast and efficient search**: The search mechanism implemented in the framework [24] allows the user to search over the entire mailbox contents. It also allows the users to query as they type, i.e. the search is constantly refined they enter more information. With the powerful search mechanism, where exactly the message is located is transparent to the user. We therefore suggest to omit the folder structure.
- **Message categorization**: When users read an email, a implicit categorization usually takes place. For instance: does the message imply a task? Am I going to reply now? Am I going to reply later? Do I store the message for further reference? TMail allows the users to categorize messages as "task" or "reply later". In both cases, they are added to a separate

view (the task list and the to reply list). This is a strategy to help the users focus on the important information. Users have the possibility to choose the categorization within the message window. Message contents, subject, and due date are extracted from the original email message and added to the task window. This saves the user from manually copying this information.

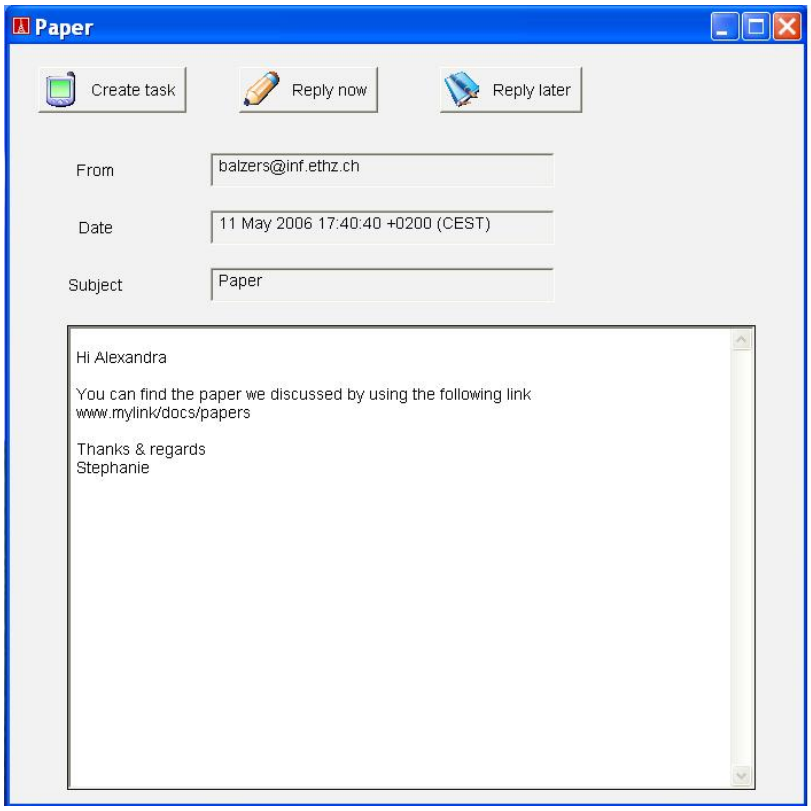

Figure 6.1: TMail message window

- **Inbox**: The Inbox serves as a repository for incoming email that has not been categorized or removed yet. It is the section of the email client that allows unstructured information. As a user can easily categorize a message upon reading it, it also means that a user is supported in keeping the Inbox small.

Chapter 7

Conclusion and Future Work

In this thesis, we have provided some conceptual background material to design a user interface for an email client. First, we provided an overview of current work in the area of email clients. In particular, we introduced the projects Remail [25], TaskMaster [4] and Inner Circle [28].

Next, we analysed existing commercial and open source email client solutions and compared features and functionalities. Based on our selection criteria, we selected eleven email clients for further analysis. We also conducted a survey about email clients, to understand what problems users face. This step was particularly important to provide information about the non-technical aspects of email clients (e.g. usability).

The information collected in the previous steps was then used to design a set of use cases concerned with the four functionality areas: Task Management, Visualization, Search and Attention Management. The goal of these use cases was to point out situations that are problematic when applied to the existing email clients. First, we collected the common behaviour into deferred use cases. For each of the email clients we analysed, we introduced a descendant use case that inherited from the deferred use case.

We then described a set of use scenarios to describe, what elements of a functionality should be implemented. Again, we incorporated information from the previous steps, in particular from the use cases and the user survey.

Finally, we proposed TMail, a simple email client implementation built upon the existing email client framework [24]. As future work, TMail could be extended to include the functionalities described in the use scenarios:

- **Extend task management**: The task management of TMail could be extended to include the elements described in the use scenarios. In particular, future work in this area should provide a subtasking mechanism: it should be possible for users to break tasks down into subtasks and manage them accordingly.

- **Extend the concept of categorization**: Currently, there are three categorizations: none, task and reply to. It would be nice if the users could define their own categories. Also, it would be convenient to be able to apply categorization to a collection of messages. For instance, to a collection of email resulting from a search. Categorization could also be combined with Visualization strategies such as color schemes (i.e. different colors for the categories).

- **Include attention management**: The concepts described in the use scenarios: attention management could also be integrated. An out of office reply management system and a naming scheme for message subjects could also be added to TMail.

- **Extend access to the search mechanism**: The search menu could be extended to allow search by fields: subject, sender, category, date etc. These functionalities are provided by the framework.

Chapter 8

Appendix

8.1 Email Client Comparison

Email Client	Visualization	Task Management	Search	Special Features
Apple Mail			Search selected/all mailboxes content, subject or to/from fields	Smart Mailboxes: filled by filters Parental Control: Monitor Kids Email [a] Photo Attachment Resizing [b] iSync: Synchronise multiple Mail accounts
Microsoft Outlook	Arrange Inbox by Conversation	Prioritize/Tag messages Task panel	Content, subject or to/from fields Online dictionaries	Instant Messaging Save common queries
Lotus Notes		To-do list	full-text search	Personal Journal Instant Messaging
Thunderbird			subject/sender fields	Anti-Phishing alert Automatic Updates
Eudora	BossWatch: warning for particular email addresses [c]		Search Mailbox, Folder or Web (with Google)	ScamWatch: Anti-Phishing ContentConcentrator: removes redundant content from view [d] MoodWatch: monitors offensive content
KMail	Basic threading visualization Color coding for content		content, subject or to/from fields	
Balsa			content, subject or to/from fields	
Groupwise	Color coded message categories	To-do list, checklist	content, subject or to/from fields	Instant Messaging Palm and pager support
Novell Evolution		To-do list Customized reminders	content, subject or to/from fields	Palm device support
The Bat!		Task scheduler	content, subject or to/from fields	Multilingual interface Smart Sorting Office: autorespond with templates Built-in image viewer
Pegasus	Color schemes		date, size, sender, subject, colour or thread	Circulation messages: send in successive order [e]

Table 8.1: Email Client Comparison

[a] Parents can define the email addresses that their children can send emails to or receive emails from, respectively.
[b] Allows users to choose between small, medium and large picture attachments. It also alerts if the attachment is too large to send.
[c] This addresses the problem of inadvertently replying to all. A user can define email addresses (such as the boss') which are then specially marked
[d] It hides content in a sequence of replies to make the thread more readable
[e] This is used to send a message to a list of people in order, e.g. for each member to add a comment

Bibliography

[1] Scott W. Ambler. *The object primer: the application developer's guide to object-orientation*. SIGS Publications, Inc., New York, NY, USA, 1996.

[2] Balsa. http://balsa.gnome.org/, 2006.

[3] The Bat! http://www.ritlabs.com/en/products/thebat/, 2006.

[4] Victoria Bellotti, Nicolas Ducheneaut, Mark Howard, and Ian Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 345–352, New York, NY, USA, 2003. ACM Press.

[5] Laura A. Dabbish, Robert E. Kraut, Susan R. Fussell, and Sara B. Kiesler. Understanding email use: predicting action on a message. In *CHI*, pages 691–700, 2005.

[6] Remail: Reinventing Email. http://www.research.ibm.com/remail/index.html.

[7] Eudora. http://www.eudora.com/, 2006.

[8] Novell Evolution. http://www.novell.com/products/desktop/features/evolution.html, 2006.

[9] Groupwise. http://www.novell.com/products/groupwise/, 2006.

[10] Daniel Gruen, Steven L. Rohall, Suzanne O. Minassian, Bernard Kerr, Paul Moody, Bob Stachel, Martin Wattenberg, and Eric Wilcox. Lessons from the remail prototypes. In *CSCW*, pages 152–161, 2004.

[11] Thomas B. Hodel, Roger Hacmac, and Klaus R. Dittrich. Using text editing creation time meta data for document management. In *CAiSE*, pages 105–118, 2005.

[12] Ivar Jacobson. The use-case construct in object-oriented software engineering. pages 309–336, 1995.

[13] Ivar Jacobson. Use cases - yesterday, today, and tomorrow. *Software and System Modeling*, 3(3):210–220, 2004.

[14] KMail. http://kmail.kde.org/, 2006.

[15] Georg Kösters, Hans-Werner Six, and Mario Winter. Coupling use cases and class models as a means for validation and verification of requirements specifications. *Requir. Eng.*, 6(1):3–17, 2001.

[16] Luc Lamontagne and Guy Lapalme. Textual reuse for email response. In *ECCBR*, pages 242–256, 2004.

[17] David D. Lewis and K. A. Knowles. Threading electronic mail - a preliminary study. *Inf. Process. Manage.*, 33(2):209–217, 1997.

[18] Apple Mail. http://www.apple.com/macosx/features/mail/, 2006.

[19] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, 2nd edition, 1997.

[20] Gail C. Murphy, Mik Kersten, Martin P. Robillard, and Davor Cubranic. The emergent structure of development tasks. In *ECOOP*, pages 33–48, 2005.

[21] Lotus Notes. http://www-128.ibm.com/developerworks/lotus, 2006.

[22] Microsoft Outlook. http://office.microsoft.com/en-us/fx010857931033.aspx, 2006.

[23] Pegasus. http://www.pmail.com/, 2006.

[24] Andrea Rezzonico. Master thesis: Designing an innovative email client, 2005.

[25] Steven L. Rohall, Dan Gruen, Paul Moody, Martin Wattenberg, Mia Stern, Bernard Kerr, Bob Stachel, Kushal Dave, Robert Armes, and Eric Wilcox. Remail: a reinvented email prototype. In *CHI Extended Abstracts*, pages 791–792, 2004.

[26] Mia K. Stern. Dates and times in email messages. In *Intelligent User Interfaces*, pages 328–330, 2004.

[27] Mozilla Thunderbird. http://www.mozilla.com/thunderbird/, 2006.

[28] Andrzej Turski, Debbie Warnack, Lili Cheng, Shelly Farnham, and Susan Yee. Inner circle: people centered email client. In *CHI Extended Abstracts*, pages 1845–1848, 2005.

[29] Gina Danielle Venolia and Carman Neustaedter. Understanding sequence and reply relationships within email conversations: a mixed-model visualization. In *CHI*, pages 361–368, 2003.

[30] Hironori Washizaki and Yoshiaki Fukazawa. A model-view separation architecture for gui application components. In *ITCC (2)*, pages 359–364, 2005.

[31] Steve Whittaker and Candace L. Sidner. Email overload: Exploring personal information management of email. In *CHI*, pages 276–283, 1996.

[32] Wikipedia. List of email clients, 2006.