# CS190 Birch Specifications Document

Junichi Kimura (jun)
Christine Waggoner (cmw)
February 15, 1999

## 1 Introduction

### 1.1 Document Overview

This document is a draft of the specifications for Birch, a Unix email client with sophisticated data manipulation features, aptly named after Pine and Elm.

### 1.2 Problem Statement

This project seeks to address the following two major problems:

- On Unix systems, there is a general lack of GUI-based mainstream email clients. While this dearth can be attributed to the general culture of Unix users, novice Unix users often encounter difficulties executing even the simple task of sending and receiving email. Consequently, many novices turn to WIMP email clients such as Netscape Mail and Microsoft's Outlook Express, whereas more experienced users prefer traditional mailers with text-based interfaces such as mail, Elm, and Pine. This rift between a novice's and an expert's needs is omnipresent in computing. More specifically, it remains an open question as to whether an email client for Unix which appeals to both novice and expert users can be created.

- The mailbox folder paradigm used in most popular mailers today encounter problems when used in conjunction with the filtering mechanism available in many email clients. Usually, filtering works by moving messages that fit certain conditions into another mail folder. This scheme breaks down when a message meet the condition of several filters simultaneously. For example, a message from John Doe regarding the CS190 project may fit the filters "all mail from John@doe.com", "all messages with cs190 in the subject", and "all messages in the past two days". In such cases a copy of the message will be created in all folders that the message needs to be rerouted to.

### 1.3 Solution Overview

Proposed solutions to the lack of email clients for both novices and experts and to the inadequacies of current email clients' data manipulation capabilities:

- In order to appeal to both new users and experts, Birch will reach a middle ground between completely text-based, keyboard-driven interfaces and cumbersome WIMP-based interfaces. Birch will be endowed with numerous keyboard shortcuts and a WYSIWYG WIMP interface. Birch's primary interface will be a graphical one. However, Birch's visual interface will be designed with the principle "less is more" in mind. Birch's interface will not be overly crowded with icons and buttons. Moreover, Birch may provide a shell-based interface to serve experienced Unix users and to facilitate remote access, i.e., the invocation and running of Birch in a shell.

- As some existing email clients, Birch will filter mail by treating the user's mailbox as one large database. Birch will specifically skirt copying the same message to multiple folders and will continually refer to a central mail respository. In addition, Birch will allow users to specify and save filters that will allow the manipulation of messages that meet the filter condition. In addition, users will be able to cross-reference a single message from several filters, and be able to combine existing filters to create a new filter. Finally, it should be noted that users will not only be able to view messages by using filters, but also to delete messages.

### 1.4 Client and Target Users

Birch was conceived as a class project for CS190, Software Systems Design at Brown University by Junichi Kimura and Christine Waggoner. The primary motivation for creating Birch comes from Kimura's and Waggoner's personal experiences on Unix systems at Brown.

Potential target users of Birch will be the inhabitants of the Brown Computer Science Department. While Birch could be use to people outside the the Brown CS Department, it will be designed primarily with Brown CS students, staff and faculty in mind. More specifically, Birch will be designed as a general purpose email program to be used by individuals and not for example, as a business email application, as an email app to allow computer artists to view multimedia attachments, or as a groupware program.

## 2 Functional Specifications

### 2.1 Basic Email Functionality

#### 2.1.1 Message Transmission

This version of Birch will use MH as its messaging back-end. Unix users with an account already set up for electronic mail should not have to modify their configuration in order to use Birch. Users can also use Birch as a graphical front-end to MH, such as xmh and exmh.

#### 2.1.2 Message Composition

Birch will provide the user with an editor window to compose and review messages. Users will have the option to use their own editor, if this is preferred.

Birch's message editor will provide the following functionality:

- GUI-based scrollable text editing window with standard cut, copy, paste behaviors
- Text fields to allow the user to enter the recipient(s) and the subject of the message. Additional GUI elements to let the user specify recipients from the address book may be included.

- The ability to save messages for future editing and transmission
- The ability to discard a message
- Configurable keyboard shortcuts for the commonly used commands
- No styled text support is planned for the initial implementation. If time permits, MIME styled-text and/or HTML message editing may be supported in the editor.
- Possible multi-language support as time permits.

### 2.1.3 Message Viewing

Messages will be displayed in a window similar to that of the message editing window. It will have the following functionality:

- Display the sender's name and email address
- Display the subject and time of the message
- Scrollable text window displaying the message, with an option to display/hide the message headers
- Allow the user to reply to, forward, and delete the message
- Allow the user to save any attached files
- Allow the user to move a message to a different folder. Although this mechanism is not strictly necessary due to the folderless filtering mechanism of Birch, users may find it useful when using Birch in conjunction with plain MH.
- Configurable keyboard shortcuts for the commonly used commands

In addition, there will be a message listing window that will list multiple messages. It will have the following functionality:

- List the messages according to the current filter setting
- Sort messages
- All the functionality of the message viewing window except for the actual display of email. An option may be included which will allow the user to switch between having a separate message and message list window (a la Eudora), and a split-pane approach (Netscape Mail, Outlook Express).

### 2.1.4 Address Book

Birch will have an address book of email addresses. Users will be able to add, delete, and remove email addresses to the address list, and have easy access to these when composing email. In addition, users will be able to assign nicknames to address book entries, in order to make entering addresses easier.

As time permits, time-saving features such as auto-completion of addresses may be included.

### 2.1.5 Attachments

Birch will support standard MIME-compliant multipart messages. Support implies that Birch will recognize that the message contains nontext data and allow the user to save the attachment to a file. It may not necessarily be able to decode or display the file. For example, Birch will not be able to decode attached Macintosh files encoded in MacBinary format. Users will need to save the attachment portion to a file and run a decoding software in order to use the contents of the attached file.

## 2.2 Message Filtering

### 2.2.1 Birch's Filtering Paradigm

Birch treats its mailbox differently from the other mailers in that all mail is treated as if they were in one large folder. Users customize which mail are listed in the mail listing window through the use of filters. Mail folders will be supported for backward compatibility with MH, in order to enable users to switch between Birch and MH.

### 2.2.2 Filter Conditions

Birch will allow the user to filter messages according to these catergories:

- Sender
- Recipient
- Date
- Subject
- Message Contents
- Priority
- Status
- User-defined Labels

Users will be able to specify an arbitrary number of conditions for each filter. In addition, users will be able to create new filters from existing ones.

### 2.3 Configuration

Birch users will be able to customize the functionality as well as look and feel of the program through GUI preference panels. These panels will allow the user to specify settings for features such as Filters and address books.

In addition to the GUI preference panels, users will be able to customize the settings by editing the Birch configuration files directly. The format of the configuration files will be specified in the user documentation.

### 2.4 Miscellaneous Functionality Specifications

#### 2.4.1 File Formats

Birch will use the identical mailbox file format and configuration files as MH. Detailed specifications of the MH mailbox and configuration file formats can be found in the mh (1) man pages, as well as in the book *MH & xmh: Email for Users & Programmers*, ISBN 1-56592-093-7, written by Jerry Peek, published by and is copyright ©1991, 1992, 1995 by O'Reilly & Associates, Inc. The online version of the book is available free of charge at http://www.ics.uci.edu/~mh/book/

In addition to the MH configuration files, Birch will use its own configuration files to store settings specific to Birch. The format of such configuration files will be noted in the user documentation.

#### 2.4.2 Error Handling

Whenever possible, Birch will alert the user of a possible error conditions and provide error recovery information. Birch will periodically create backups for files currently being edited in order to prevent data loss.

## 3 User Interface Specifications

These specifications begin by outlining the main principles of the user interface. Secondly, a broad overview of the conceptual design is given. Finally, mock-ups and the semantic design are given for the major components of the program: the mailer/text editor, filter panel, and addressbook.

### 3.1 User Interface Mission Statement

- As stated in the problem statement, one of the major problems in creating software is to accomodate the needs to both novice users and expert users. While it is perfectly acceptable to acknowledge the fact that a single piece of software can not meet the needs of every user, software should still be designed so that is easy to use as a novice, but not aggravatingly simplistic for an expert. However, to design a piece of software which is usable by both novices and experts is easier said than done. Nonetheless, Birch will make it one of its primary goals to provide decent functionality in the eyes of the novice and expert user.

- Birch's user interface will also be built on the principle that "less is more." Since Birch is a non-commericial product at this point, Birch is poised in the unique position of not being plagued by "featurism", the need to constantly add new features to the program and to display all the new features prominently in its UI. Consequently, Birch's interface will strive to be simple yet robust.

## 3.2 Analysis of Other Email Interfaces

Here are analyses of some commonly available email clients. Not every single existing mail client is analyzed, but a wide variety of email clients is represented. This analysis is intended to help further define the design for Birch.

### 3.2.1 Netscape Mail

Pros:

- Provides a lot of functionality in a single application. For example, a user can read email, write email, send email, receive email as well as read newsgroups and post to newsgroups from within its Messenger Mailbox interface.
- Elements of the interface are customizable.
- Provides some context-sensitive help
- Provides some keyboard shortcuts
- Spell-checking included

Cons:

- Interface is jumbled with large icons and buttons
- Originally, unclear how to address messages
- Difficult to separate news-reading functionality from personal email functionality
- Assumes that the addition of hypertext constructs to email are desirable
- Clutters large portion of message editing interface with options to add hypertext constructs to messages
- Clutters the mail program with Communicator's Web browsing interface
- Takes up a large portion of the screen
- Clutters the screen with multiple windows

### 3.2.2 Microsoft Outlook Express

Pros:

- Leads the user through initial configuration requirements such as email address and name when the mailer is first started
- Does not clutter the mail interface with InternetExplorer's Web browsing interface.
- Provides restricted message filtering
- Clearly explains the Bcc and Cc fields

Cons:

- Does not display context-sensitive help to explain icons
- Takes up a large portion of the screen
- Icons and toolbar are extremely large
- Incredibly slow
- Clutters large portion of message editing interface with options to add hypertext constructs to messages

### 3.2.3 mail

Pros:

- Compact and fast
- Easy to use remotely during a telnet session

Cons:

- User must memorize commands and read documentation to use well
- Provides absolutely no context-sensitive help

### 3.2.4 The Mutt Mail User Agent

Pros:

- Compact and fast
- Possesses threaded sorting mode
- Allows user to use text editor of choice
- Supports "scoring", i.e., sorting by pattern matching and ranking

Cons:

- Does not resize well on some systems
- User must memorize commands and read documentation to use well

### 3.2.5 Pine

Pros:

- Provides a simple and clear interface for essential functionality
- Compact and fast
- Allows user to use text editor of choice
- Easy to use remotely during a telnet session

Cons:

- Many keyboard commands involve the use of the control key in conjunction with other characters. This is displayed with a caret (ˆ) symbol on screen, and it is not obvious to the new user exactly what they are supposed to be typing.
- Customization is achieved through a complex list of options, and editing some options involve opening the configuration file in a text editor and entering the settings by hand.

### 3.2.6 Elm

Pros:

- Provides a simple and clear interface for essential functionality
- Compact and fast
- Allows user to use text editor of choice
- Functions well in a single screen
- Easy to use remotely during a telnet session
- Interface conducive to learning set of commands to becoming an expert

Cons:

- User must memorize numerous commands and read documentation to use well

## 3.3 Conceptual Design

### 3.3.1 Metaphors

The primary metaphors that will be used in Birch already exist in common email applications.

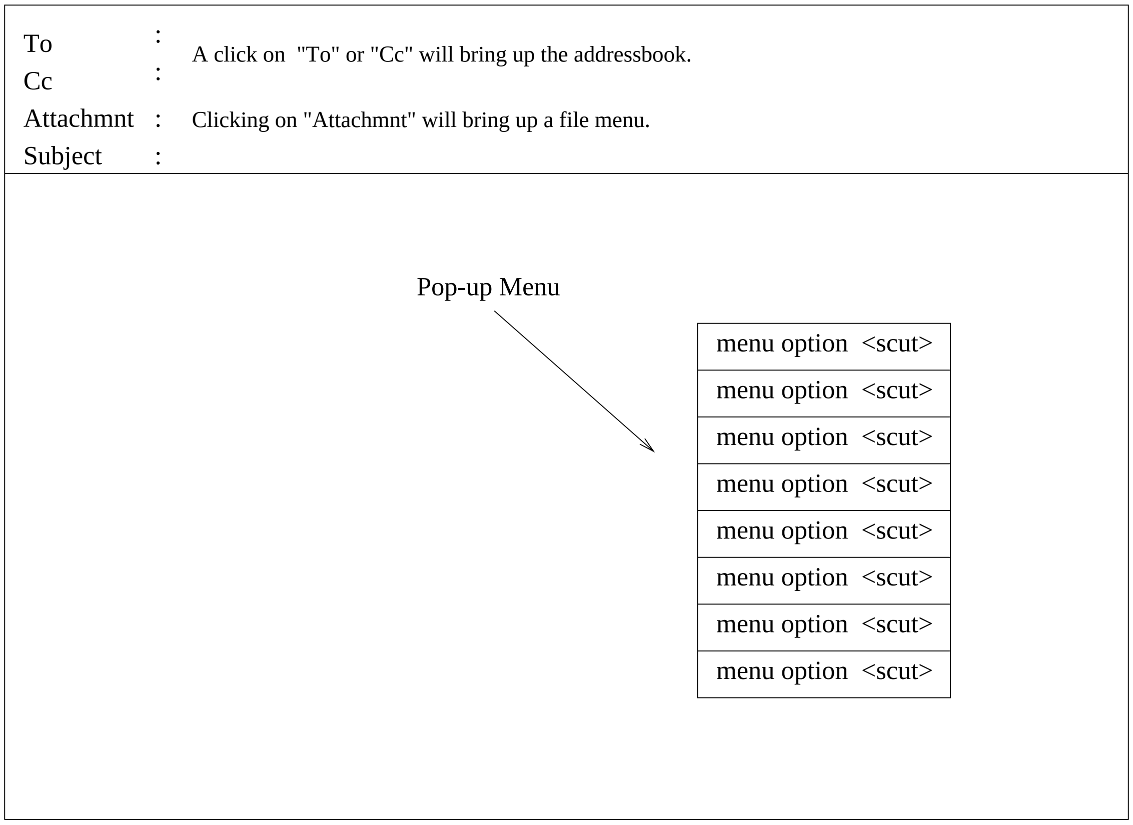
- As usual, the mailbox folder metaphor will be used to allow users to organize and categorize messages.
- The addressbook metaphor will also be used by Birch so the user can easily view and organize personal contacts.

### 3.3.2 Look and Feel

- As stated in section 3.1, Birch's interface will be simple and robust. The interface will look simpler than Netscape Mail and Microsoft's Outlook Express, but will provide a more visual interface than Pine, Mail, Elm, or Mutt. Birch's default look will be akin to Microsoft Outlook Express's, but will not use as many icons and large buttons as Outlook Express. In addition, unlike Netscape Mail and Outlook Express, Birch will default to mostly terse keyboard shortcuts, thereby relating strongly to traditional Unix programs such as mail. However, the keyboard shortcuts will be fully customizable. In sum, Birch will look simpler than most modern software applications and be more visual than Unix programs written for vt's, but will feel based in Unix.

## 3.4 Mailer Interface

### 3.4.1 Visual Design

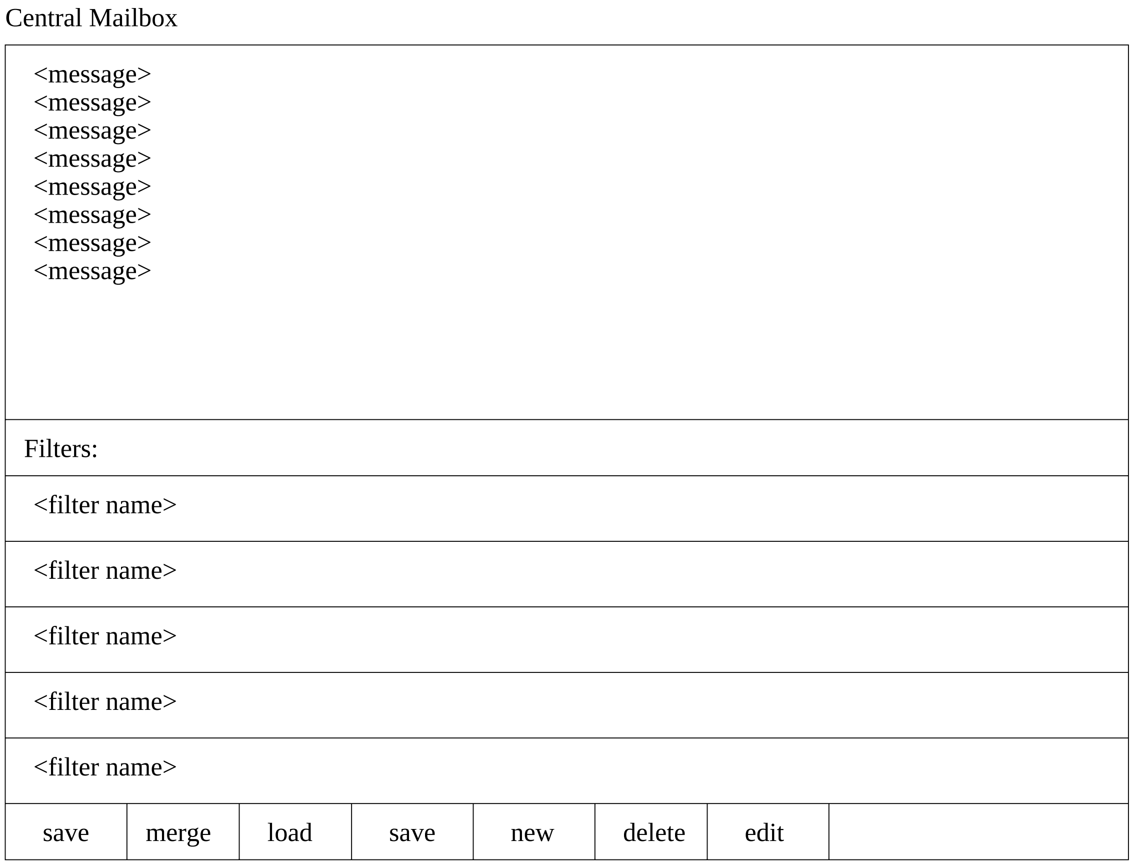


### 3.4.2 Semantic Design

The broad goal of this component of Birch is simply to allow the user to write and send email messages. The editor is extremely simple. The user can click on the components of the header for further assistance in choosing the element to complete the fields of the header. For example, if the user is having trouble remembering an email address and wishes to refer to the addressbook, clicking on "To" will bring up the addressbook. The editor will be extremely simply. The user will be able to type test directly into the message area and click on the background of the message area to bring up a menu. On the menu, will be commands to help the user accomplish the task at hand, i.e., creating and sending an email message.

## 3.5 Filter Interface

### 3.5.1 Visual Design



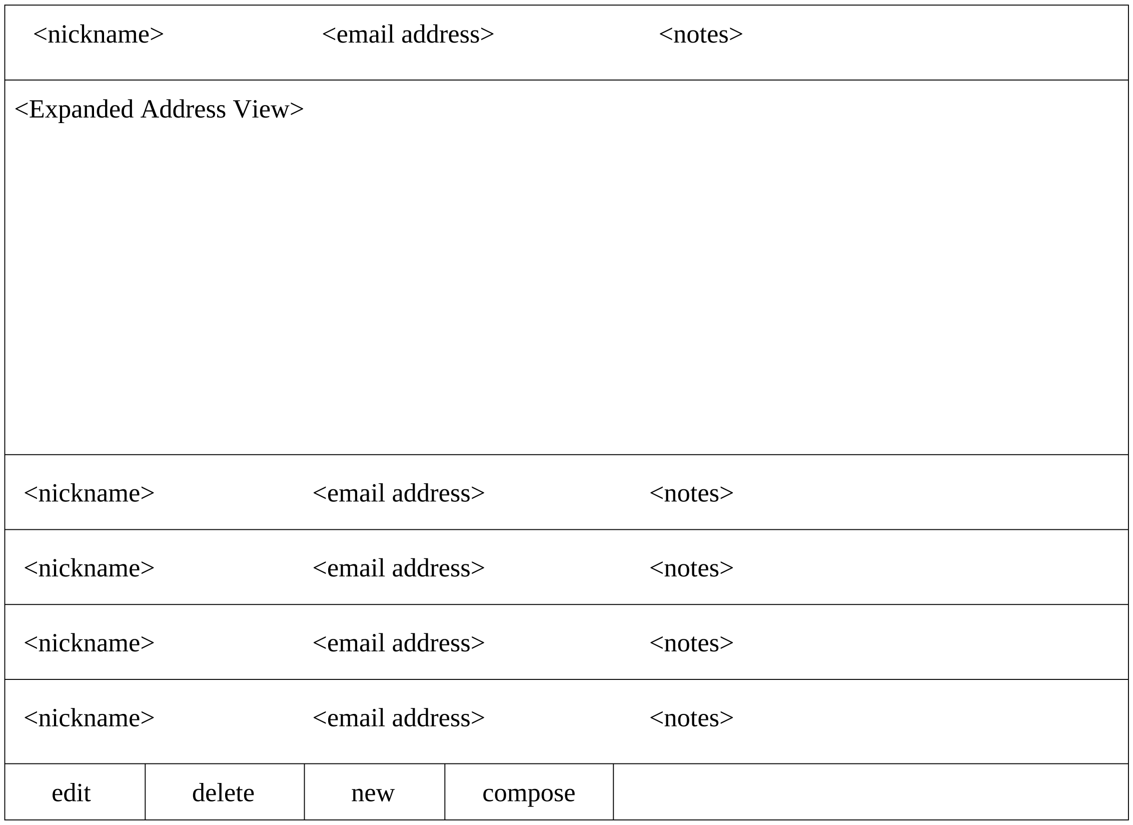
### 3.5.2 Semantic Design

The filtering portion of the program is probably the most complicated component of the user interface. The filtering interface will be much like the Photoshop layers menu. For example, in Photoshop when a user wishes to edit an image from a specific layer, the user must first click on the layer and then in another window edit the content of the layer. The same schema will predominate with Birch.

In Birch, we should view the Central mailbox as the primary work area in Photoshop. For example, by clicking on messages in the central mailbox, it will be possible for the user to load a default filter into the filter menu. The default filter will be formulated from data extracted from the selected message. For example, the recipient and sender will be "known." Then by using the options at the bottom of the filter panel, the user will be able to manipulate the filters and the contents of the central mailbox.

## 3.6 Addressbook Interface

### 3.6.1 Visual Design



### 3.6.2 Semantic Design

The addressbook component of Birch will be closely related to the filtering panel of the program, but will not be nearly as complicated. The addressbook will provide a simple interface which permits the user to view, select, delete, and edit entries in the addressbook. Visually, it will be very similar to the filter panel with its editing options located at the bottom of the screen, thereby establishing some consistency in the user interface.

## 3.7 Help Interface

Throughout the program, there will be context-sensitive labels. Moreover, it will be a rule that the label on each component should never simply repeat what it written on the component. For example, the save button should not pop up a context-sensitive label that repeats "save." Although context-sensitive help can sometimes be aggravating, Birch's implementation of context-sensitive help should never be overwhelming or verbose. On the contrary, it should be simple and concise. More specifically, the help will not take the form of large help bubbles popping up after every mouse movement, nor will there be any agents.

### 3.7.1 Error Messages and Alerts

The error messages and alerts in the program will mostly be reported by pop-up error dialogs. The messages of the errors should also be concise, but of course helpful. No error should ever be reported as something on par with "System Error 12:78903."

### 3.7.2 Documentation

- There will be an extensive man page on Birch. It will outline Birch's functionality and highlight its filtering capabilities.
- Birch will have Web documentation which will exist on a Web server, but the option to download the Web documentation should also be open. The Web documention will include a description of Birch's functionality, a Birch tutorial, and a FAQ.

## 4 Runtime Environment

Birch will require the following environment to run:

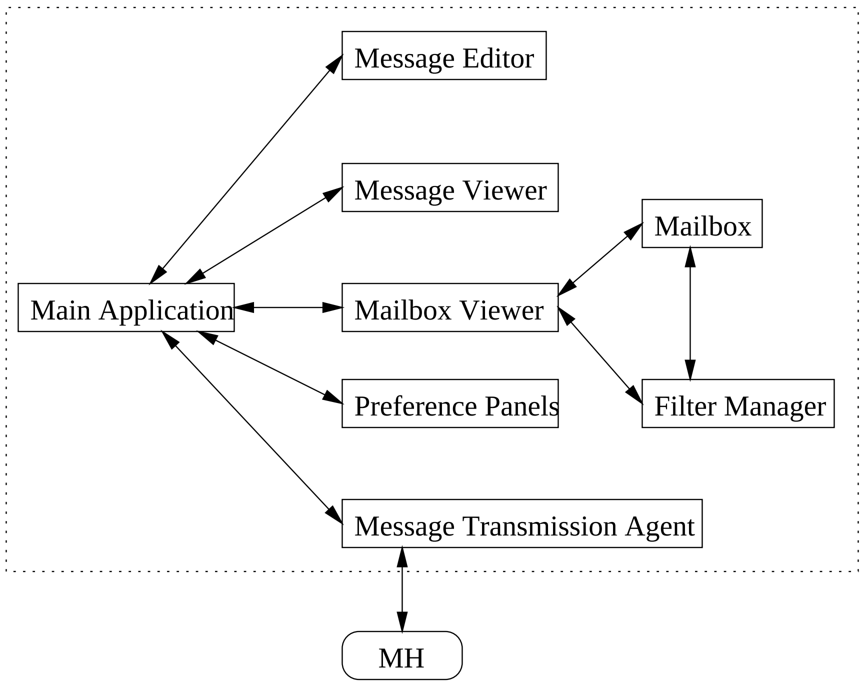
- A Unix system running MH. Developement will be done on the Sun Ultra 2 workstations. Compatible Unix environments should be able to recompile and use the Birch sources without a major porting effort.
- Motif
- Exact memory and storage requirements will vary, depending on hardware and operating system. Memory required by Birch itself should be around 3-4 Megabytes of RAM in addition to the RAM required by Motif, increasing with mailbox size. Storage requirement is proporational to the mailbox size, plus the storage required for the configuration files.
- Whatever networking setting required for the transmission of messages

## 5 Implementation Requirements

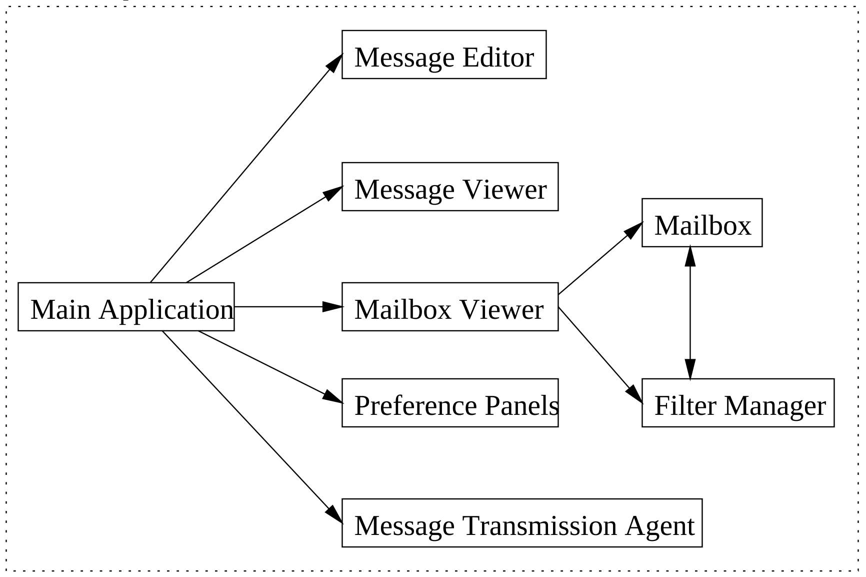
The following environment will be used in the development of Birch. Users with a compatible environment should be able to recompile Birch with minimal effort.

- The latest Sun C++ compiler at the time of this writing
- Motif
- SGI version of STL
- CVS for source code control

## 6 Component Breakdown



Communication flow diagram. Arrows indicate which components talks to each other.



Dependency diagram. Arrows indicate which components depend upon each other.

## 7 Development Schedule

2/24: Top-level Design Proposals
3/10: Final Top-level Design
3/15: Interface Proposals
3/19: Interface Comments
3/24: Final Interface Definition
4/7: Detailed Designs
4/19: Initial System Integration
5/3: Full System Implementation
5/15 9am: System Submission