

BROWN UNIVERSITY
Department of Computer Science
Master's Thesis
CS-92-M18

“NeXTPIEMail

A Graphical, Personalized Environment Mail Tool”

by
Stuart Hamlyn

NeXTPIEMail

A Graphical, Personalized Environment Mail Tool.

Stuart Hamlyn
Department of Computer Science
Brown University

Submitted in partial fulfillment of the requirements for the
Degree of Master of Science in the Department of Computer Science
at Brown University

June 14, 1992

This research project by Stuart Hamlyn is accepted in its present form
by the Department of Computer Science at Brown University
in partial fulfillment of the requirements for the Degree of Master of Science

Peter Wegner

Professor Peter Wegner

Advisor

July 15 1992

Date

Introduction	6
Abstract	6
Traditional Pitfalls of Electronic Mail	6
NeXTPIEMAIL's Solutions	7
Automatic Message Filing / Task Support	7
Query Processing	7
Filtering	7
Conversation Tracking	8
Interactive Rolodex Features	8

The Electronic Mail Handling Model	9
Standard UNIX E-mail funtions.....	9
Second Generation E-mail Systems	9
PIEmail Systems	9

NeXTPIEMAIL Requirements	11
Overview	11
Message Browsing	11
Browser View of Mailboxes	11
Viewing.....	12
Rolodex Support	12
Message Management	12
Categorizations	12
Conversations	12
Sets	13
Task Support	13
Automation	13
Message Composition	13
Editing.....	13
Intelligent Forms	13
Message Retrieval	14
Querying	14
Address Lookup	14

NeXTPIEMAIL Design	15
Overview	15
General Approach	15
Algorithms and Data Structures	16
NeXTPIEMAIL [Application] Object	18

MBoxWindow Class	20
ReadWindow Class	22
SendWindow Class	24
AddressBook Class	25

NeXTPIEMAIL Implementation	26
The NeXT Development Environment	26
Objective C - Class Organization and Code Sizes	26
Mark Crispin's C-client and MailManager	28

Future Enhancements	29
Performance	29
Nested Folders	29
Querying Across Folders	29
Storing Queries	29
Task and Role-based Enhancements	29

Conclusion	30
-------------------------	-----------

Sample User Scenario	31
Role Definitions	31
Message Composition	31
Message Reply	33
New Role/Task Creation	34
Conversation Tracking	34
Address Book Addition	35
Query Command	35

NeXTPIEMAIL Source Code	36
--------------------------------------	-----------

References	37
-------------------------	-----------

Chapter Summary

Chapter 1: Introduction

The first chapter provides an abstract of what the NeXTPIEMAIL project is all about. It discusses the drawbacks of traditional electronic mail systems and some of the major steps NeXTPIEMAIL has taken to overcome them.

Chapter 2: The Electronic Mail Handling Model

The second chapter presents NeXTPIEMAIL in context with electronic mail handling models. It considers the second generation E-mail systems currently in place and explains why PEmail is a step toward the third generation.

Chapter 3: NeXTPIEMAIL Requirements

The requirements document offers a point by point breakdown of what functionality the system requires to become a "PEmail" entity. It breaks electronic mail functionality into browsing, management, composition, and retrieval functions.

Chapter 4: NeXTPIEMAIL Design.

The design chapter goes through some rudimentary design issues, describing the platform and system on which NeXTPIEMAIL was designed. The chapter is mostly concerned with a class-by-class breakdown of the functionality of NeXTPIEMAIL.

Chapter 5: NeXTPIEMAIL Implementation

The implementation is concerned with how the NeXT computer and Objective C contributed to NeXTPIEMAIL's development. The role of Mark Crispin's C-Client is explained fully, and the class organization and code lengths are given for reference.

Chapter 6: Future Enhancements

The future enhancements chapter focuses on what key design issues one should concentrate on if one is to expand the NeXTPIEMAIL system.

Chapter 7: Conclusion

The conclusion offers some parting remarks on the nature of personalization and the degree to which NeXTPIEMAIL succeeded in its goals.

Appendix A: Example Scenario

Presents a sample interactive session with NeXTPIEMAIL.

Appendix B: Source Code

1. Introduction

1.1 Abstract

Processing several hundred, or more, messages a day opens up a Pandora's box of not only electronic mail issues but user interface issues as well. NeXTPIEMAIL is a research project that examines this problem of information overload and investigates to what degree personalizing the virtual desktop via an intelligent, graphical, user interface can solve these problems. It is NOT the goal of NeXTPIEMAIL to develop a more sophisticated mail server or to change the electronic mail messaging system internally. Rather, NeXTPIEMAIL seeks to demonstrate the effectiveness of intelligently processing electronic mail locally, on the user's workstation. Intelligent, local mail processing should lead to less information loss, less duplicate information flooding bandwidth, and open up new vistas for electronic mail use.

NeXTPIEMAIL is an entirely object-oriented program developed in Objective C. It's implementation demonstrates the synergies of combining a graphical user interface with an object-oriented language. The design closely parallel's the user - interface itself with major classes defined for major interface entities. This is in sharp contrast to most applications written in non object-oriented systems where the program often parallel's the data structures holding the user's information.

While NeXTPIEMAIL addresses only information overload in the realm of electronic mail, it is the hope of the author that this work will serve as a model for structuring other workstation environments. The same user-interface and object-oriented design principles could ideally be carried into the document management aspects of an operating system, for example.

1.2 Traditional Pitfalls of Electronic Mail

Electronic mail systems today encourage users to treat mail as "disposable information." That is, users typically use electronic mail (E-mail) to send trivial logistical-type information. For example, who is meeting who when, etc... Although E-mail is a fine method of resolving these concerns, it has the potential to be much more than a glorified answering machine system. To enlighten today's E-mail users and to expand the role of electronic mail, we need tools that do more than satisfy a bare minimum of requirements such as message delivery, receiving, and forwarding. We need to develop tools that not only handle more complicated multi-media messages, but also allow the user to cope with a high-volume of messages. A flat file mail system, where mail from any sender is deposited in a contiguous pile, unfiltered and uncategorized - will simply not suffice if one is receiving several hundred messages per day.

1.3 NeXTPIEMAIL's Solutions

NeXTPIEMAIL offers essential attributes vital to high-volume / personalized electronic mail handling. By combining a graphical user interface with an intelligent environment, mail throughput can be increased and inefficient mail management avoided.

The graphical user interface chosen, NeXTSTEP, has extensive support for multiple layered menus and windows, allowing the user to browse the available commands at leisure, to observe and operate on more than one entity at a time, and to receive extensive on-line documentation.

The "intelligent environment" encompasses the automation and personalization features consistent of the PImail system. These features include:

- Automatic Message Filing / Task Support
- Query Processing
- Filtering
- Conversation Tracking
- Interactive Rolodex Features

A brief description of each of these main features is provided below. A more extensive discussion of NeXTPIEMAIL's functionality can be found under "PImail Systems" on page 9 and in "NeXTPIEMAIL Design" on page 15.

1.3.1 Automatic Message Filing / Task Support

Automatic message filing allows the user to identify certain mailboxes or "folders" that mail is deposited into automatically. For example, if the user is working on a project code-named "bluedog"; it is a simple matter to differentiate this mail from the rest he receives. He merely creates a "role" ¹ called "bluedog" and any mail with "bluedog" in the subject line is automatically filed in this folder. This allows the user to keep his mail categorized minimizing information loss, duplicate requests, and multiple filing (by saving mail into textfiles) on the user's workstation.

1.3.2 Query Processing

Query processing is a complex pattern matching algorithm implemented via a simple one form quick-search function. The user is allowed to search for any word or phrase in any part of a message (i.e. subject, date, body, etc...) The user need only fill out one form indicating what he is searching for, and the system automatically highlights all the messages in the current mailbox which meet his request.

1.3.3 Filtering

Filtering is the ability of NeXTPIEMAIL to filter out messages that are not of interest when examining the message base. For example, if the user is only interested in mes-

1. please see Scenerio section of the Appendix for more detailed examples of roles, views, and query processing.

sages pertaining to his tennis hobby, he can enter perform a query with "tennis" as his keyword, and then click <filter> to reduce his active message set to only those messages that meet his request.

1.3.4 Conversation Tracking

Conversation Tracking is a powerful feature allowing the user to locate all messages in an ongoing email discussion by clicking on *any* message in the set. NeXTPIEMAIL automatically scans the chain of reply-to's for messages that reference the message the user clicks on and reports all related messages. Combined with the filtering feature, this feature offers a powerful means of tracking ideas and project development.

1.3.5 Interactive Rolodex Features

Interactive rolodex features allow the user to keep libraries of E-mail correspondents with biographical information. The rolodex is bidirectional - that is - a name in the rolodex can be clicked on and automatically inserted into a mail message being composed, or a name in a mail message can be clicked on and the corresponding biographical information and name highlighted in the rolodex.

2. The Electronic Mail Handling Model

2.1 Standard UNIX E-mail functions

The traditional "model" for electronic mail-management systems provides a flat-file interface that allows the user to send, receive, print, reply-to, and forward electronic mail. The system has a small dose of not-so-elegant text-management features such as allowing the user to insert a file in a mail message and to save a particular message to a file. The standard E-mail software also allows one to "preview" the contents of a mailbox by examining their corresponding message headers.

This traditional model, although a bare bones, textual system, does set certain precedents. For example, the summary-level view (where the message headers containing the date, subject, and message sender of each message is displayed), has become the preferred method of examining the contents of a mailbox. Furthermore the manner in which replying is done has been duplicated. This is accomplished by appending an "re:" to the start of the message subject in the reply note.

This first-generation mail system was designed to be used as a means for various users of the system to report problems to one another and to communicate about the system in general. It offers no multi-media capability, multi-level filing, or any features consistent with using mail for everyday correspondence in the workplace.

2.2 Second Generation E-mail Systems

The first significant enhancement to this traditional mail system was probably the EMACS mail reader, RMAIL [Stallman87]. RMAIL and its derivatives such as PCMAIL [Lambert] offered for the first time the ability to send "attachments" along with the traditional E-mail correspondence. Attachments take the form of spreadsheet files, sound files, graphics files, etc... In short, E-mail could now be used as a means of sharing "work" and not just textual information usually pertaining to the computer system being used.

These systems also allowed the creation of multiple mailboxes, so that one could organize messages, or categorize them. This was the first sign that E-mail was to be used as something more than a disposable information dissemination system. [Mackay88] If one could send important documents, then it was a natural extension to demand the ability to categorize them and preserve them for future reference. In short, such systems ushered in the revelation that mail was not only multimedia in nature, but a vehicle for group, collaborative work. This represents a paradigm shift, not simply a boost in performance or an elegant menu-driven E-mail system.

2.3 PImail Systems

While there is no clear third-generation E-mail system under development, it seems that large returns to scale can be achieved by personalizing the workstation environment so that it is better suited to mass management of information. In his work *PImail Technical*

Report [Weiner92], Bob Weiner discusses these aspects of personalization concerning mail overload. One facet of this personalization is the concept of personal "roles" that the user can define to help him manage his mail. For example a mail user can establish roles such as "jim.student" and "jim.doctor" so that mail addressed specifically to these roles will automatically be filed in a specific folder Jim has created.

NeXTPIEMAIL seeks to take this PIE concept one step further by overlaying a rich, graphical interface above a PIE system. In this manner, complicated functions can be exercised over arbitrary message groups by simply pointing and clicking. Similarly, the user is immediately aware of the functionality of the system through on-line help and a menuing system that is easily browsed. For example, if one wants to forward all messages that concern a "flywheel" project to Dave, the user simply clicks on the query window, types "flywheel" in the text box, clicks ok, then clicks "forward" at which point all highlighted messages (which are the ones concerning the query on flywheel) are sent to an arbitrary recipient (which can either be keyed in directly by the user or selected from the on-line rolodex.)

By personalizing the workspace the functionality of section 1.2 is achieved without altering the fundamental mail messaging system. NeXTPIEMAIL, then, is a graphical extension of a Personalized Mail Environment (PIE), which is itself an enhancement built out of second generation E-mail software. The work represents a commitment to the paradigm of mail as a critical communication agent of tomorrow necessary for collaborative work and not as a disposable messaging facility.

3. NeXTPIEMAIL Requirements²

3.1 Overview

A mail reader's critical tasks are those of browsing, management, composition, and retrieval of messages. Browsing requires a means of viewing messages generally from a number of different perspectives via sorting on different categories and flags. In short, it must permit the user to find saved messages according to known criteria.

Message management should permit the logging of outgoing messages, tagging of messages with attributes, as well as message deletion, and refiling. In a graphical environment, it is also necessary to act on groups of messages as easily as acting on single entities.

Message composition requires easy-to-use cut/paste like editing facilities, and a means of managing message attachments accordingly. The composition should involve filling out a form that is self-explanatory to the user and the attachments should be added by clicking or dragging on the external information that is to be attached (i.e. sound file, graphic, text file, etc...)

Message retrieval necessitates searching over groups of messages. A query feature should be provided which gives the user a simple form-based query request that will automatically locate messages matching regular expressions in any message field. For example, one should be able to find all messages sent by Fred, or all messages with the word "bluedog" in the message text quickly and easily. Retrieval also necessitates an interactive rolodex that can serve as an address finder and biographical information review tool.

The following are more specific requirements expanded from the above paragraphs.

3.2 Message Browsing

Electronic mail tools are often called mail readers, indicating that their primary purpose is to display mail for viewing. NeXTPIEMAIL seeks to provide a flexible, graphical viewing environment that allows the user to scan message boxes and individual messages in an intuitive manner.

3.2.1 Browser View of Mailboxes

Sets of messages (mailboxes) should be displayed in a "browser-like" format where message header information is displayed in a list. The user should be able to scroll through the list easily, and highlight messages on which he would like to perform operations. Double clicking on any message should bring up a window with the detailed informa-

2. The reader may note that these requirements parallel closely the work of [Weiner92], and as such reflect traditional features in a Personalized Information Environment (PIE). NeXTPIEMAIL is, in one sense, a graphical extension of a PIE.

tion about the message and the message text itself. The user should be able to filter out unnecessary information, such as message id # ,etc...

3.2.2 Viewing

The user should be able to order the list of messages in this top-level browser in a variety of ways (for example, sorted by subject, date, or on a specific keyword.) Furthermore, the status of a message should be indicated by the message's header line in the browser. Message status should indicate whether it was read, whether it was answered, deleted, or flagged as important.

3.2.3 Rolodex Support

The browser should automatically highlight the proper page in the rolodex when a name is clicked on. When a message comes in from someone unfamiliar, it should be easy to add his name and mailing address information to a personal address book. The address book should also permit free annotation of this information with additional material.

3.3 Message Management

Managing messages refers to logging, filing, ordering, and otherwise organizing the message base. Organization influences information access, quality, and utility. In the NeXTPIEMAIL domain, having a graphical means of manipulating the message base is critical. Appropriate graphical means of selecting sets of messages to be operated upon, highlighting these messages, and offering feedback should exist.

3.3.1 Categorizations

One should be able to categorize messages, to prioritize them, to reorder them, to produce groups of messages that match particular conditions and to tag messages with standard or personalized labels.

Categorization is critical for high volume message bases. As volume increases, it is an all too common occurrence for important messages to become buried. Prioritization keeps unimportant messages from obscuring high priority ones that are categorized together. By sorting on priority, one can quickly see the messages that must be handled first. Furthermore, one should be able to filter out messages that are irrelevant for a moment. One can then get a clear view of particular messages, e.g. those from your boss.

3.3.2 Conversations

Conversation summaries should be offered so one can see how discussion has developed, i.e. who replied to whom, and what was said. Lack of conversation tracking has long plagued electronic mail tools. It is a chicken and egg problem. Due to weakness of E-mail readers, people often try to process and delete messages quickly to keep from being inundated. Then it becomes impossible to track the conversation because the messages are gone. A richer set of views of message-bases reduces the need to remove messages permanently, permitting more comprehensive tracking of inter-message relations. In keeping with the GUI nature of NeXTSTEP, the conversation should be highlighted in

a graphical manner by simply clicking on any message in the conversation and selecting "track".

3.3.3 Sets

Most operations available for individual messages should be available for sets of messages. When scanning message summaries (browsing), one often finds a number of messages that need to be processed the same way. Applying a procedure to the entire group saves time and avoids the mental distraction of repetitively exercising the same commands over-and-over, one message at a time. A critical aspect of NeXTPIEMAIL is that the group of target messages is easy to select and that it is highlighted in an appropriate manner. A filtering mechanism should also exist to display solely the selected messages and then the entire set of messages - allowing the user to isolate those messages of interest to him.

3.3.4 Task Support

Users should be able to create their own mail addresses in a way that supports project-oriented work, but does not permit "spoofing" of another user. Putting mail handling control into the receiver rather than the sender was researched at Xerox Parc in 1989 [Parc89] and found to be particularly conducive to collaborative work. A user could give a personally generated address to a working group he needs to correspond with to help ensure that all mail from the group, regardless of subject, is processed the same way.

3.3.5 Automation

NeXTPIEMAIL should offer a support structure for automation. If the user wants all messages containing "bluedog" filed in a folder called "special projects" then he should be able to specify this request, so that upon logging on, his message is automatically filed for him. The ability to automatically forward and copy, etc... should also be supported.

3.4 Message Composition

3.4.1 Editing

Editing should be accomplished by filling out electronic "forms" that support cutting and pasting and multi-media insertion / reviewing. NeXTPIEMAIL is built on top of the NeXTSTEP development environment, and therefore offers a rich set of graphics-based text editing tools.

3.4.2 Intelligent Forms

As a form-based, graphical environment, NeXTPIEMAIL should intelligently bring up slightly different forms for different message editing scenarios. For example, if the user selects "reply" NeXTPIEMAIL should bring up a form with the *subject* field and the *in-reply-to* field already completed. Furthermore, if the user is creating a new message, the rolodex should be activated, so the user can easily pick an address to be inserted into the message-edit form.

3.5 Message Retrieval

Retrieval facilities can help motivate efforts to organize message bases. Full-text searching is the preferred retrieval technique in messaging domains because the known information used for searches may often be scanty. For those times where one knows which fields particular information occurs in, structured searches should also be permitted.

3.5.1 Querying

Queries should be able to search large message bases and to display summaries of matching messages. One should be able to traverse these summaries looking for messages of interest. Queries should normally be given by filling out a query form with known information to match against. The form should be very similar to the basic mail composition form. Queries should return a summary listing of matching messages which are highlighted and can be filtered to located desired messages. See "Filtering" on page 7.

It is also desirable to save queries that act as "macros." For example, a query to highlight all messages from your boss, or one to archive last quarter's messages.

3.5.2 Address Lookup

Mail addresses and aliases should be easy to lookup and to insert into outgoing messages. Given a name, one should be able to ask for the person's mail address only or his entire record of information from the address book.

4. NeXTPIEMAIL Design

4.1 Overview

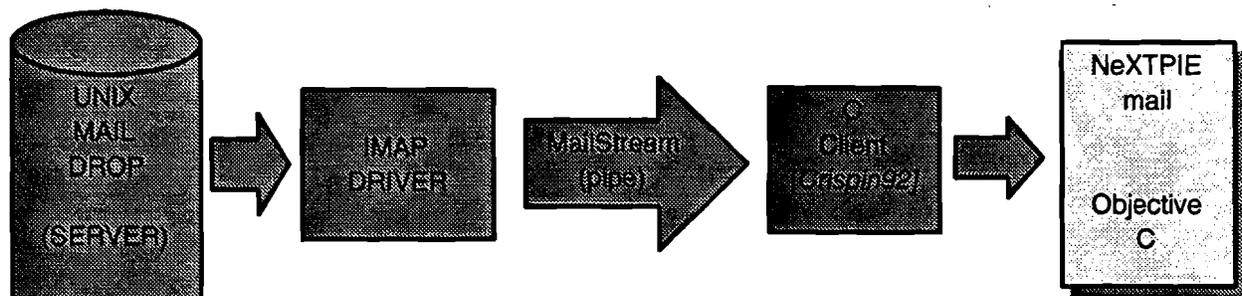
The design portion of this technical report, although presenting all of the major class structures in the NeXTPIEMAIL system, is concerned mostly with the user-interface and personalization aspects of the system. The class structures are thus presented with an eye toward relevant UI elements and structures.

For a broader design perspective regarding PIEs please refer to [Weiner92]. This PIEmail system runs under GNU Emacs [Stallman87] and may be used on terminals and UNIX workstations. (Under the X window version of Emacs called Epoch [KaLoCaLa92], it offers a number of enhanced display features.)

4.2 General Approach

NeXTPIEMAIL is written in Objective C with the aid of *Interface Builder*, the standard NeXT user interface design tool. NeXTPIEMAIL accomplishes most of its mail messaging through a C-Client developed by Mark Crispin at the University of Washington [Crispin92]. The C-Client in turn calls an IMAP driver [Internet Message Access Protocol] via TCP/IP pipes, also developed by Crispin, which contains the low-level calls to the UNIX mail server.

The C-Client was developed as a multi-platform mail library that would facilitate application cross-compiling and development on many common implementations of UNIX and DOS computers. The IMAP driver does the low-level information extracting from the UNIX server, and the C-Client offers higher level support such as envelope fetching, mail queuing, etc...



E-Mail System Architecture under NeXTPIEMAIL

One can see from the above diagram that NeXTPIEMAIL sits at the end of this interacting chain of software beginning with the UNIX mail server. It represents the highest level of abstraction in the chain, the personalized environment with which the user inter-

acts. The C-Client allows it to achieve a high degree of functionality without the overhead usually associated with making calls to the UNIX mail daemon.

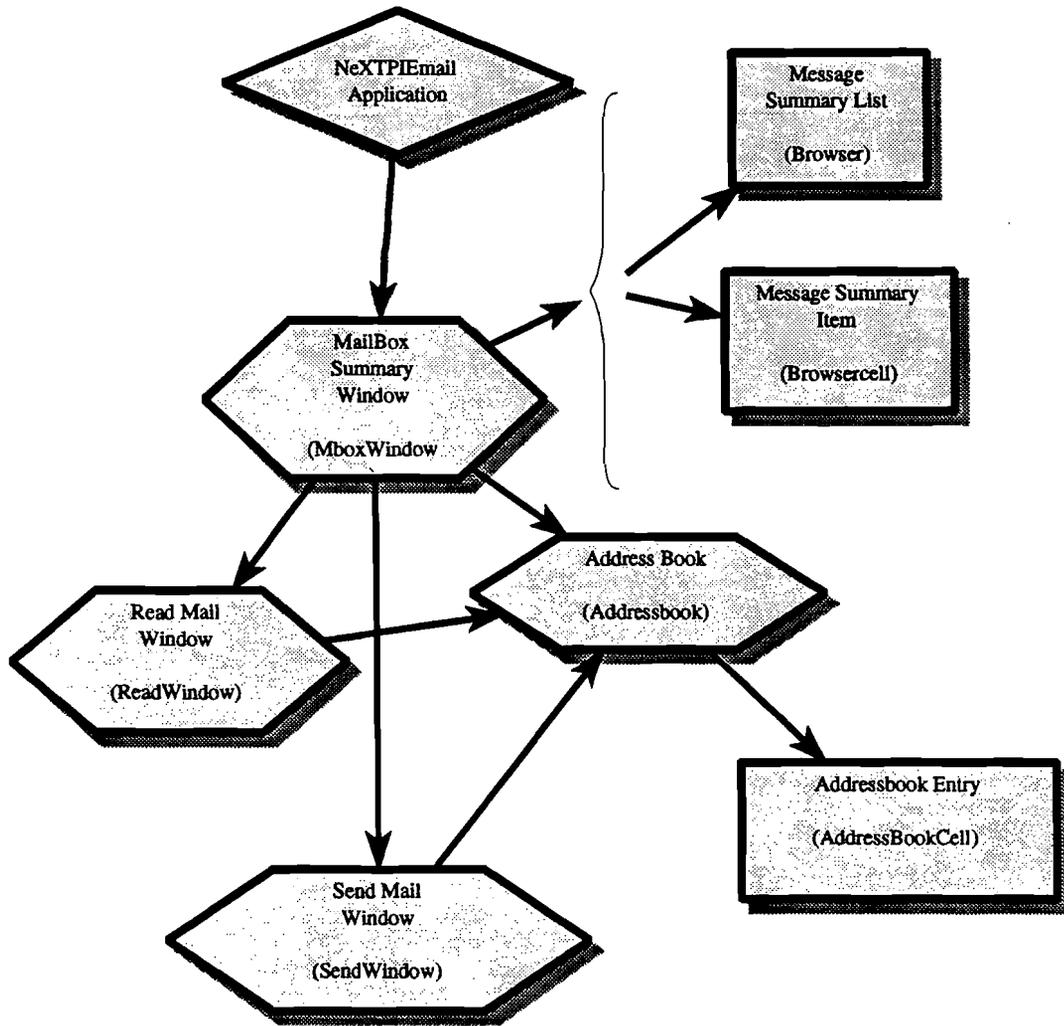
4.3 Algorithms and Data Structures

NeXTPIEmail is in one dimension, an object oriented shell written around the provided C-Client routines. By taking advantage of a high powered graphical user interface and a premier object oriented programming language, NeXTPIEMAIL is able to significantly boost the functionality of the C-Client. There is essentially one major class for each aspect of the NeXTPIEMAIL user-interface.

The important class definitions will be presented and the hierarchy explained below. There are, however several base types that the C-Client relies upon and which are passed among the classes of NeXTPIEmail that may warrant investigation. For this reason it may be desirable to browse the *mail.h* file in the Appendix containing the source code before reading this section. (Note : All C-Client mail attributes refer to RFC822 message formats³)

Figure #2 illustrates the primary objects in the NeXTPIEMAIL system. The NeXTPIEMAIL application class (MailManager) defines various mailboxes (MBoxWindows) which instantiate the other classes as needed. Classes (or views) exist for browsing mail (MBoxWindow), reading mail (ReadWindow), sending mail (SendWindow), and interacting with the address book (AddressBook). In figure #2, the destination object of an arrow is instantiated by the object at the start of the arrow. For example, the (AddressBookCell) is instantiated by the (AddressBook) object.

3. Available as Internet Draft Document RFC-822 from the Internet



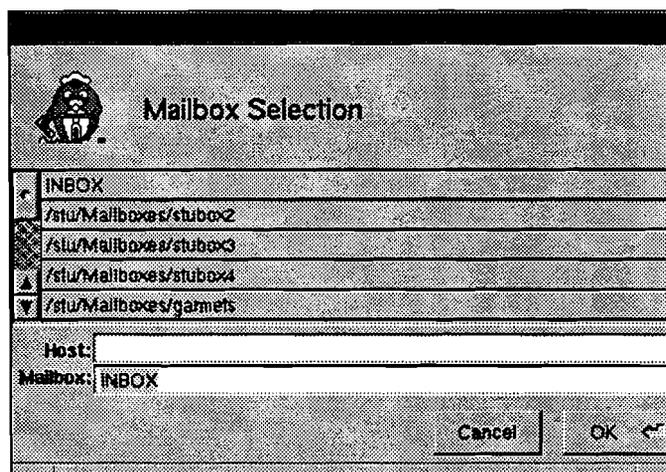
Dependency of Major Objects in NeXTPIEmail
(target object instantiated by source)

Figure 2 - NeXTPIEMAIL Object Dependency

4.3.1 NeXTPIEMAIL [Application] Object

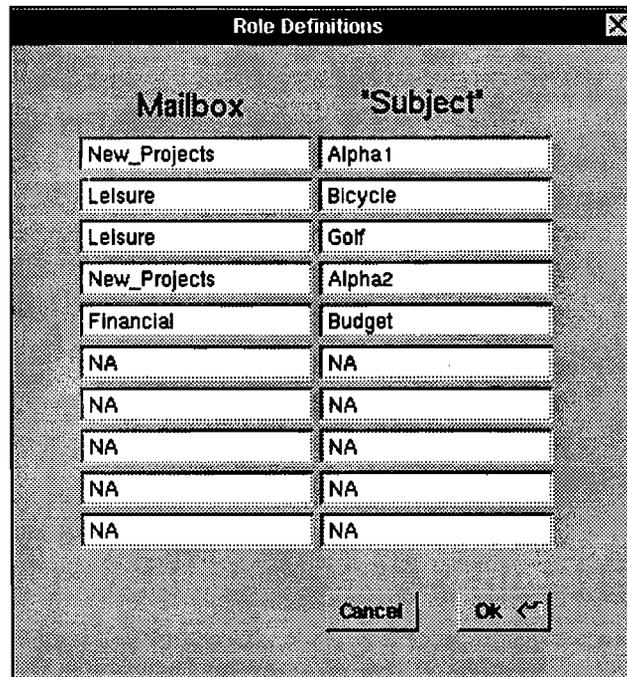
The object NeXTPIEMAIL, a subclass of *Application* is the central object from which all others are instantiated. It manages the pointers to the necessary mail drivers, and the constants declaring the locations of the rest of the user interface objects. All of the low-level (driver level) file operations are performed by NeXTPIEmail. It also offers the following functionality.

- Manages dialog box for logging on to system, this includes looking for any UNIX mail drops (servers and mailboxes) distributed anywhere on the system.
- Includes necessary code for selecting a mailbox from list of available mailboxes and opening a stream to a specific mailbox and instantiating the *MBoxWindow* object (mailbox view containing list of messages)



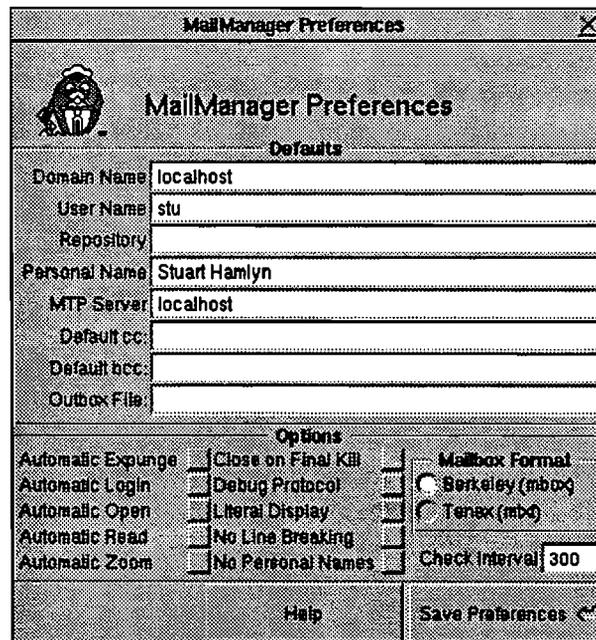
Mailbox Selection Window

- Handles deleting a mailbox, copying a mailbox, and creating a new mailbox.
- The automatic filing feature is accomplished by the method *DoRoleFile* which automatically reads the user's list of automatic filing assignments, or tasks (see "Automatic Message Filing / Task Support" on page 7) Mail is thus distributed from the default UNIX mailbox to the automatic file-messages created by the user each time the system is started up.
- Manages table of automatic filing criteria. This is simply a list of mailboxes and strings to match-on in the subject fields of messages. These can be edited by the user from the NeXTPIEMAIL application object.



Role / Task Creation

- Manages dialog box which contain user preferences, such as default mailbox to search first, what font to display messages in, the name of the local host, what format to store messages in, etc...



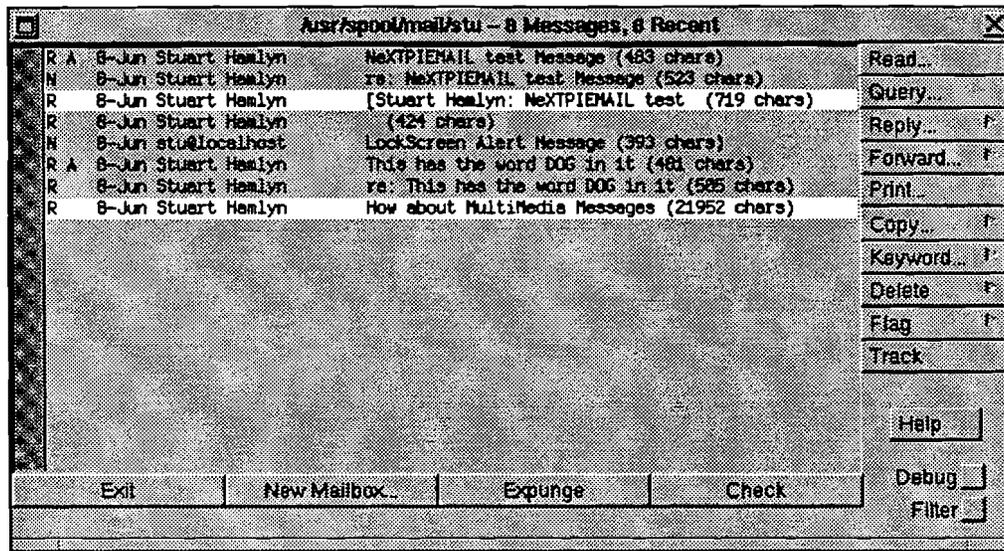
Preferences Window

4.3.2 MBoxWindow Class

The MboxWindow object is instantiated from the application class, MailManager; it is responsible for all the operations performed on items within a single mailbox. It maintains the browser (list) of items within the mailbox and all of the user interface responsibilities associated with this list.

MboxWindow instantiates ReadWindow and SendWindow to allow the user to read and send messages respectively. Similarly, the AddressBook object functions as a delegate for name look-up from the message list. This is in partial fulfillment of the rolodex feature in the PImail system.

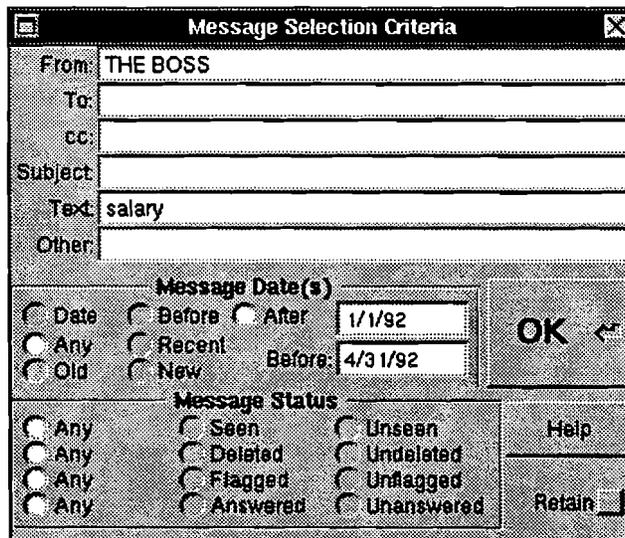
- A `check_for_new_mail` feature is implemented to allow the user to check to see if any new mail has arrived while he has been looking over his mail.
- Delete and Expunge features are provided from the message view. The user can click on a single message to delete it, (the message entry turns to light-grey) and then choose Expunge to erase all of the deleted messages from the mail file. Before expunging, the user can choose undelete to restore the message.
- A message flagging feature is supported to allow the user to flag his messages as urgent/normal.
- Message printing is accomplished through the *ReadWindow* delegate.
- Copy/Move message commands are available to allow the user to personalize his mail library by filing the selected messages in any mailbox. These functions allow one to store multiple copies of messages or to simply empty out the <INBOX>, so that one isn't presented with all the messages each time he logs on the system.
- A conversation tracking feature is activated by clicking on the <track> button, where all references to the current message are automatically selected by the user. The system uses a recursive algorithm to track the chain of reply-to's from a particular message. It is not a "fake" conversation tracker since it does not match on subject lines containing "re: <original title>" as many packages do. The code for this is in [*MBoxWindow contrack*]
- Message sets can be selected by using the standard shift-click method, messages are highlighted (white) when they are selected, and selected messages can be viewed exclusively by clicking on the <filter> button. Clicking the filter button a second time, causes the original view (of all messages) to be restored. This allows one to concentrate on a particular query or set of important messages and to operate on them easily. For example, one can click on a message, select conversation track, and then click on filter, and deal with only messages pertaining to a particular conversation.
- An on-line help facility is provided by clicking on the <help> button. A small window appears discussing how to invoke the various commands in the mail



Example of Message Selection Highlighting

browser view. It explains how to transfer messages between mailboxes, and what known bugs are in the software.

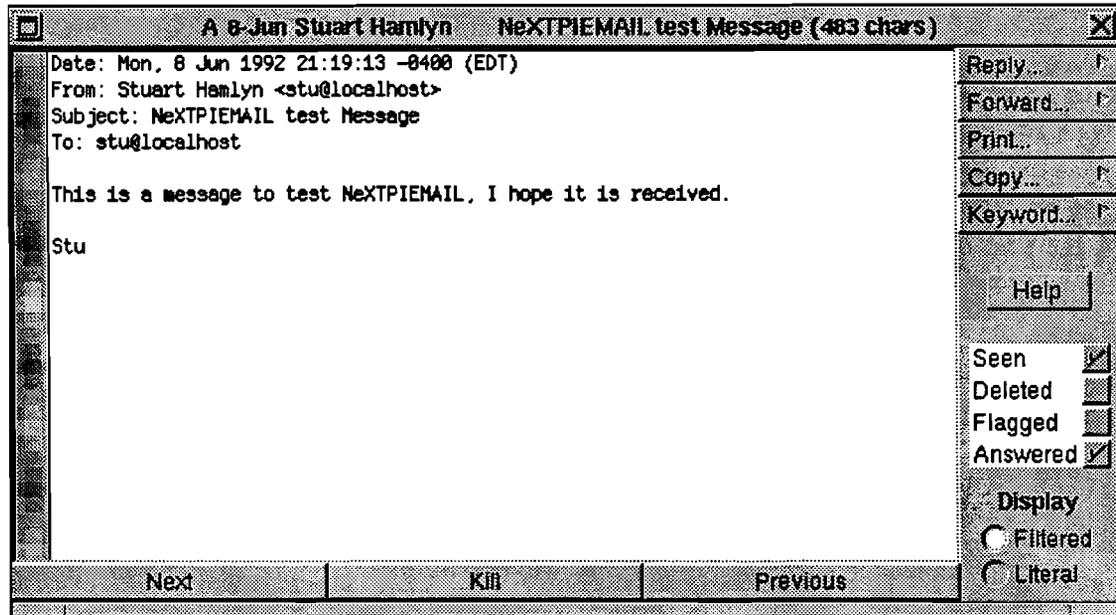
- A high-speed message pattern matching algorithm is contained in [MBoxWindow Select]. From a simple form that resembles the message composition window, the user can enter his query, click <ok> and all requests matching his query are highlighted. The following illustration demonstrates how to select all messages from the first quarter from your boss concerning your salary, for example.



Showing all mail from "the boss" last quarter

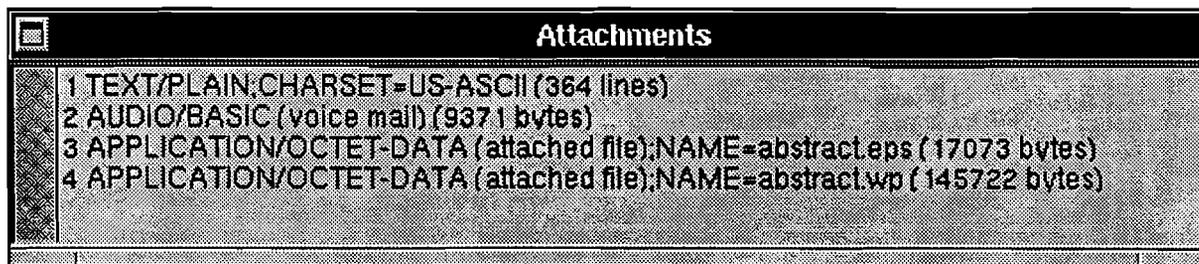
4.3.3 ReadWindow Class

The *ReadWindow* object is instantiated when the user clicks on a message summary line in an *MboxWindow* object. It handles the window for reading a message and its associated attachments. Multiple *ReadWindow* objects can be displayed on the screen at once, additionally if more than one *ReadWindow* exists for a single message, the other is updated automatically.



Example of Filtered Message ReadWindow

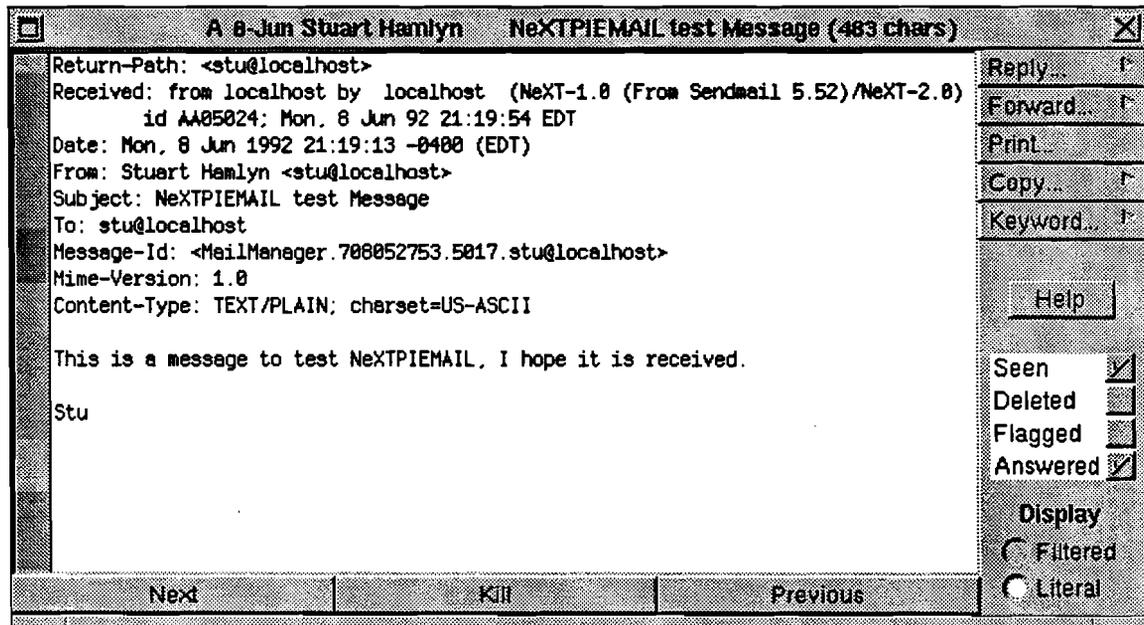
- Multimedia attachments can be viewed from the message reading window. By clicking on <view attachments> a list can be previewed detailing exactly what kinds of attachments are contained within a message.



NeXTPIEMAIL shows the list of attachments in a message before displaying them

- Facilities are provided for moving to the next message in the current mailbox, the previous message, or to move to a new mailbox.
- The user can copy or move the message right from the read window without moving to the browser window

- The following flags can be viewed and/or modified: **answered, deleted, flagged, seen**. The modification feature is provided so that future versions can have user customizable flags.
- If desired, the unfiltered RFC822 Internet mail-message information can be viewed. This may be desirable for power users trying to track down particular message id's, for example.



A literal view of a message (include Internet header info)

4.3.4 SendWindow Class

SendWindow offers the user a form to fill out that matches the necessary fields in an RFC-822 Internet message. The fields of the form are fully editable and support cut/paste, etc.. The form is user friendly and interfaces with the rolodex.

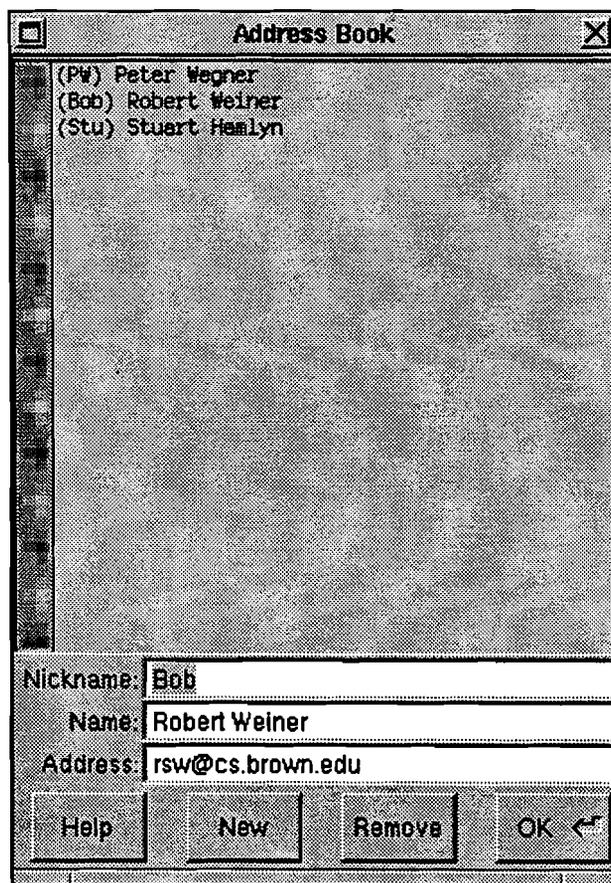
The NeXTPIEMAIL user-friendly mail composition form (SendWindow)

- The user is obviously allowed to edit the envelope fields and the body field of the message. The address book is "live". That is, the user can type a nickname and NeXTPIEMAIL will automatically substitute the proper E-mail address.
- An attachments manager is included with *SendWindow*. Sounds can be inserted and deleted. This multimedia support is particularly impressive on the NeXT since it has a built in microphone and excellent graphics.
- On line help is available.

4.3.5 AddressBook Class

The class maintaining the address book is quite simple. It is simply a window manager and a tiny database implemented as a text file reader. It supports the setting of four instance variables : Nickname, Name, Address, and AddressItems

- Future versions of the addressbook should hold icons and biographical information about the entry. This is an easy addition
- On-line help is included
- Add / Remove buttons are provided to edit the list of addresses in the book.
- A lookup method is provided, thus classes such as *ReadWindow* and *SendWindow* can easily switch from the abbreviated nickname to the full Internet address. This is necessary to route outgoing mail based only on a nickname.



The NeXTPIEMAIL AddressBook

5. NeXTPIEMAIL Implementation

As mentioned in "NeXTPIEMAIL Design" on page 15, NeXTPIEMAIL was written in Objective C on the NeXT Workstation. The design tools of the NeXT including a modified version of the GDB debugger, the user interface design tool, *Interface Builder*, and the NeXTSTEP class library were heavily relied upon. The mail access, IMAP drivers developed by Mark Crispin, written in ANSI C were also critical in the development of NeXTPIEMAIL, as were Crispin's *MailManager* class objects - which provided the foundation of the user-interface. In the following sections we shall tell the tale of implementation and highlight key issues of object oriented interface development on windowing environments.

5.1 The NeXT Development Environment

The NeXT Workstation development environment, with its rich object library and *Interface Builder* - user interface design tool, offered an excellent platform. All of the graphical elements of NeXTPIEMAIL were prototyped first on *Interface Builder* without a single line of code. The interface was then "parsed" into Objective C by *Interface Builder* - and the underlying code to fill the objects was written. Take for example, the task of creating the message browser: the message browser with all its windows and buttons - was drawn first with *Interface Builder*, parsed, then code was written to insert the proper message header information into the blank text-fields created by *Interface Builder*.

Abstracting the interface from the underlying code is a significant advantage in developing object-oriented software. Without this abstraction, creating personalized, graphic environments would be prohibitively tedious. To produce effective interfaces, one must be able to test them first without the overhead of implementation. In developing NeXTPIEMAIL, over a dozen MBoxWindow interfaces were tested - some before any underlying code was written, and some after. If one had to manually generate the underlying Objective C code to test each interface, and recompile the program - surely only three or four interfaces would have been tested and performance would have been sacrificed.

Another critical feature for object-oriented development that proved itself invaluable was the object-browser extension to GDB [*Stallman87*] [*Next89*]. Browsing objects as a program is running, "peeking" in NeXT terminology, greatly enhances debugging. One no longer needs to follow complicated stack frames and print out obscure memory locations. Tracing bugs becomes a matter of tracking message calls and watching instance variables change. When one spots the erroneous instance variable being set, it is a simple matter to trace out which object sent the problem message and correct the underlying code.

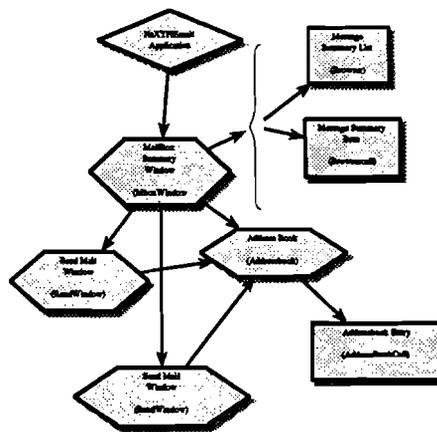
5.2 Objective C - Class Organization and Code Sizes

Objective C, unlike its cousin C++, is a true object-oriented language. It is more similar to Smalltalk in many respects than C. Without such a tightly structured OO language, it

would have been vastly more difficult to attain NeXTPIEMAIL's flexible user-interface and underlying code design.

With Objective C it was a simple matter of designing one major "class" for each user-interface or program view. For example, an *Application* Class was defined to handle logging on, another class was created for the summary view of messages, another for the address book, etc... Following this scheme, it was a simple matter of adding methods to each of these classes to provide new functionality. For example, to add the conversation tracking, one method was created inside the MBoxWindow (summary view) class.

The program segments and their respective code-sizes are listed below.



Dependency of Major Objects in NeXTPIEMAIL
(target object instantiated by source)

Table 1: Major Object Class Sizes / Functionality

Class	# Lines	Functionality
NeXTPIEMAIL	1400	Instantiates all objects, handles logging on, automatic filing, and sets all mail stream information
MBoxWindow	900	Message Summary View, selection, conversation tracking, querying, contains main menu
ReadWindow	100	Manages view of single message, contains shortened version of main menu
SendWindow	200	Manages form for sending window

Table 1: Major Object Class Sizes / Functionality

Class	# Lines	Functionality
AddressBook	90	Contains small database for rolodex and has look-up functions for other classes
BrowserCell	30	Class for single-line in the summary-view window, subclass of NeXT MatrixCell
AddressBookCell	30	Class for single line in the address book
Browser	150	Subclass of Matrix - specialized for managing message summary view items.

From the Table 1, it is apparent that 90% of the functionality is contained in the two largest classes, *NeXTPIEMAIL* and *MBoxWindow*. While these perhaps seem like unmanageably-large chunks of code, they are not. Most features are implemented by one method in either *MBoxWindow* or *NeXTPIEMAIL*. None of these methods are over two pages in length, and each class has less than 20 instance variables it shares between its methods.

This tight code is due to both the exceptionally rich class-library available on the NeXT and to Objective C's ease of sub-classing and type independent messaging system.

5.3 Mark Crispin's C-client and MailManager

The author of *NeXTPIEMAIL* is greatly indebted to Mr. Mark Crispin, who has developed a high-level library of mail access functions in C. Without this "C-Client" *NeXTPIEMAIL* would be a considerably larger and less modular application. This library was used as extensively as the Objective C class library supplied with the NeXT in the development of *NeXTPIEMAIL*. The Documentation of the C-Client is included in the last appendix with the source code of *NeXTPIEMAIL*.

Mark Crispin also wrote a program called *MailManager* which accesses the C-Client and runs on the NeXT. It is equipped to handle standard electronic mail processing on multiple hosts across multiple servers. It is on top of this infrastructure that *NeXTPIEMAIL* is built.⁴

4. Note, as of this printing, the 15 December 1991 version of the C-Client is being used. Many bugs in *mail_copy* and *mail_move* have been discovered, later versions should be sought from Crispin if one is intending to modify *NeXTPIEMAIL*.

6. Future Enhancements

6.1 Performance

As the system is still in its alpha stage, performance issues are obviously not a priority. In terms of speed, the system is entirely acceptable. The only drawback arises with large message bases during start-up time. Since the system redistributes mail each time it starts up (i.e. it takes ALL messages from a default mailbox and files them according to the roles the user has defined) with over 1000 installed messages this may produce significant delays. When the C-Client mailbox routines are more thoroughly supported, it will be easy to optimize this process.

6.2 Nested Folders

In the current version, mailboxes (folders), are only flat-file entities. It is desirable to have nested folders. That way, one could store topic related information in one folder and have conversations automatically tracked and stored in sub-folders. Any query done at the outer folder would automatically span all of the inner-folders, while queries in the lower level sub-folders would span only the messages or conversations at hand.

6.3 Querying Across Folders

The ability to query across more than one folder is obviously desirable, since one may not know which folder a particular message was filed in.

6.4 Storing Queries

It is also desirable to have some manner of storing queries. Storing the selection criteria for a query would work as a sort of macro. For example, one could define a macro to find all of the messages received or sent in the last week regarding a particular project. From this point it is a simple matter to hint <print> to get a hard-copy of the progress of a project for a manager's meeting. Storing and naming the search request would act like storing a macro.

6.5 Task and Role-based Enhancements

In the current version, the user is asked upon start-up if the system should tell all requestors about his/her roles. That is if someone sends the message "stu.whoareyou", the system will flag this and ask the user (stu) if the originator of this message should be sent a list of the tasks or roles defined for the user (stu). It is desirable to automate this, so that it happens immediately and not simply when the system is started. It is also desirable to have classified roles that the system does not broadcast to any users.

7. Conclusion

NeXTPIEMAIL offers substantial increased functionality with no alteration of the underlying, physical E-Mail messaging system. We can summarize them as follows:

- Automated mail processing via role/task definitions
- Grouping of messages via interactive mailboxes
- Selecting message groups via point & click browsing
- Effective mail retrieval via queries
- Conversation tracking
- Addressee management via interactive rolodex support

The NeXTPIEMAIL design philosophy is heavily user-centered for two reasons. NeXTPIEMAIL seeks to demonstrate the productivity impact of graphical user interfaces (GUIs) and also seeks to demonstrate the use of E-mail as a critical element of the collaborative work group. Both of these goals rely on a formidable user-interface. Furthermore, it is the hope of the designers that these systems are adopted as both end-user tools and frameworks for targeting specific mail and information delivery applications.

Although the modifications mentioned in the previous section are commendable and would make PEmail systems state-of -the-art information processors, fundamental change must occur on the user side of the equation before these systems are implemented. It is our hope that those who use PEmail systems will stop thinking of electronic mail and digital communications as glorified voice-mail or messaging systems accross which disposable information is sent; and begin to think in terms of using such telecommunications as a vehicle for bridging geographic distance and opening up new vistas for collaborative work.

Appendix A : Sample User Scenario

In order to demonstrate the functionality of NeXTPIEMAIL in an easy to understand manner, we have created a typical scenario representative of PIEmail users. The scenario chosen represents that of two students working jointly on a Masters thesis.

Appendix A.1. Role Definitions

Stuart Hamlyn, one of Brown University's typical Renaissance students fulfills many roles in life, three of which he tells his PIEmail system about. In particular he has one mailbox set up for each of his three interests: *art*, *music*, and *school*. The NeXTPIEMAIL system will automatically file any messages with the "subject" flags into the appropriate mailboxes for Stuart, automatically categorizing his mail.

The screenshot shows a dialog box titled "Role Definitions" with a close button (X) in the top right corner. It contains a table with two columns: "Mailbox" and "Subject". The table has 11 rows. The first three rows are filled with "art stuff", "school stuff", and "music stuff" in the "Mailbox" column, and "art", "school", and "music" in the "Subject" column. The remaining eight rows have "NA" in both columns. At the bottom of the dialog box, there are two buttons: "Cancel" and "Ok" with a return key symbol.

Mailbox	'Subject'
art stuff	art
school stuff	school
music stuff	music
NA	NA

Stuart's Roles

John Weiner, an engineer for Motorola, a colleague of Stuart's and also a Renaissance man enjoys four specific roles that refer to different tasks in his life and sets up his PIE-mail system accordingly. John has folders configured for: *personal*, *motorola*, *student*.

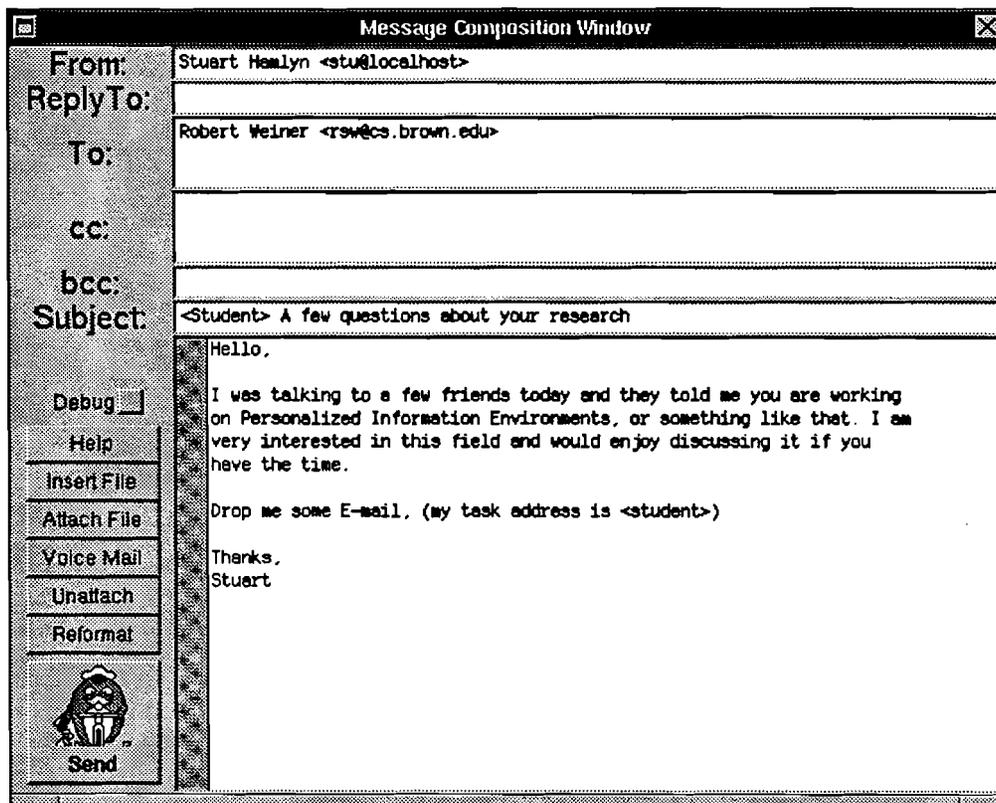
Appendix A.2. Message Composition

One fine morning Stuart comes to the sudden realization that he had better get started on his Masters Thesis. He speaks with a couple of colleagues in his office who tell him that this fellow, John Weiner, is working on an interesting desktop productivity application. Stuart decides to mail him a letter about his workgroup productivity interests.

Stuart then enters the NeXTPIEMAIL system and invokes the <role query> function which tells him what roles John has defined for himself.⁵ The system tells Stuart that the following roles for John Weiner exist:

- John.student
- John.personal
- John.Motorola

Stuart assumes that his message relates closest to John's *student* folder, so he invokes the message composition command and includes "student" in his subject header.

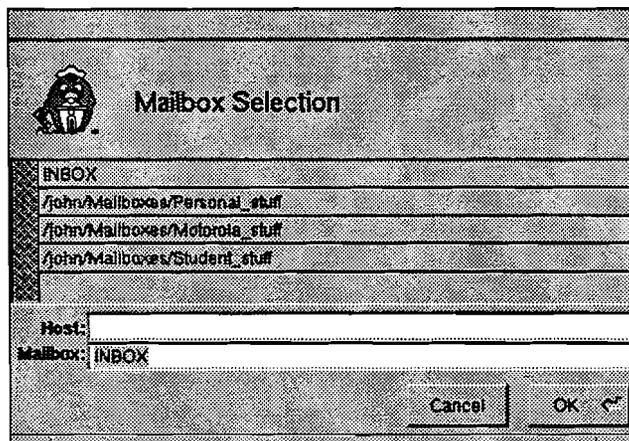


Stuart sends letter to John exercising the compose message function

5. Note, The <Role Query> function has not yet been implemented due to a lack of knowledge about modifying the UNIX *sendmail* program. For now, we will just assume that John's roles were published somewhere in the office.

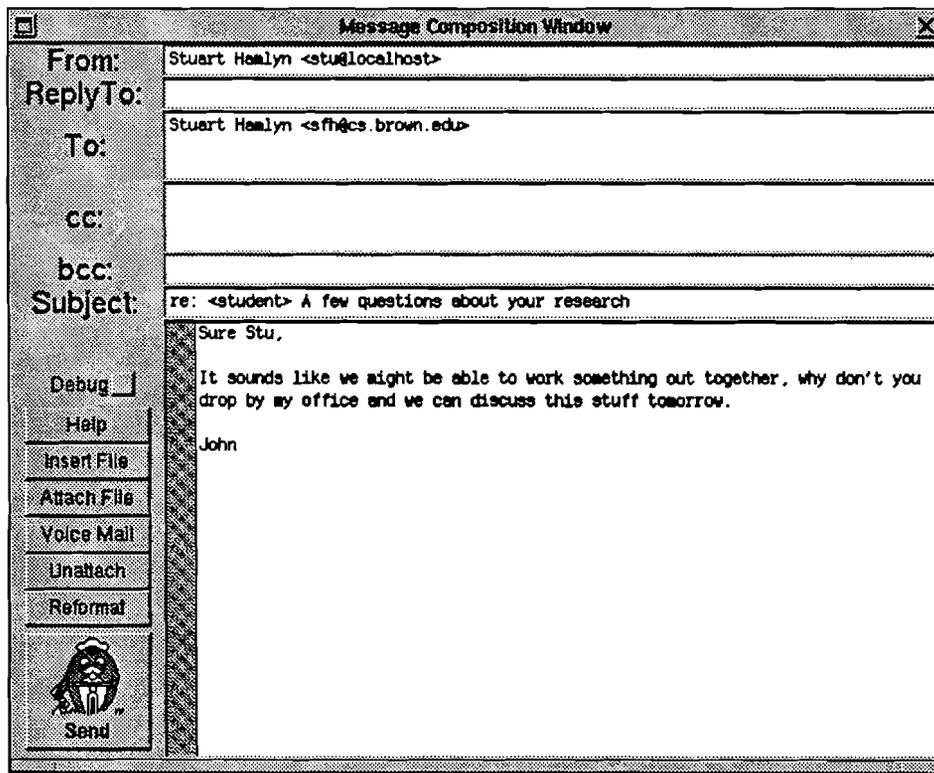
Appendix A.3. Message Reply

John logs on to NeXTPIEMAIL and notices that the folder entitled "Student_stuff" is highlighted, so he double clicks on it and opens up the message view.



Mailbox Selection window showing roles

Bob wonders if he has ever communicated with Stuart before, and if so, about what, so he looks up his name in the address book. The search comes up blank, so he decides to send a message back to Stuart suggesting that they meet and discuss their ideas together. He accomplishes this from selecting the reply command from the message-edit view.

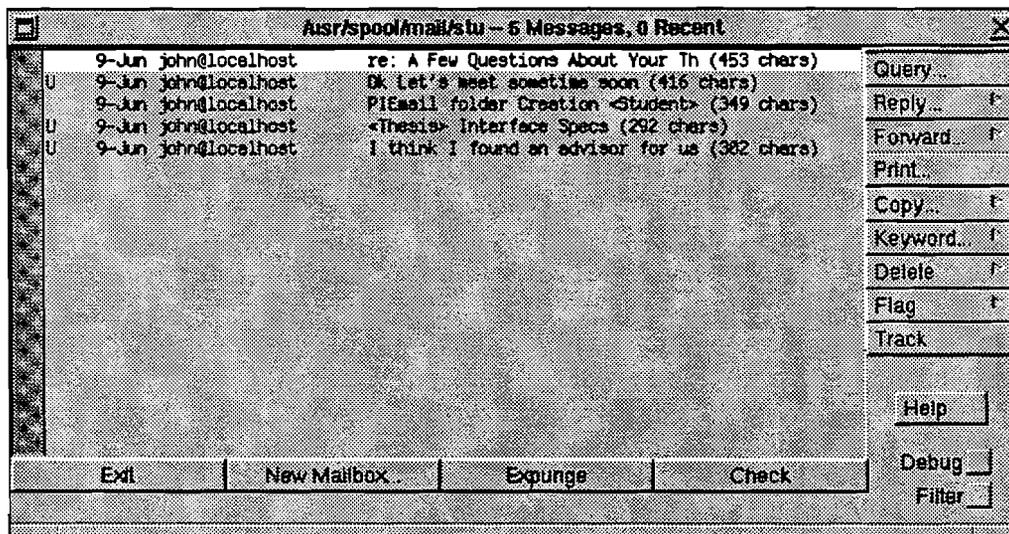


The reply to the first message

Appendix A.4. New Role/Task Creation

Stuart and John meet, discuss their ideas and agree to work on some initial project ideas independently, type up their thoughts, and meet again to review their progress. They will send each other their work electronically as it develops to make sure their ideas are in sync. Both parties agree to add one another to their respective address book and to create new task folders where their mail for this particular project will be filed. They each create new folders called "thesis_work" where any subject line with the word "thesis" in it is to be filed automatically by the PImail system.

The "thesis_work" folder begins to fill up with messages. Here is an idea of what it might look like after some correspondence between John and Stuart.



The message browser after a bit of correspondence

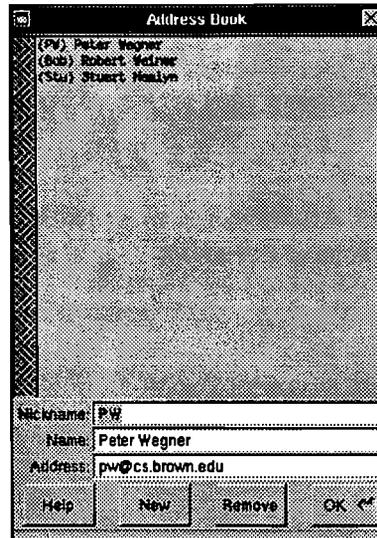
Appendix A.5. Conversation Tracking

John locates a professor, Peter Wegner, who is interested in their work and wants to be brought up to date ASAP. John has three options: He could go through all the messages he feels professor should read click on them, and click forward to send the group, or he could select the whole <thesis> folder and send it, or he could exercise the innovative NeXTPIEMAIL feature, <track> He chooses the latter.

John simply clicks on one of the design letters Stu sent him, then clicks <track>, and all the messages that relate to it (are part of the reply-to-chain) are highlighted and ready to send to Professor Wegner by just clicking forward.

Appendix A.6. Address Book Addition

Stuart finds about Professor Wegner and decides to add him to his address book, so he can correspond with him more easily in the future. This is accomplished by simply bringing up the address Book and clicking <New>



Appendix A.7. Query Command

Now let us assume Stuart sends John a message asking him if he knows anything about user interfaces on the NeXT Computer. John thinks that someone at work might have said something about interfaces but he can't remember if he mentioned the NeXT. It is a simple matter to find out. John simply clicks on his <Motorola> folder and then does a query for messages containing either "interface" or "NeXT" in them. If any exist, they are highlighted automatically to send on to Stuart.

 A screenshot of the "Message Selection Criteria" dialog box. It has several sections:

- From:** [empty field]
- To:** [empty field]
- cc:** [empty field]
- Subject:** [empty field]
- Text:** "NeXT || Interface"
- Other:** [empty field]
- Message Date(s):**
 - Date Before After [empty field]
 - Any Recent
 - Old New
- Message Status:**
 - Any Seen Unseen
 - Any Deleted Undeleted
 - Any Flagged Unflagged
 - Any Answered Unanswered

 There are "OK", "Help", and "Retain" buttons at the bottom right.

A Sample Query Form

Appendix B : NeXTPIEMAIL Source Code

The NeXTPIEMAIL program is reprinted here 2-up in order to save paper. The program is written in Objective C with the exception of Utilities.h which is written in standard ANSI C. The program consists of the following main modules. The header files are .h files and the source code is contained in the .m files.

Appendix B.1. MailManager

MailManager contains the main code for the application and instantiates the other objects. It handles logging on, preferences, etc.. See "NeXTPIEMAIL [Application] Object" on page 18.

Appendix B.2. MBoxWindow

The MBoxWindow handles the browser and all associated functionality. See "MBoxWindow Class" on page 20.

Appendix B.3. SendWindow & ReadWindow

These classes handle sending and reading messages respectively. See "SendWindow Class" on page 24.

Appendix B.4. AddressBook

AddressBook handles all access functions associated with the address book. See "AddressBook Class" on page 25.

Appendix B.5. AddressBookEntry

This is the content-record of the address book

Appendix B.6. Browser

This is a customized version of NeXTSTEP's Matrix Class, it is customized to handle NeXTPIEMAIL's selection criteria.

Appendix B.7. Browsercell

This is the item on the message summary-line. It formats the text to line up the subject, date, etc...

Appendix C : References

The following is presented as a Bibliography / list of suggested background reading for those interested in personalized environments, mail systems, or collaborative work group support software.

- [BoFr92] N.S. Borenstein and N.Freed. *MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies*. Published as Internet Drafts, June/October/December/February, 1991/1992, Proposed Internet Standard (expected April 1992)
- [BoTh88] N.S. Borenstein and C.A. Thyberg. Cooperative work in the Andrew Message System, *Proceedings of the CSCW'88 ACM Conference on Computer Supported Cooperative Work*, 1988, pp.306-223.
- [BoTh91] N.S. Borenstein and C.A. Thyberg. "Power, Ease of Use, and Cooperative Work in a practical Multimedia Message," *International Journal of Man-Machine Studies*, 1991, pp.229-259
- [CaRoMa91] S.K. Card, G.G. Robertson, and J.D. Mackinlay. The Information Visualizer, An Information Workspace, *Proceedings of the ACM CHI'91 Conference on Human Factors in Computing Systems*, 1991, pp.181-188.
- [CaSh91] R.Carr and D.Shafer. *The Power of PenPoint*. Addison Wesley, Reading, Ma, 1991.
- [Crocker82] D.H. Crocker. *Standard for the Format of ARPA Internet Text Messages*. Internet Request for Comments Number 822 (RFC-822), Department of Electrical Engineering, University of Delaware, August 13, 1982.
- [Crispin92] M. Crispin. "MailManager Program," Available via anonymous ftp from [sonata.purdue.edu](ftp://sonata.purdue.edu)
- [DoOr85] J. Donahue and W. Orr. *Walnut: Storing Electronic Mail in a Database*. Xerox Palo Alto Research Centers Technical Report, CSL-85-9, November 1985.
- [Engelbart82] D.C. Engelbart. "Toward High Performance Knowledge Workers," *Proceedings of the AFIPS Office Automation Conference (OAC '82 Digest)*, San Francisco, CA April 5-7, 1982, pp.279-290.
- [Horton83] M. Horton, *Standard for Interchange of USENET Messages*. Internet Request for Comments Number 850 (RFC-850), USENET Project, June 1983.

-
- [KaLoCaLa92] S.Kaplan, C.Lowe, A.Carrol, D. LaLiberte. *Epoch: GNU Emacs for the X Windowing System*. Release 4.0, University of Illinois, Urbana Il, March 1992
- [Mackay88] W.E. Mackay. "More than Just a Communication System: Diversity in the Use of Electronic Mail," *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, 1988, pp.344-353.
- [MaGrLa87] T.W. Malone, K.R. Grant, K.Y. Lai, R. Rao, and D.R. Rosenblatt. "Semistructured messages are surprisingly useful for computer-supported coordination", *ACM Transactions on Office Information Systems*, 1987, 5(2), pp. 211-216
- [Next89] Next Computer Inc. "NeXTSTEP Reference", "NeXT Development Tools", "NeXT Operating System Software", Addison Wessley, 1989.
- [Stallman87] R.Stallman. *GNU Emacs Manual*. Free Software Foundation, Cambridge, MA, March 1987
- [Terry90] D.B. Terry. "7 Steps to a Better Mail System," Xerox Palo Alto Research Center Technical Report, CSL-90-12, September 1990.
- [Weiner92] B. Weiner. "Hyperbole Manual." Brown University, Providence, RI 1992. Available via anonymous ftp from: /anonymous@wilma.cs.-brown.edu:pub/hyperbole.
- [Weiner92] B. Weiner. "PIEmail Technical Report," Brown University, Providence, RI 1992.