Working Paper Series
ISSN 1170-487X

Notes:
An Experiment in CSCW

**by: Mark Apperley, Simon Gianoutsos,
John Grundy, Gordon Paynter,
Steve Reeves, John Venable**

Working Paper 96/9

April 1996

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Notes: An Experiment in CSCW

Mark Apperley, Simon Gianoutsos, John Grundy
Gordon Paynter, Steve Reeves
John Venable

Department of Computer Science
University of Waikato
HAMILTON
New Zealand

Abstract

Computer Supported Co-operative Work (CSCW) systems are complex, yet no computer-based tools of any sophistication exist to support their development. Since several people often need to work together on the same project simultaneously, the computer system often proves to be a bottleneck. CSCW tools are a means of allowing several users to work towards their goal. Systems development is essentially a team process, yet support for CSCW on these systems in its infancy.

The aim of this report is to record the building and experimental use of two prototype systems. These systems were developed for two main reasons: to explore the usefulness of two different environments for building CSCW systems; to experiment with some ideas that had arisen from group discussions on particular artefacts—notes—and see both how they could be implemented and how useful they were in practice.

1 Introduction

This report has been produced as one of the outputs of the FoRST funded project "Improved Computer Supported Collaborative Work Systems" conducted within the Department of Computer Science at the University of Waikato.

The initial idea of notes (which we seem to have come back to, see section 5) was to provide at least the sort of facility of use that yellow sticky notes provide in the world of paper. So, notes could be left anywhere on any document, they could be added to with other notes and they could be specially treated by sending them to named people. See section 2.3.1 for more on this.

The report has a simple structure: each of the two environments and systems produced within them is described (sections two and three) and then they are compared and contrasted in section four. A simpler system, designed in the light of experience with the earlier work, is presented in section five.

The example chosen as a target application to which the note was applied was an editor for entity-relationship (ER) diagrams.

2 A Groupkit-based editor

This section contains an overview of the Multi-User ER Diagram Editor (called "Muer") . Muer was written using GroupKit 3.0 [Roseman et al. 1996], Tcl 7.3 [Ousterhout 1994], Tk 3.6 [Ousterhout 1994], and Tcl-DP 3.2 [Smith et al. 1993].

2.1 Overview

The rest of this section contains an overview of the functionality of Muer. In section 2.2 an outline of the Muer's diagram editing functionality is given. That is followed by an overview of the communication and collaboration facilities added to Muer in section 2.3, and a description, justification, and evaluation of each of the "groupware" features in section 2.4.

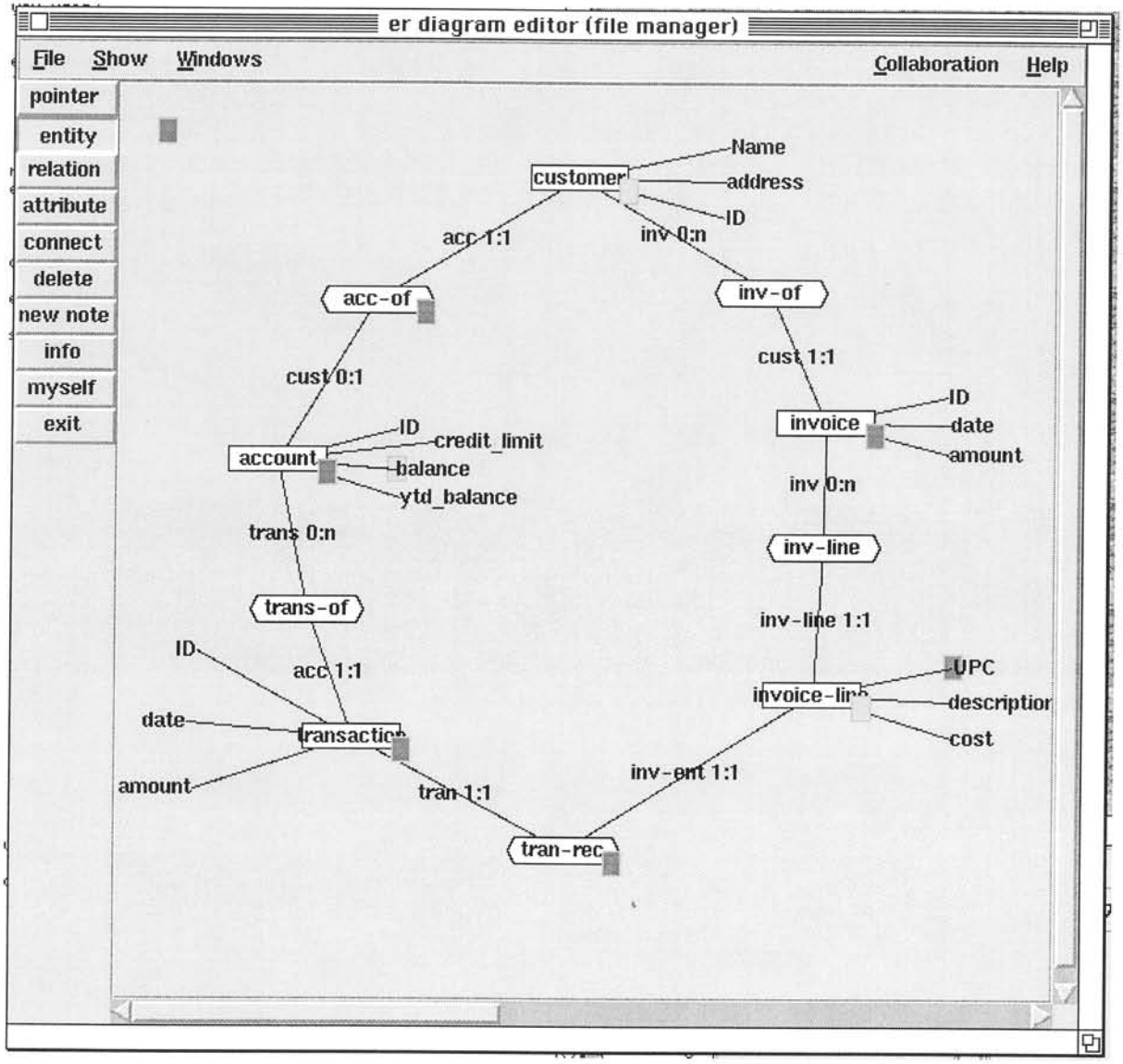

Figure 2.1: The main screen

2.2 ER diagram editing features

Muer was written as a single user ER diagram editor, to which group awareness was subsequently added. The original interface was loosely based on the MViews ER prototype [Venable and Grundy 1995], the main differences arising in the differing uses of icons. Like MViews ER, the Muer editor interface consists of a main screen with a set of buttons on the left hand side and menu bars available at the top. The "pointer", "entity", "relationship", "connector", "attribute", and "delete" buttons shown in figure 2.1 are the same as tools in MViews ER.

The pointer tool is used to move and select objects in the ER diagram. Entity, relationship, connector, and attribute tools are used to create the objects of their respective types, and to rename those objects when they already exist. The delete tool, as it name suggests, is used to toggle a delete flag and remove the objects from the view.

The basic interface differs from MViews ER in three major ways. Firstly, only one diagram view, the main screen, is included in the model (though other "views" of the individual objects are described later). Secondly, there is no equivalent to the MViews ER "Hide" tool, though the "Show" menu allows the user to selectively display classes. Finally, the pointer tool's move operation in Muer uses a "Click-Motion-Click" operation to approximate the dragging in MViews ER.

2.3 Communication and collaboration overview

The step in turning Muer into a "groupware" program was to change it from a diagram editor into a collaborative diagram editor. The GroupKit libraries were used to allow several people to join a Muer conference at once. One of GroupKit's design goals was to make this kind of change simple and straightforward, and the change presented no problem.

2.3.1 Notes

Although Muer had a significant collaborative component, there were still no inherent communication tools. After much discussion, the CSCW group proposed a system of note-taking, messaging and dialogue capture to allow communication between users. The basis of the note system is a simple text form that can be attached to objects as notes and sent to people as messages. Because people are objects too, Muer combines the concepts of notes and messages, and the terms are interchangeable throughout this document (though it has been suggested that there should be an explicit differentiation, even if it is only in the mind of the user).

Each note has several properties, including a sender, subject, list of recipients, date, urgency settings, kind, and text. The kind of note defaults to one of a range of categories that were expected to occur often. These include questions, answers and comments for gaining and providing information; explanations, examples, and change notices to be used as documentation; requests for action, cancellation, counters, agreements, and completion notices for coordinating work; and the junk mail category for mail that isn't strictly necessary. In addition, the user can enter a category of their own. The test users used a couple of other categories, including "to-do item", "test", and "statement of action".

Notes are arranged into threads. When a new note is created, it becomes the first note in its own thread. When a note is replied to, the reply is considered part of the same thread as the original note. Notes can only be replies to a single other note, though any number of other objects, including threads, can be added to the list of objects that a note is attached to.

2.3.2 Performance issues

As research software, and incomplete research software at that, Muer was never intended to be 100% reliable. Experimentation with new features was the first goal. As a result, the operation of the program is not completely reliable: for example, one user crashed twice while creating the diagram, and at another point both users were asked to leave the conference and rejoin it to avoid a probable repetition. However, the nature of groupware provides a safety net in these cases: people continuously leave and rejoin conferences during normal use. Therefore, under most circumstances, the user can crash but then rejoin a conference without any major loss of work.

This is possible because there are two types of process that can connect to a Muer conference. The first person to join a conference becomes the "file manager". The file manager process is responsible for most of the file operations such as loading and saving documents, and for ensuring that all the participants in a conference have all the object information (particularly participants who are joining a conference while work is still in progress). Most of the additional functionality is transparent to the user of the file manager process, and it can be used to edit the diagram in the same way as any other process. The only visible differences are the title of the Muer window (as shown in figure 2.1) and the options under the "File" menu (described in section 2.4.1).

If you wish to use Muer for an important task, it is advisable to create a file manager process that will not be used to edit the diagram by any of the participants. Initially, this program should be used to save the diagram under a suitable name, and subsequently the other users can make the file manager save the diagram with the "Request Master Save" option in the "File" menu. The other users can use the "Save Local Copy" option to store additional copies of the diagram. Should any of the users have an error that forces them to leave the conference, they can rejoin and the file manager, and hence the diagram, will, remain intact.

Although Muer ran quite quickly on small examples (i.e. less than 30 objects) it proved to be far too slow to use practically with diagrams as large as the one built in the test, which had 93

objects. Operations that take a particularly long time are those that involve iteration through all the objects, such as creating listboxes and refreshing the screen. Muer will need to be radically revised before it can be used for diagrams of any size. File operations in Tcl-Tk take a very long time, though this does not usually interfere with the normal running of Muer. Network operations for machines that connected from other domains on the Internet took a long time to connect to the process and to make and receive updates.

The speed issues can be best addressed by rebuilding parts of Muer in C, by abandoning environments (GroupKit's shared data structures, which seem to be inefficient, though this unconfirmed) in favour of direct remote procedure calls, and by reducing the amount of network traffic caused by redundant remote procedure calls and environment updates.

2.4 Communication and collaboration features

This Section describes the individual communication and collaboration features added to Muer. Each sub-section describes a particular window and the features that are present on it. Where there are features that appear in more than one window a reference to the explanation will normally be given.

2.4.1 The Main View

Most of the functionality of the main Muer screen (shown in figure 2.1) is described in section 2. This section describes the communication and collaboration features that appear in this window.

2.4.1.1 Note Flags

Every object created can have notes attached, and each object has an associated "flag" that tells the user about any notes that are attached. This flag can have one of five states described below. If there are notes attached to a visible object then the flag is displayed in the form of a small icon, intended to represent a "sticky label", attached to the object's icon (or attached to the top left of the screen, for notes attached to the main view).

The note flags were implemented in order to let the user know that there were notes attached to each object. Although they served this purpose quite well (except for those occasions when MacX, the X terminal emulator lost its colourmap). Unfortunately, *every* user thought that the note icon was separate from the icon of the object it was attached to, and tried to get to the note information by clicking or double-clicking on the note flag. A second flaw is that the flags convey some information, but not enough. They give no idea of the number of notes attached to an object, nor how recently they have arrived. Finally, it is a little unclear what the note flag represents: the users tended to see it as representing the notes, when in fact it contained status information about the object the notes were addressed to. This distinction is described in section 2.4.4.

Flag	Meaning	Icon
0	No notes are attached	None
1	Read, non-urgent notes are attached	Yellow note, gold border
2	New, non-urgent notes are attached	Yellow note, red border
3	Read, urgent notes are attached	Violet note, gold border
4	New, urgent notes are attached	Violet note, red border

Table 2.1: Note flags and their meanings

2.4.1.2 Buttons

There are several buttons on the left of the screen that do not have an equivalent in MViews ER. These are (with the exception of the "exit" button) used to access collaborative features.

The "new note" button is used to send notes. When it is selected, clicking on any part of the diagram will bring up the dialogue in section 2.4.5, allowing the user to send a note. It was included to allow messages to be sent quickly and easily. One user suggested that a "New Message" (and possibly "Urgent New Note") button be added as well, so that messages could be sent to other users in a similar manner. This would allow messages to be sent to users more

quickly, reduce the amount of adding and removing required in the "To" field of the message, and would persuade the user that there was a difference between a note attached to an object for other users to see and a message sent directly to a person (although in the implementation no such distinction exists).

The "info" button is used to view the information about the object. Clicking on this button and then an object (or the background) brings up the dialogue described in section 2.4.4. Users complained that they were continually having to do this in order to see the notes attached to an object and desired an easier way to look straight at the notes attached to an object. (Had they but know, there is a partial solution: the Shift-Click operation will perform the same function as the "info" button in any mode without changing the mode.)

The "myself" button brings up a window identical to that in section 2.4.4 but the object described in the window is the user. This button was intended to allow the user to view messages to themselves. (Users requested that this window always be open, and updated when new messages arrive. This functionality is, in fact provided by opening this window and choosing "show new" option from the "List Type" menu.)

2.4.1.3 Menus

There are five menus on the main screen. The "File" menu contains different options for the file manager it does for the other users. The file manager has the "New" option that clears the diagram and all the data, and the "Save Master Copy" option, which saves a copy of a diagram. Other users have the option of choosing "Request Master Save" from the file menu, which is equivalent to the file master choosing "Save Master Copy". Other members of the conference may save local copies of the diagram if they want to with the "Save Local Copy" option. All the "Save" options will automatically save the diagram if it has a filename (so that users can request a master save without requiring the file master to give any input), and the menu allows users to choose "Save As" equivalents that always prompt for a filename before saving. All the users have an "Exit" option that allows them to leave the conference. However, when the file master quits, all the other members of the conference are first forced to exit.

The "Show" menu allows the user to choose whether each class of object should be displayed in the view. The user can also use this menu to suppress the display of notes, new notes, urgent notes, or new, urgent notes. These options were designed to allow the users to filter out the notes and objects that were not important, but were not used. The "Mark All Notes Read" option is included in this menu to reset the user's flags if they should be forced to leave and rejoin the conference.

The "Windows" menu allows the user to view other windows (these can loosely be described as similar to other views in MViews ER) that display the object information in other ways. These are the "View All Objects" window described in section 2.4.2, the "View all Notes" window described in section 2.4.3, and a "View Notes to Self" window that is the same as the one found by clicking on the "myself" button.

The "Collaboration" menu is a GroupKit default, and contains an option that lets the user examine low-level information about each of the participants, such as what machine they are connected from.

The "Help" menu is also a GroupKit default, with information about GroupKit. I have added a brief "About Muer" option, and an "ER Revision History" screen.

2.4.2 The "View All Objects" window

The "View All Objects" window, shown in figure 2.2, is dominated by a scrolling list containing all the (selected) objects. The object id, type, name, and creator are listed. At the top is a summary of the number of objects listed, and the number that exist in total. Double-clicking on an object in the list allows the user to view that object in a separate window. This window is intended to let the user view and access all the objects that have been created, including those that were subsequently deleted. Users felt that some way was needed to reduce the complexity of the list.

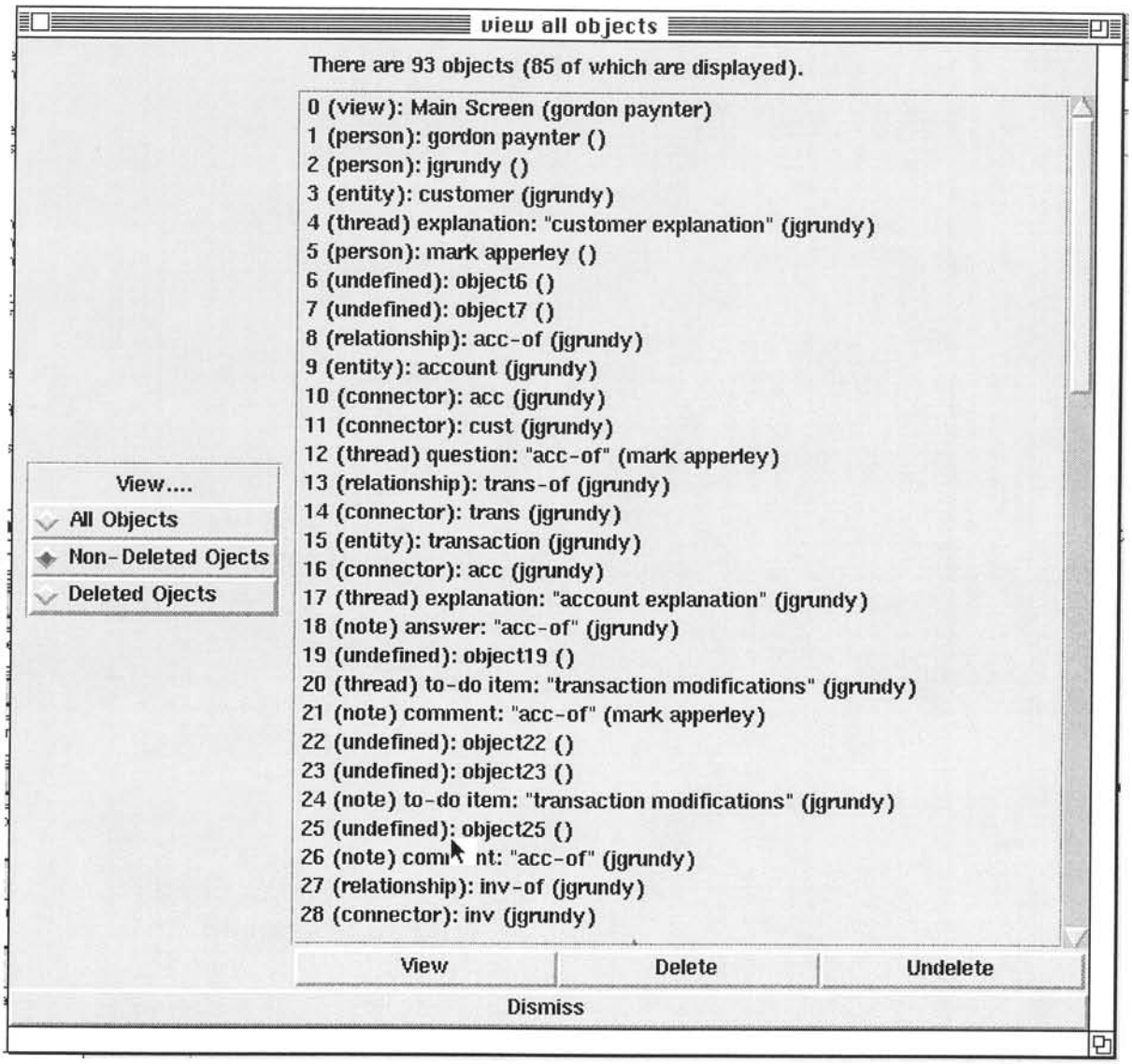

Figure 2.2: The "View All Objects" window

2.4.2.1 The "View...." radiobuttons

To the left of the object list are a set of three radiobuttons (only one button of a set of radiobuttons may be selected at any time) that filter the list slightly. The "All Objects" button puts all the objects into the main list, the "Non-Deleted Objects" button puts the non-deleted objects in a list, and the "Deleted Objects" button puts only those objects that have been deleted in the list.

2.4.2.2 Buttons

The remaining buttons are used to manipulate individual objects. The "View" button is functionally equivalent to double-clicking on an entry. When it is pressed, any objects that are selected in the listbox are displayed. (Notes are displayed in "View Note" windows, and all other objects in "Object Information" windows.)

The "Delete" and "Undelete" buttons, as their names suggest, delete objects, and recover deleted objects. When an object is deleted, a flag is toggled and no data is lost, making it easy to recover.

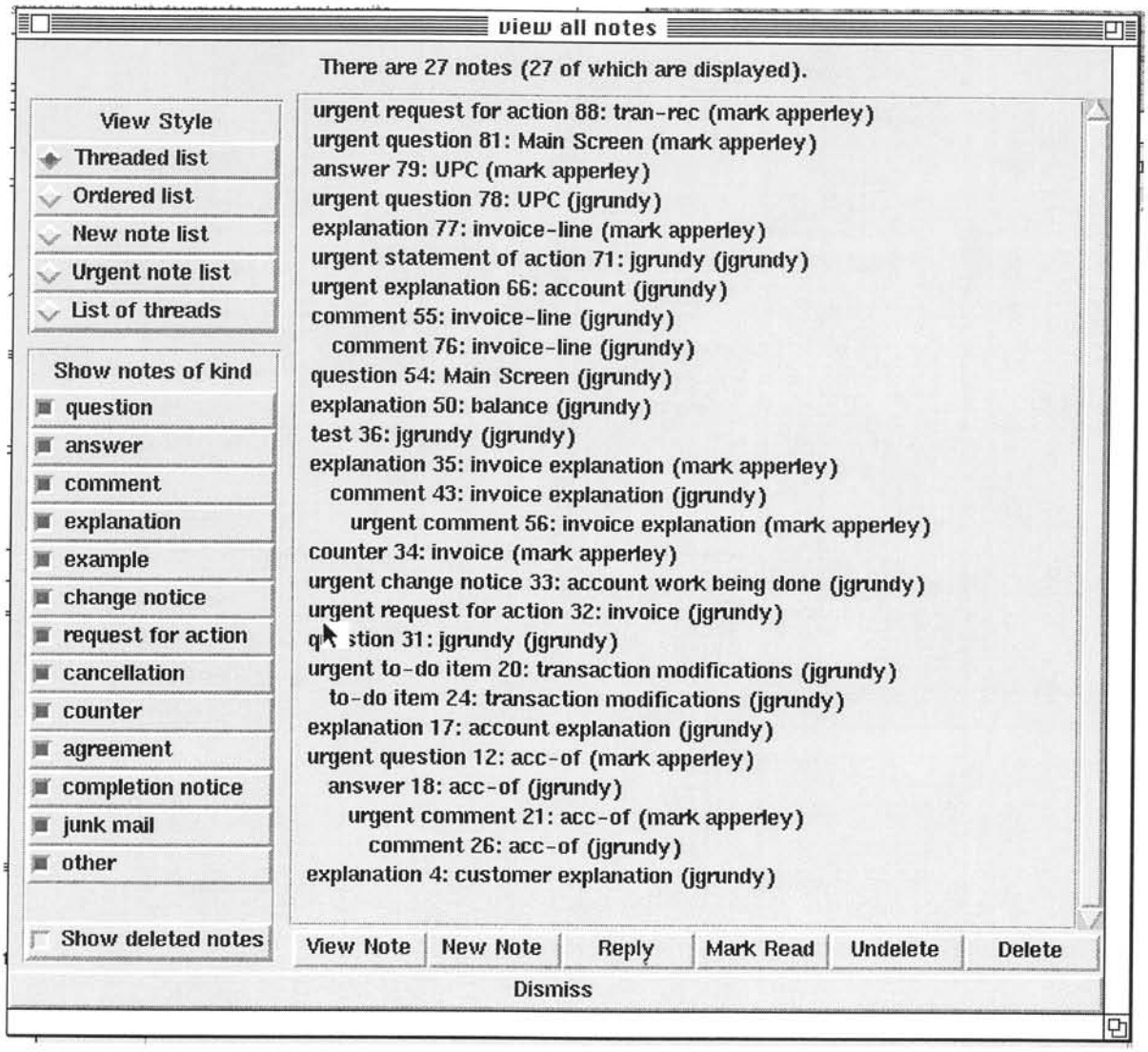

Figure 2.3: The "View All Notes" window

2.4.3 The "View All Notes" window

The "View All Notes" window is similar to the "View All Objects" window, but only notes are displayed. It contains a scrolling list with an entry for each note that includes its deletion status, urgency, type, id, subject, and sender. The purpose of this screen is to let the user search all the notes for a desired piece of information. Users felt that this list was also too complex and that better visualisation techniques were possible. They approved of the topmost string that states how many notes there are and how many are displayed.

2.4.3.1 The "View Style" radiobuttons

These radiobuttons provide several different ways of setting which notes will be included in the main list. The "Threaded list" option, shown in figure 2.3, groups the notes by thread and puts replies below and indented from their parents. This view was intended as a simple way to show note structure. The "Ordered list" option puts every note in order of id and therefore in chronological order so that the user can search for a note from around a certain date and time.

The "New note list" and "Urgent list" list new and urgent notes respectively in id order, letting the user catch up with their reading or deal with the important notes. The "List of Threads" only lists the first note in each thread, rather all the replies to that note. The users used the "Threaded List" and "New note list" options.

2.4.3.2 The "Show Notes of Kind" checkbuttons

The buttons on the lower left hand side are used to filter the list of notes based on note kind. This allows the user to, for example, only search for examples, or only look for questions and answers. The "Show deleted notes" button is the exception. When selected, it puts deleted notes into the listing.

2.4.2.3 Buttons

The "New Note" button invokes the "New Note" window described in section 2.4.5. The remaining buttons in this view act on the selected notes in the listbox. The "View Note" button will display the selected notes as in section 2.4.7, and the "Reply" button invokes the Reply To Note described in section 2.4.6. The Users indicated they used these buttons. The "Mark Read", "Undelete", and "Delete" buttons were not used in the trial.

2.4.4 The object information window

Most of the information about an object can be displayed in a window such as the one in figure 2.4. This window was originally intended as a simple listing of the notes addressed to each object, but grew into a general-purpose display and editor.

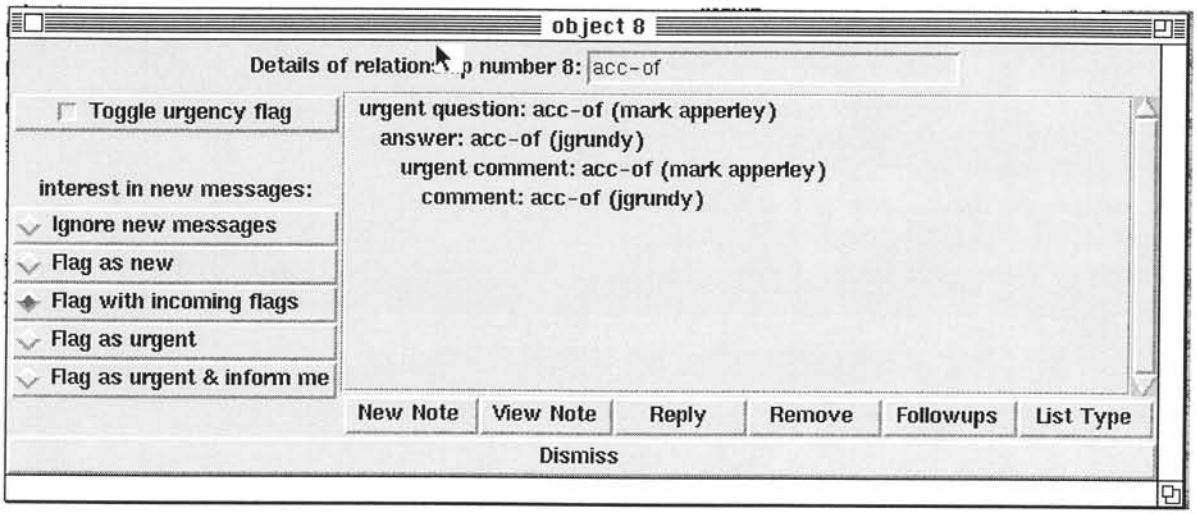

Figure 2.4: The "Object Information" window

2.4.4.1 The name field

At the top of the window is a string introducing the type of the object and displaying its name. The name is in an editable field so that the user can change it (this functionality is also available from the main screen, and this window was meant to eventually replace that as well). Users did not use this field.

2.4.4.2 The interest and urgency buttons

Because each user will be interested in various objects to different degrees, provision has been made for a user to register interest (or disinterest) in an object. This is primarily achieved through the use of the urgency flag. The flag does not report on the presence of notes, urgent or otherwise, attached to the object, it reports that something associated with that object urgently requires the user's attention.

For example, if a user has nothing to do with a certain part of the diagram, urgent notes attached in that part of the diagram are unlikely to seem urgent to that particular user. Alternatively, if a user is in the middle of important work on an object, any note, no matter how trivial the sender might think it, will seem urgent to the user. Also, a thread that was considered urgent a day ago may not be at all urgent to the user now. In any of these cases, the urgency of the notes sent to the object does not have the same effect on the urgency of the note.

It is for this reason that the urgency flag was separated from the notes and made part of the object. In the top left of the object information window is an urgency flag checkbutton. When

the object is urgent the checkbutton is active, and when it is not, the checkbutton is inactive. The user can toggle the flag by clicking on the button.

However, the flags were designed to alert the user to the presence of notes. The "Interest in new messages" radiobuttons allow the user to register a certain amount of interest in an object, and each time a new note arrives the urgency flag will be set (or otherwise) based on the urgency of the message (as set by the sender) and the interest the user has in the object (as set by the recipient). The default setting is to "Flag with incoming flags". This means that when a new message arrives, the urgency flag is set to true if the new note is urgent, but not otherwise. In other words, the urgency flag is set entirely based on the user's decision. In Figure 4, the user has the interest set to "Flag with incoming flags", and has received an urgent message in the past. However, the urgency flag is not currently set, indicating that the user turned it off manually.

Four other interest levels are available. "Ignore new messages" prevents the objects note flag from changing when new notes are attached. When a new note is attached to an object and has an interest level of "Flag as new", the object is flagged as having new messages but the urgency level does not change, even if the new note is urgent. The "Flag as urgent" treats all new notes as urgent, and flags the object's flag as such. "Flag as urgent & inform me" does the same and in addition will pop up a (non-modal) dialogue informing the user that the note has arrived, and lists the subject, sender, and kind.

Of these options, the users only ever changed the default to "Flag as urgent & Inform me".

2.4.4.3 Buttons

The buttons associated with the list box have similar functions to the buttons on the "Show all Notes" screen. "New Note" attaches a new note to this object (the new note may be attached to other objects as well through the "New Note" window in section 2.4.5). "View Note" displays all the notes highlighted on the listbox, and "Remove" deletes all of the highlighted notes. The "Reply" button can be used to reply to a single selected note, and the "followups" button is used to display the object information field of the selected note. As in the similar listings, "View Note", "New Note", and "Reply" were the most used buttons.

The "List Type" button is used to change the way the notes are displayed in the note listing. When it is pressed, a drop down menu is displayed with the option "New Notes" which lists the unread notes on the object; "Urgent Notes" which lists the urgent notes to the object; "Full Note List" which lists the notes in chronological order; "List of Threads" which lists the first note in each thread; and "Threaded List" which gives the notes grouped by thread and indented to show their place in the hierarchical structure (see figure 2.4). The "List Type" button seems to have been overlooked by the users, who did not use it but still requested some way to show a list of the new notes as they arrived, which is exactly what the "New Notes" setting does. Their overlooking the button is probably because it is inconsistently placed compared to the other, similar lists.

2.4.5 The "Create New Note" window

The "Create New Note" window, shown in figure 2.5, is generated whenever a new note is created. Much of the note information is found by the system and does not need to be entered (though it will sometimes be changed) by the user. Users reported that they used all the features in this window.

2.4.5.1 Fields

The "To" field contains a list of objects (including people) to whom this message is attached or addressed. The contents of the field can only be changed through the use of the "Add" and "Remove" buttons to the right of the field. These buttons generate drop-down menus and let the user choose which objects they wish to add to (or remove from) the "To" field from an ordered list of objects. However, the "Add" menu proved unmanageable with more than 50 objects: it takes too long to load, and is so long that many of the entries are lost off the bottom of the screen.

The "Sender" and "Date" fields cannot be altered directly by the user, though this was not evident to look at them. The formatting of the date field could be improved.

The urgency of the note, as described in the "Urgency" field, is split into three parts. This is the result of a conscious design decision to move away from the ambiguous idea of "urgency levels" and replace them with a set of specific options that the sender can use to show the receiver how urgent the note is. The options are setting the urgency flag (12 of the 28 messages were flagged), informing the most suitable recipient of the notes presence, and setting a time limit for responses (it is intended that some way of formalising the time limit be built into the program). The window for setting these options is shown in figure 2.6. These concepts are broadly based on Flores' idea of a communication coordinator.

If the user chooses to inform the most suitable recipient about the note, the program decides who is most likely to be able to help (based on the people in the "To" field and the creators of the other objects in the "To" field). The user is told who the message will be sent to in a window like that in figure 2.7, and the best recipient will be alerted with a window similar to the one in figure 2.8. The recipient sees the sender, subject, and time limit (if any), and is given the options of ignoring the message, referring it on to someone more suitable (from a drop-down menu), viewing it, or putting a decision off for a certain amount of time. The options are designed to minimise the interruption to the recipients, while making the recipient aware of the urgency of the sender's message.

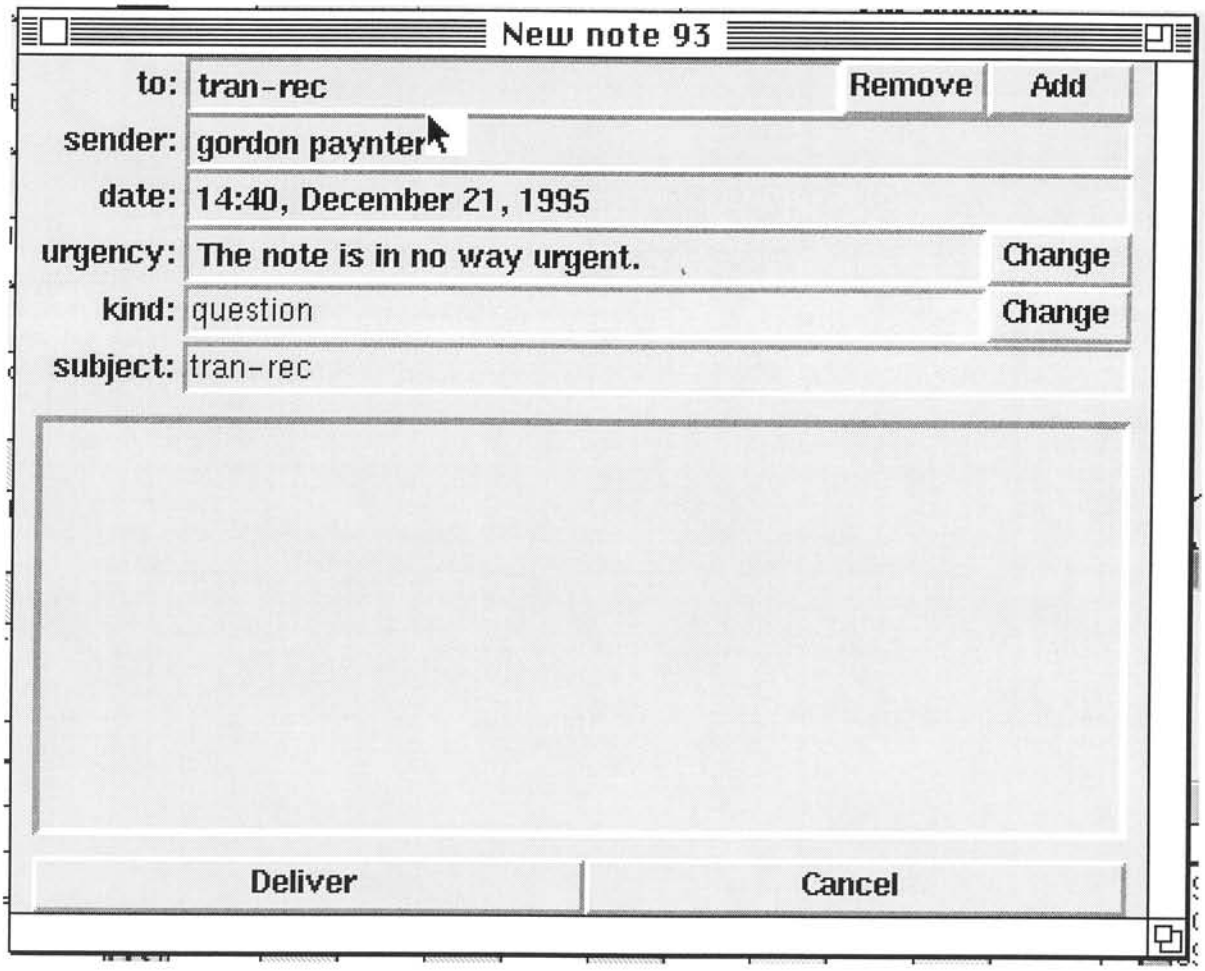

Figure 2.5: The "New Note" Window

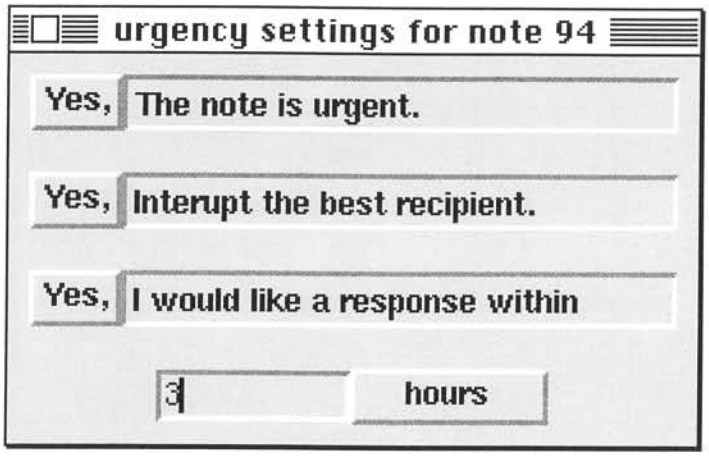

Figure 2.6: The "Urgency Settings" window

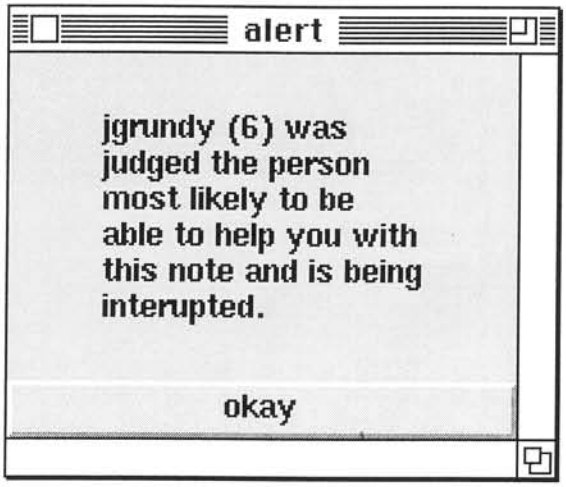

Figure 2.7: An "Alert" window

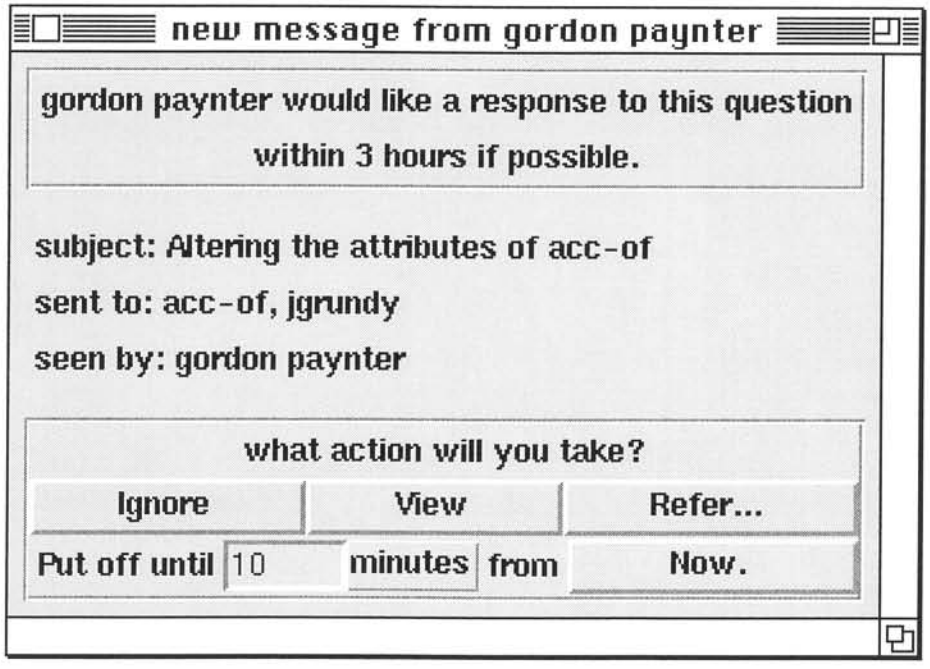

Figure 2.8: The "Best Recipient" window

The "Kind" field lets the user choose a note kind from a set of the options described in the overview. If the user is starting a thread (that is, sending a new note that is not in reply to another note) the options available are "question", "request for action", "explanation", "example", "change notice", "comment", "junk mail", and "other". The user can simply type in their own category if none of these seem suitable. If the note is a reply, then the set of replies is limited to the options listed in table 2.2. This conversation system is also derived from Flores' work.

message type	Suggested response types
question	answer, comment
answer	question, comment
comment	comment
request for action	counter, agreement, cancellation, completion notice
counter	counter, agreement, cancellation, completion notice
agreement	counter, agreement, cancellation, completion notice
cancellation	counter, agreement, cancellation, completion notice
completion notice	counter
explanation	explanation, comment
example	example, comment
change notice	comment
junk mail	junk mail
other	all options available

Table 2.2: Suggested reply kinds

The users used 28 notes. Of these, 24 were from the formal lists, and five were typed by hand. The kinds entered by hand were "test", "to-do item" (twice, though one was a reply to the other and defaulted to this value), and "statement of change" (which was a warning of changes about to be made). A breakdown of the number of messages is shown in table 2.3. Eight of the 13 available types were used, which is quite satisfactory given the size of the example. Adding "statement of proposed change" and "to-do item" to the types of note should be considered.

message type	number of uses	number of threads	number of followups
question	6	6	0
answer	2	1	1
comment	6	1	5
request for action	2	2	0
counter	1	1	0
agreement	0	0	0
cancellation	0	0	0
completion notice	0	0	0
explanation	6	6	0
change notice	1	1	0
example	0	0	0
junk mail	0	0	0
other	4	3	1

Table 2.3: Use of message types

The "Subject" field is a simple text field that can be edited by hand. It defaults to the name of the first object on the "To" list, and almost always requires changes. Extensions to the available bindings to this field (and many of the others) are necessary. Particularly useful would be arrow keys for movement and sensitivity to mouse selections.

2.4.5.2 Text

The main part of the window is the text editing area. This area requires many further bindings (like the Subject field), and a scrollbar.

2.4.5.3 Buttons

The "Deliver" button attaches the note to the appropriate places and performs the requisite notifications. The note can be cancelled with the "Cancel" button.

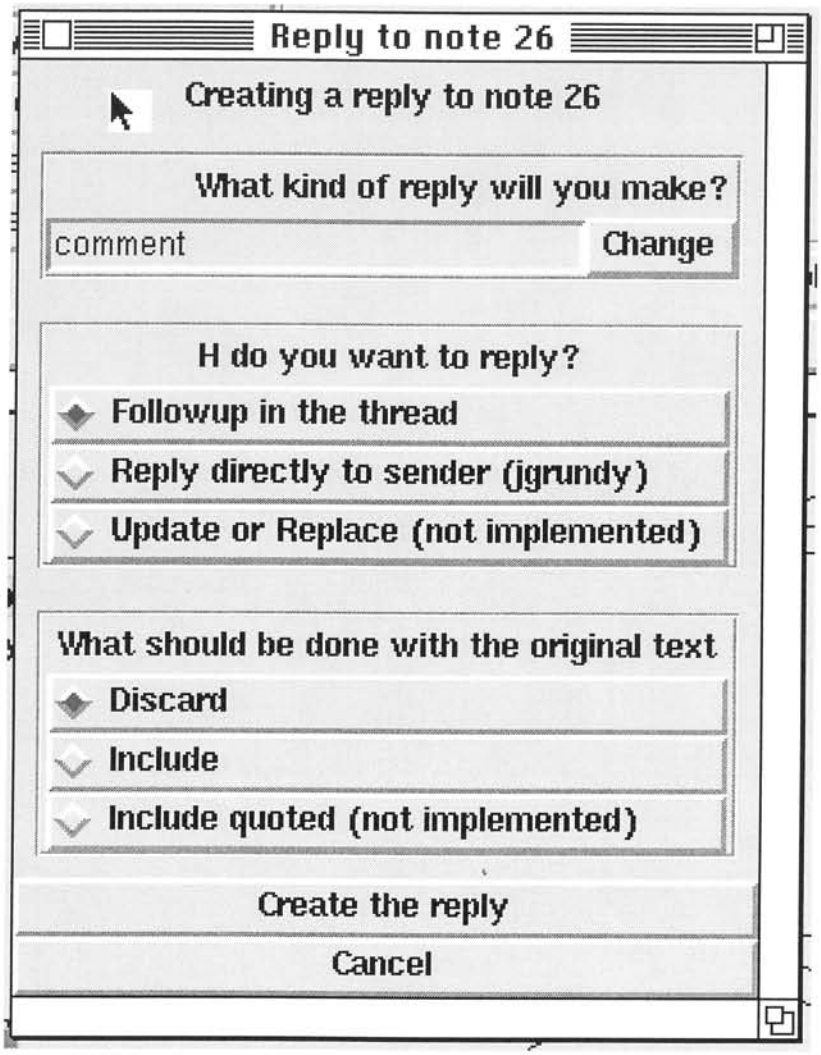

Figure 2.9: The "Reply To Note" window

2.4.6 The "Reply To Note" Window

Whenever the user decides to reply to a note and presses a "Reply" button , the "Reply To Note" window is generated. After leaving this window, the user generally goes straight into the "New Note" window and actually creates the reply. The "Reply To Note" is simply used by the user to set options and defaults for the reply. An example is given in figure 2.9.

The users said that replies should have the same urgency as their parents, though this was not implemented.

2.4.6.1 The kind field

The kind field works identically to the kind field on the "New Note" screen. It defaults to an appropriate kind that can be changed by the user. This value is the default for the "New Note" window when that window is invoked.

2.4.6.2 The how radiobuttons

The method of reply is chosen with the middle set of radiobuttons. The users' options are to perform a standard followup (that will be attached to all the objects on the "To" list of the original note), reply to the sender (and not to any other objects or people on the "To" list), or to "Replace" the current note. The last option is for use with notes such as explanations that are

intended for use as documentation. When an object is changed and the user changes the documentation with this option , the old documentation is deleted when the new documentation is created. (The deleted documentation is still viewable from the "View All Notes" window.) This final option has not been implemented.

2.4.6.3 The original text buttons

The bottom set of buttons are used to set the default text in the "New Note" window. The options are to discard the text, include the text as it appears for later editing, and to include the text but quote it (as is done in email and Usenet postings). The third option has not been included, to the extreme annoyance of the users.

2.4.6.4 Buttons

The "Create the reply" button is used to pass the defaults from this window on to the "New Note" window. The "Cancel" button cancels the reply before this action takes place.

2.4.7 The "View Note" Window

The "View Note" window, as the name suggests, is used to view notes that have already been sent. These notes are not static: some of their settings can be altered even after the user has sent them. An example is given in figure 2.10.

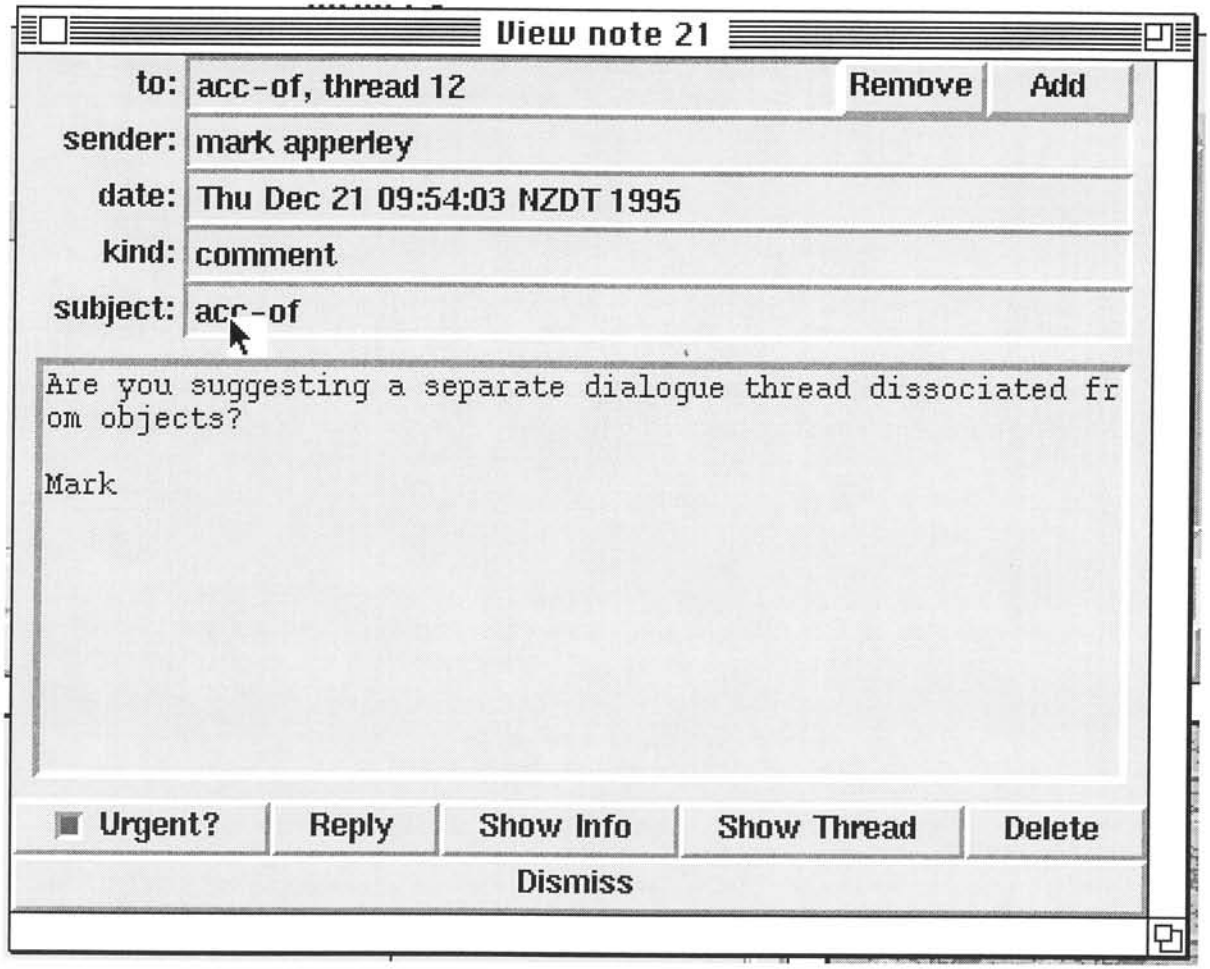

Figure 2.10: The "View Note" window

2.4.7.1 Fields

The "To", "Sender", "Date", "Kind", "Subject", and "Text" fields have basically the same layout and data that appear in the "New Note" window. The "Urgency" field is replaced by the "Urgent?" button at the bottom left of the window. However, except for the "To" field, their contents cannot be changed. The To field can be changed in the same manner as the "To" field in the "New Note" menu. This allows the list of objects to which the note is attached to be edited after the note has been sent.

2.4.7.2 Buttons

The "Urgent?" checkbutton is used to show and toggle the note's urgency flag. This flag affects the description of the note when it is listed in listboxes such as the "Show All Notes" window: it does not affect the urgency flags of the objects that the note is attached to.

The "Reply" and "Delete" buttons have the same function that they have in other places: one creates a reply and the other deletes the note. The "Show Info" button displays the "Object Information" screen for the note, while the "Show Thread" button is intended to show a dialogue capture of the thread that the note is a part of. It is not implemented.

3 The MV-based system

This section introduces the MV-based system (called "MV Notes") to the reader, and gives some of the reasons behind prototype implementation of a system to allow notes to be attached to items of interest.

This section contains issues involving the design of MV Notes and rationale for doing things in certain ways. Benefits and advantages of each feature will be discussed.

This tool was designed as an extension to an existing ER CASE tool, MViews ER. MViews ER was built using the MViews framework. MV Notes was kept as separate from MViews ER as was feasibly possible, thereby enabling it to be used for other MViews applications.

3.1 Snart

The Snart programming language was used as the development platform. Snart integrates the programming paradigms of object-orientation, logic and constraints. It is built as an extension of LPA MacProlog32 [LPA 1994] and has been used to experiment with programming language design and with the architecture of programming environments. Snart was originally designed to be easy to implement and extend [Grundy 1993].

Snart began as an OO extension of Prolog. To that base has been added: classifiers (based on Kea), object spaces and persistence, and constraints [Mugridge 1994].

In general terms the object-oriented features of Snart are similar to other class-based OO languages, such as Eiffel and C++ [Mugridge 1994].

3.2 MViews

MViews [Grundy 1993] is a model and framework for constructing Integrated Software Development Environments (ISDEs) , which represent the abstract syntax, semantic attribute values and multiple views of a software system as graphs. Graph components are modified by operations to construct a program, and software developers view and manipulate the view graphs in concrete textual and graphical forms.

Consistency management between updated view components is supported by a change propagation mechanism. MViews graphs are dependency graphs: descriptions of changes to graph components (called *update records*) are broadcast to related (dependent) graph components, which then respond to these update records and update their own state to maintain consistency. This mechanism provides a way of keeping textual and graphical views of software development consistent.

Storage of update records by graph components supports a generic undo/redo facility, modification histories, version control, external tool integration, and collaborative software development. Extensibility is supported by components and views can be added or modified without affecting existing structures (via the dependency graph mechanism). New environments are constructed by specialising an object-oriented framework, which provides a consistent user interface for views. MViews is currently implemented using the Snart language, an object-oriented Prolog, however it is being ported to C++.

3.3 MViews ER

MViews ER [Venable and Grundy 1995] is a single-user environment that integrates ER (Entity-Relationship) and RDS (Relational Database Schema) specification to provide graphical ER modelling views and complementary textual RDS views, with consistency management between the two. Figure 3.1 shows an example of MViews ER, which takes this approach to database model specification.

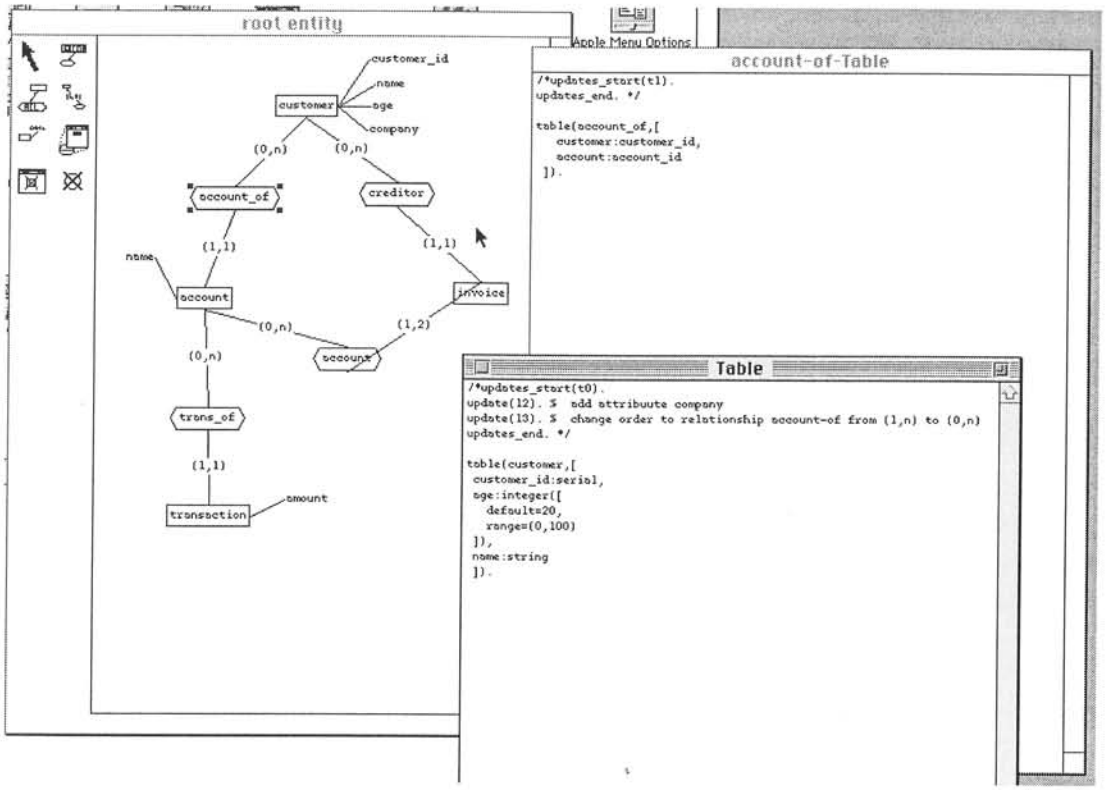

Figure 3.1 : MViews ER environment

MViews ER supports graphical ER diagram views with diagrams constructed using tools, dialogues and menus. Textual RDS views contain a table definition including table fields, field types, and zero or more field values used to specify various attributes for fields. RDS views are parsed to update table information. The graphical ER views provide a high-level specification system with details about RDS requirements ignored. Textual RDS views can be generated from ER data and provide extra information about field types, defaults, ranges and so on.

3.4 MV Notes Architecture

A high level inheritance diagram showing how MViews, MViews ER and MV Notes relate is shown in figure 3.2:

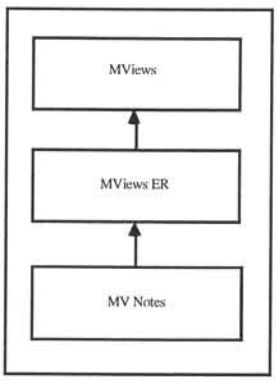

Figure 3.2 : MV Notes High Level Inheritance Diagram

MV Notes was kept as separate from MViews ER as was feasibly possible, thereby enabling it to be used for other MViews applications if desired. Ideally MV Notes would be integrated into MViews, thereby ensuring greater transparency.

3.5 MV Notes

MV Notes is a textual and graphical means of attaching notes to MViews objects. Notes can have various attributes associated with them (see next paragraph), and these attributes may effect the graphical appearance of the note. For example, an unread note is graphically displayed with its name in red. Figure 3.3 shows an example of MV Notes applied to MViews ER.

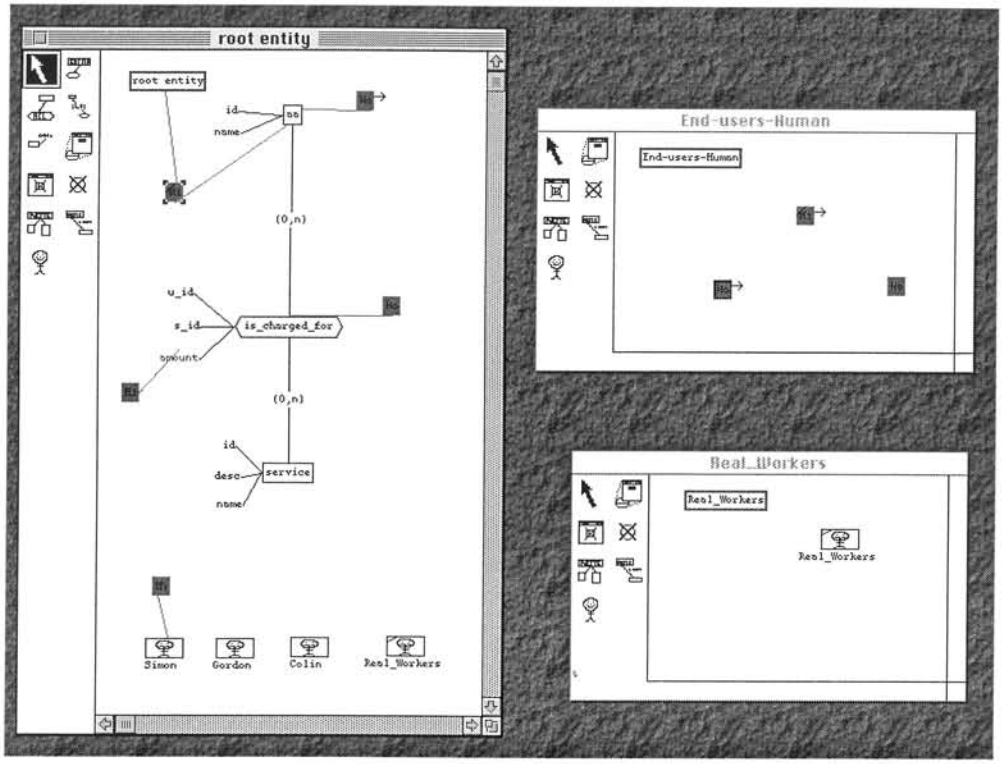

Figure 3.3 : MV Notes applied to MViews ER

The toolbar on the left hand side of each graphical window contains the tools that can be used on the graphical objects. A menu option is provided that contains access to other information about MV Notes. This menu would definitely not be considered a masterpiece by HCI experts. All aspects of this menu are discussed in further sections.

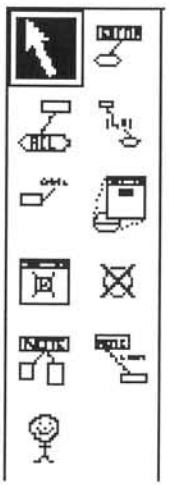

Figure 3.4 : MV Notes Tool Bar

The "Pointer" icon in figure 3.4 is used to select and move objects. Double clicking on an MV Note Icon with the "Pointer" will edit the MV Note, and likewise within MV Note User Icon. The "Entity tool", "Relationship tool" , "Entity/Relationship tool" and "Attribute tool" are part of MViews ER and have no effect on MV Notes. "Add View" is used to create a view of an object (graphical or textual). "Hide" is used to remove a graphical object from the display. "Delete" is used to delete an object. The "Note tool" is used to create and edit MV Notes. Once this tool is selected , clicking on an existing MV Note edits it, and clicking on an empty location creates a new MV Note. The "Note Link tool" is used to link MV Notes and MV Note Users to MViews objects. The "Note User Icon" is used to create and edit users. The functionality is similar to the "Note tool".

Visual representations of notes can be found in section 3.5.5. Attributes of a note include:

Id	A user defined id. By default this is the MViews Object id.
Name	A user defined name.
Purpose	A short text string to overview the note contents.
Text	The text the note contains.
Owner	The user who is responsible for this note. By default this is the creator of the note.
Kind	A way of classifying notes into types. The only difference the kind currently makes is in the graphical appearance of the note. Notes kinds are: • Explanation (default) • Message • Question • Example
Priority	A scale used to signify the importance of a note. This is set by the creator of the note. Note priorities are: • Lowest • Low • Normal (default) • High • Highest Section 3.6.1 explains the impact of priority settings.
Private	A flag to indicate whether the note is readable by other users. This makes the note only visible to the owner.
Creation Date	The date the note was created.
Creation Time	The time the note was created.
Last Update Date	The date the note was last updated.
Last Update Time	The time the note was last updated.
Default Access Rights	The default privileges that a user has to this note (see section 3.5.2).

MV Notes also have the ability to have multiple views. An MV Note view may contain MV Notes, MV Note Users or views of these components. It is intended that MV Note Views are used to reply to a specific note. This helps to organise the context of notes.

MV Notes can also be attached to the links between objects (notably other Notes). This could be useful to explain the relationship between notes.

3.5.1 Registration of Interest

Different users have different levels of interest in objects. MV Notes allows users to register interest in any MViews object. Whenever this object is modified, the user would be notified appropriately (see section 3.6.1). Four different levels of registration are available:

- Very Interested
- Slightly Interested
- Reasonably Interested
- NOT Interested

A comment field is also provided for the user to enter a reason why they have registered this level of interest or disinterest.

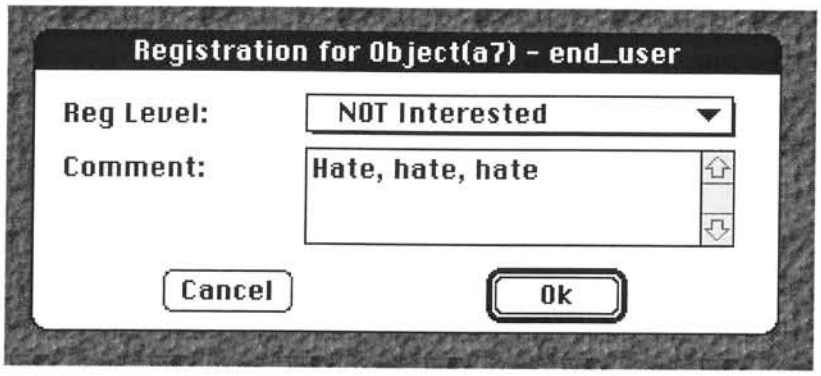

Figure 3.5 : Registration of Interest

Registration of disinterest can also be registered in objects. By registering disinterest, the user will not be notified about anything involving the object concerned. This can help to filter out annoying notification of change messages.

3.5.2 Access Rights

Whenever an MV Note is created it is given default access rights. These rights, definable by the user, state what other users can and cannot do to the note.

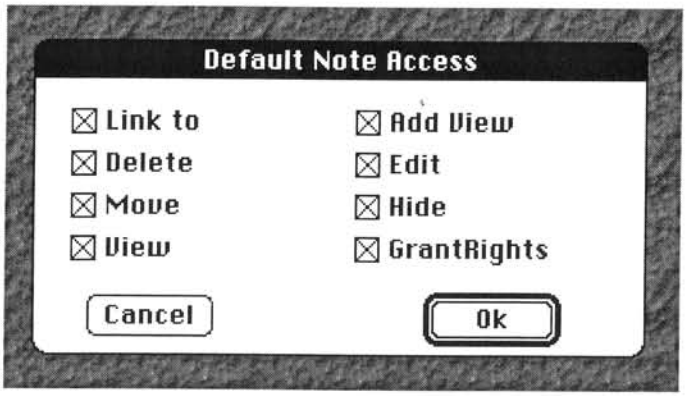

Figure 3.6 : Default Note Access

The rights are assigned to tasks that can be performed on a particular object by an individual user. By providing security constraints at this level, a high degree of flexibility is available for restricting access. Access rights available are:

Link to	The ability for the user to link this note to another object, or to link another object to this note.
Delete	The ability to delete the note.
Move	The ability to change the physical location of this note on the screen.
View	The ability to view the contents of the note (this option is not currently used. 'Edit' is used instead).
Add View	The ability to add a view to the current note.
Edit	The ability to edit the contents of the note.
Hide	The ability to hide the note.
Grant Rights	The ability to grant rights to other users for the note.

The default access rights can be overridden for a particular user by any user who has grant rights for the note concerned. Overriding can be used to add and/or remove access rights. For example, Gordon may have all rights on Note X, yet Colin may only be allowed to perform a subset of these rights. Figures 3.7 and 3.8 show how these access rights can be modified.

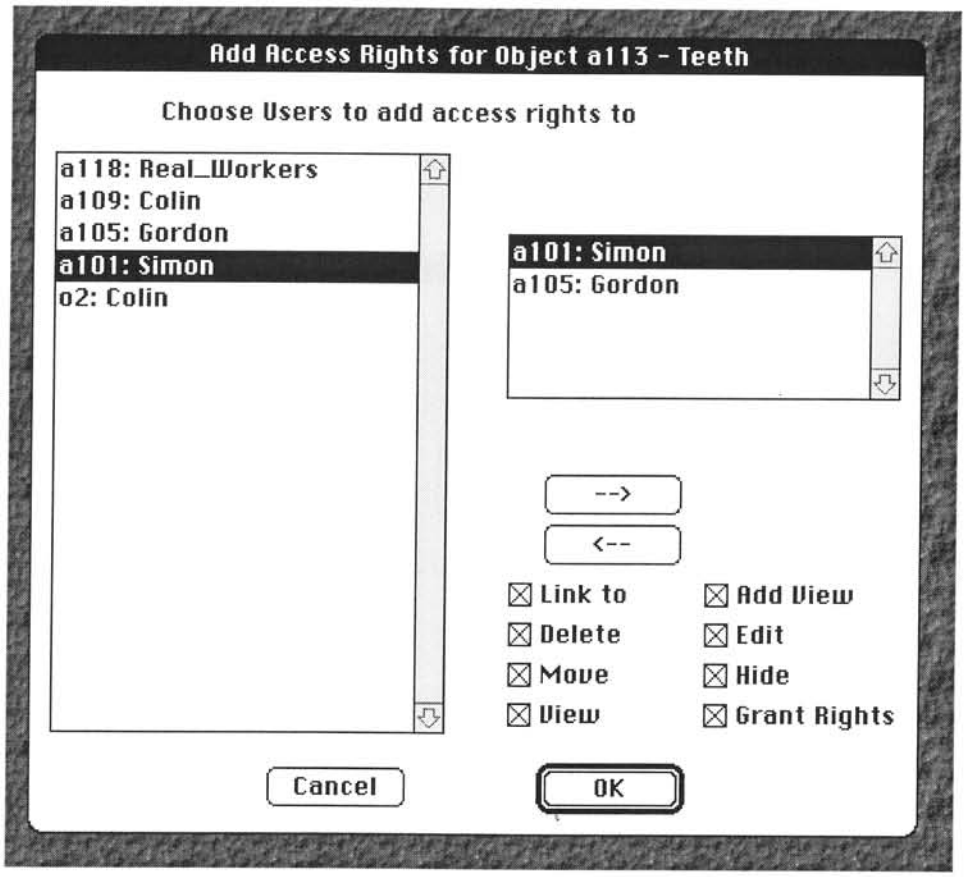

Figure 3.7 : Add Access Rights

In figure 3.7 the list on the left hand side is the full list of users and user groups. To give a user rights, the user should be highlighted in the left list. Pressing the arrow button pointing to the right will copy the selected user to the list on the right hand side. Likewise using the arrow button pointing to the left will remove the user from the list of users to add access rights to. The list on the right hand side is the list of users to add access rights to. The access rights to grant each of the users can be selected by choosing the relevant checkboxes. Clicking the Ok button will add the access rights chosen to the users in the right hand list. If the user already has access rights these will be appended to, and not overwritten.

Figure 3.8 works in a similar way to figure 3.7, except that access rights are removed from users in the list on the right hand side.

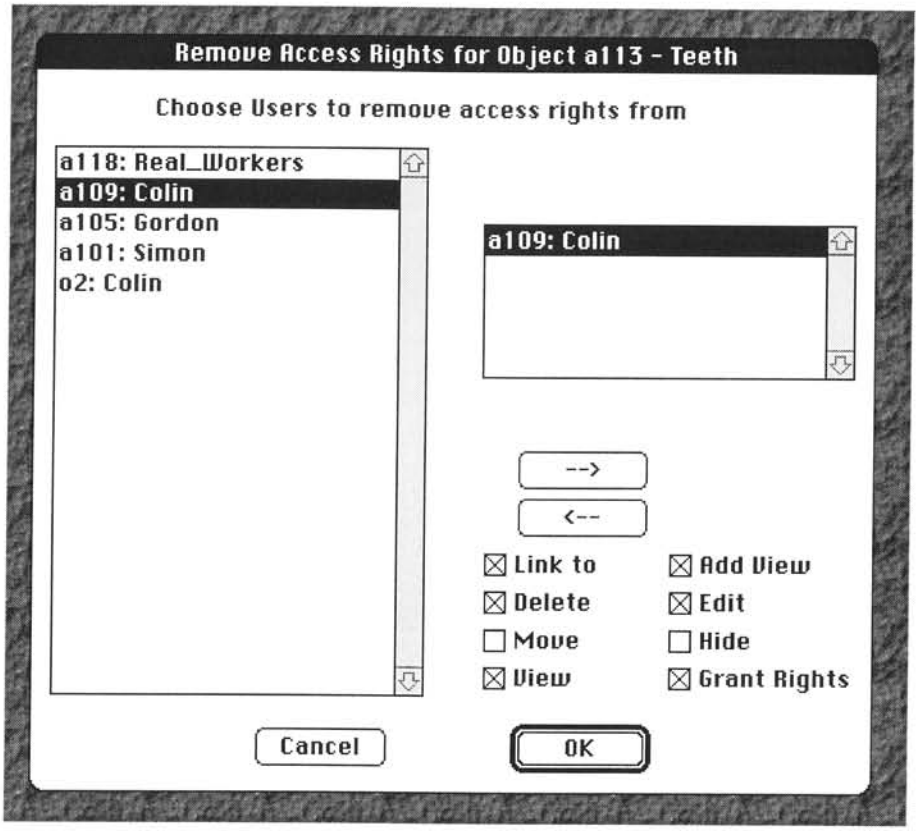

Figure 3.8 : Remove Access Rights

3.5.3 Filtering

A basic filtering system has been set up to hide a subset of notes from the graphical display. The user can choose which types of notes they are interested in and only these notes will be graphically displayed on the screen. This works both for existing notes graphically displayed, for new notes, and for modified notes. For example, if filtering states that 'Explanations' are to be filtered out, and the kind of a note is changed to 'Explanation', then it will be removed from the graphical display.

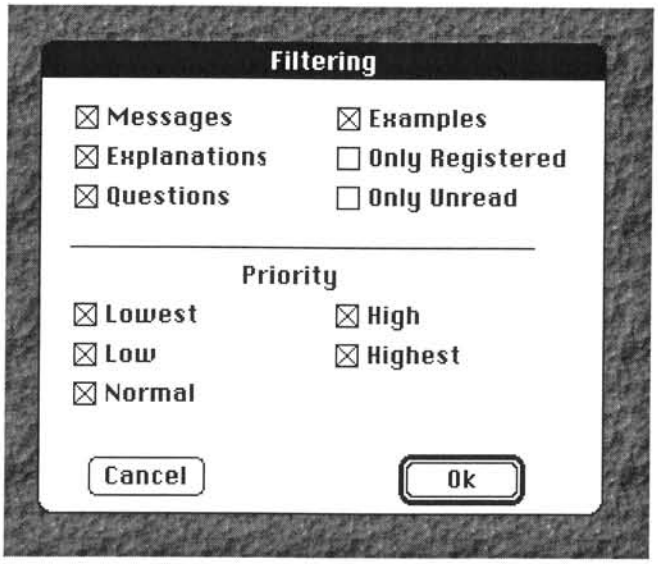

Figure 3.9 : Filtering

Filtering is currently only applied to graphical views, although it should not be difficult to apply filtering to queries.

3.5.4 Update History

The MViews update history can be used to store a list of updates made to any MViews object (including views). This has been extended to provide information about who made the change and the time the change was made. Figures 3.10 and 3.11 show the update history for the MV Note named Currency.

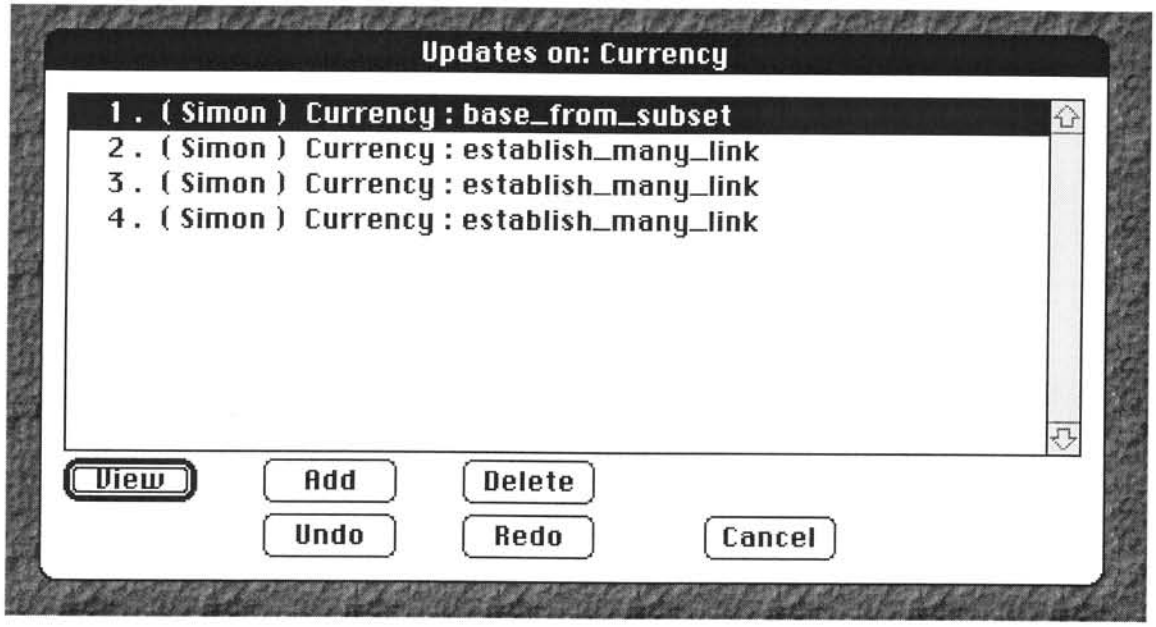

Figure 3.10 : Update History

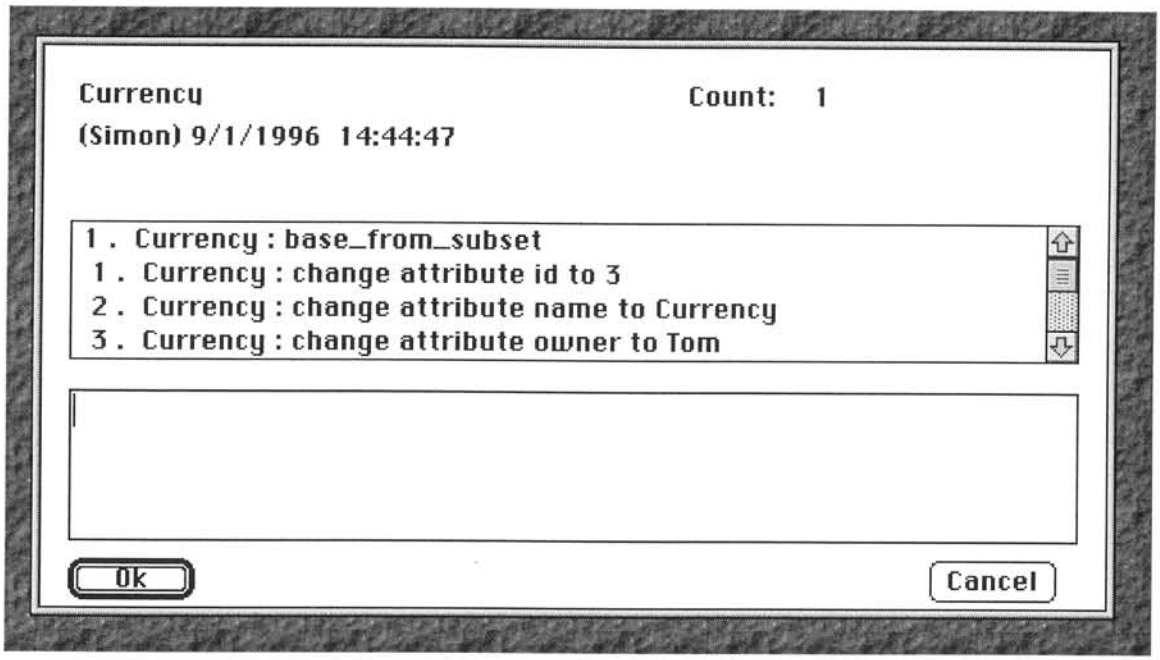

Figure 3.11 : Update History Detailed Information

3.5.5 Visual Representations

An example of an MV Note is given in figure 3.12. These settings plays a large part in determining the visual representation of the MV Note Icon. Double clicking on an MV Note Icon edits the note information. Clicking on the "-> To do" button of figure 3.12 adds the current note to the "To do List" (see section 3.6.3).

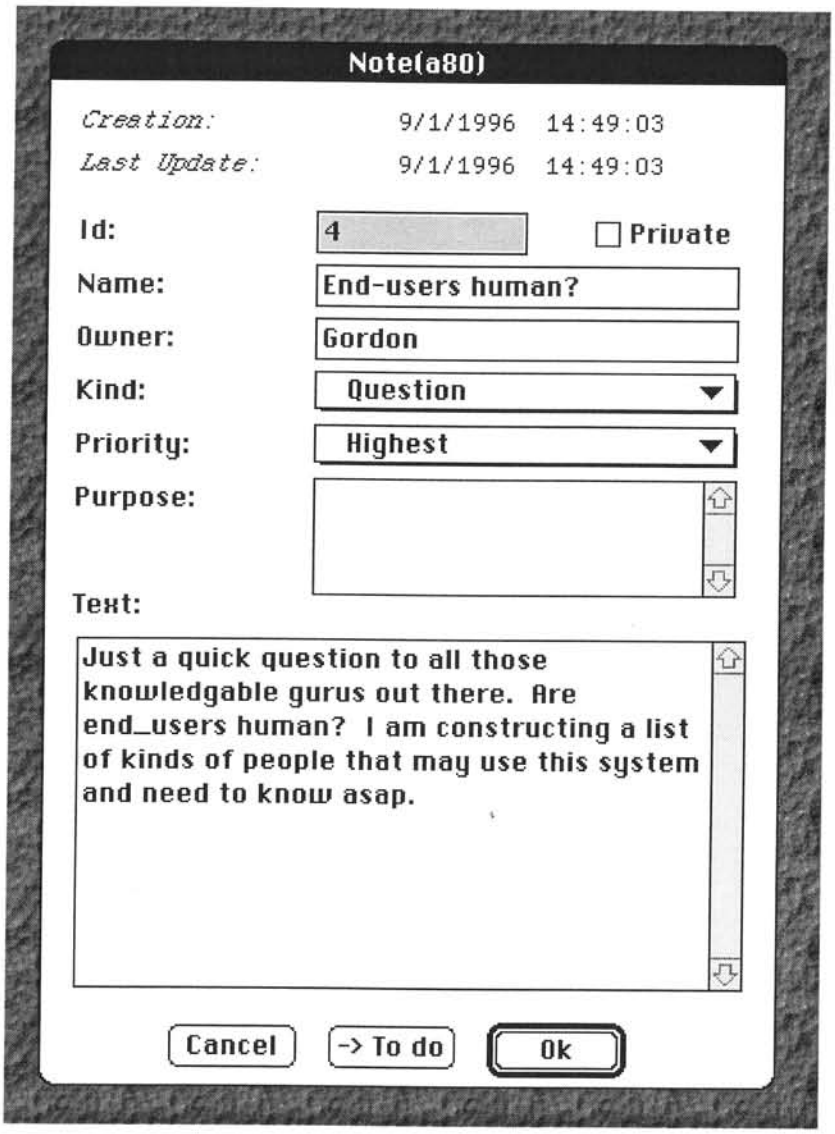

Figure 3.12 : MV Note

The kind of a note (i.e. Explanation, Message, Question, Example) is used to determine the colour of the note icon on the screen. These colours are fairly gaudy since there were very few colours that could be displayed properly by LPA MacProlog32. The rgb colour scale did not work as documented.

Figure 3.13 : MV Note Icon

The text of an unread note is graphically displayed with the border of its icon in red. This colour will automatically be changed to black once the note has been read. A menu option is also available for the user to toggle a note between read and unread.

The first two letters of the priority of a note is given in the bottom half of the note icon. This means that "Lowest" and "Low" will both be represented as "Lo", and likewise for "High" and "Highest". It was deemed that this was significant by users.

If the note contains any other views (whether textual of graphical) a diagonal line is displayed in the top left corner of the icon.

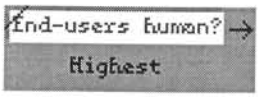

Figure 3.14 : Original MV Note Icon

In the first prototype a significantly larger note icon was used (see figure 3.14). This note icon was too obtrusive, and thereby refined to the icon in figure 3.13. The original note icon contained the name of the note in the top half of its icon. An unread note was graphically displayed with the name on its icon in red opposed to the change in border colour. The priority of a note was given in fill in the bottom half of the note icon. This extra information was not deemed necessary, therefore removed from the subsequent icon.

An example of the clumsiness of the use of the old icons is displayed in figure 3.15. This figure contains exactly the same information as figure 3.3.

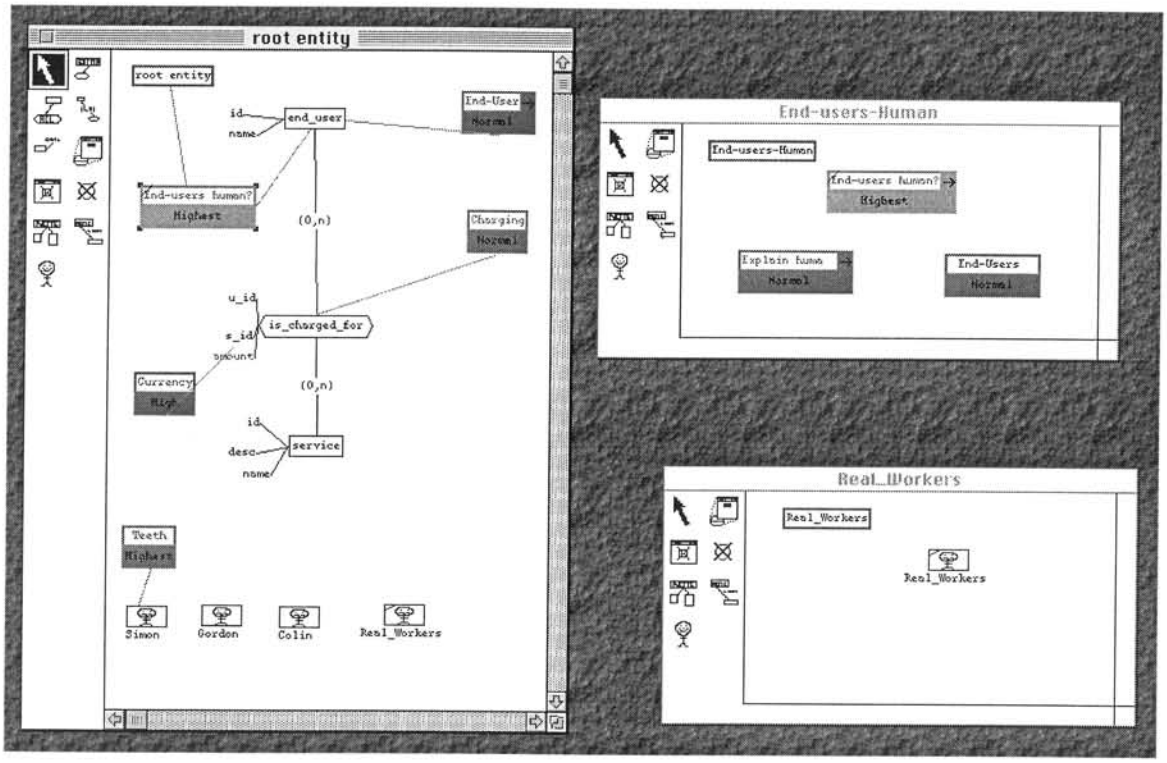

Figure 3.15: Old MV Notes applied to MViews ER

Notes can be graphically linked to other MViews objects, and even to objects in other views. Links to objects in the same view are shown via a line connecting the objects concerned, but this is not so easy across multiple views. Therefore, if an object contains links in other views an arrow symbol is displayed on the right side of the MV Note icon.

Textual views are also available for notes. A textual view can be compiled to update the base note information. A base note is the actual note, and visual representations of this note are different views of the base note.

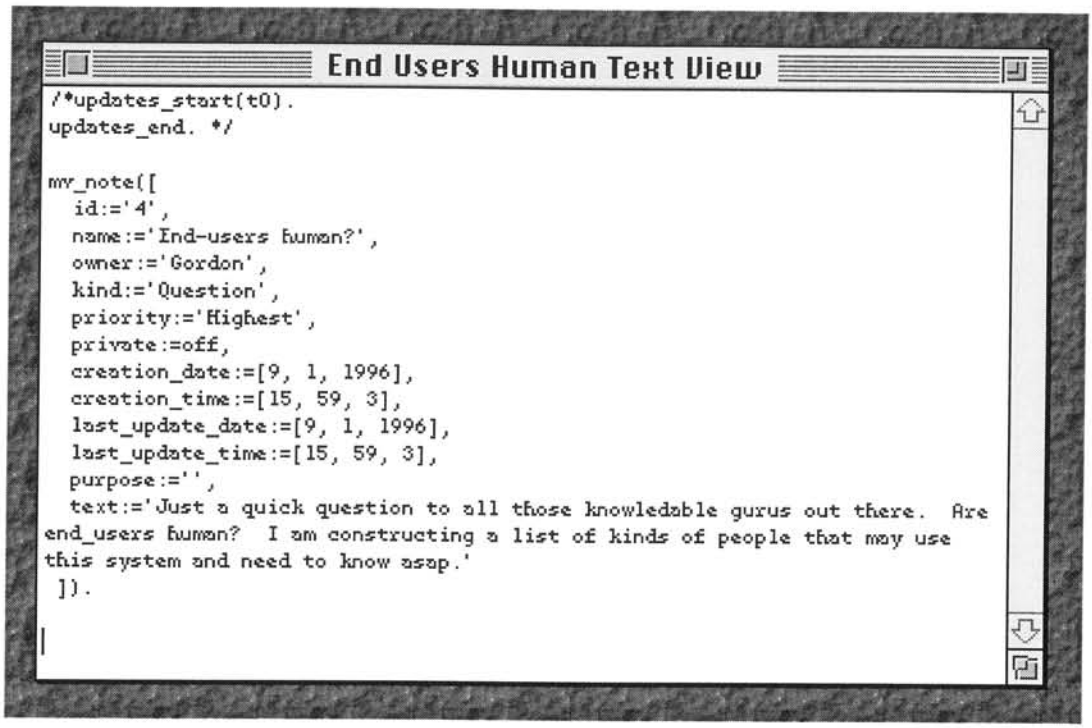

Figure 3.16 : Textual View

3.5.6 Linking between Components

MV Notes can be graphically linked to any other graphical MViews object in the same view via the 'Link Notes' tool. This tool however is restricted to linking within the same view. Since notes can appear in multiple views, a dialogue box was constructed to allow linking between components in different views. Linking is currently only permitted between MV Notes and MV Note users (i.e. linking of Note to an Entity in another view is not permitted).

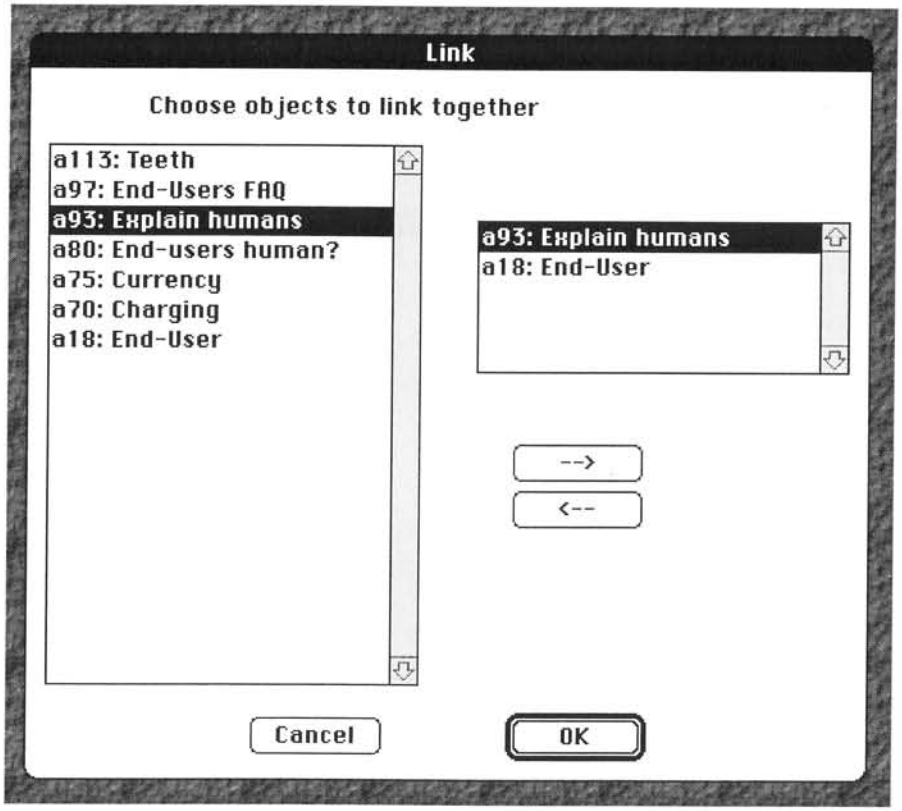

Figure 3.17 : Link Objects

MV Notes can be attached to the view by linking them to an icon with the same name as the view in the top left corner of a newly created view. This icon was added for this purpose, and is part of MV Notes.

A graphical means of linking between components in different views would be beneficial to many MViews applications, however has not been implemented.

3.5.7 Querying Support

The provisions for querying notes are minimal and inadequate. Pre-defined queries available are:

Show All	Displays a list of all notes, listed with id and name
Show All Unread	Displays a list of all unread notes, listed with id and name
Show All Registered	Displays a list of registered notes, listed with id, name and registration level. The option is available to unregister interest in a note.
Show All Hidden	Displays a list of all hidden notes, listed with id and name. The option is available to unhide a note
Show Links	Displays a list of all notes, listed with id and name, that the currently selected note is connected to. External links (i.e. links to objects in other views) are marked accordingly.

Notes can be viewed and edited from any of the pre-defined queries in the table above. All of these queries are automatically updated if they are left open.

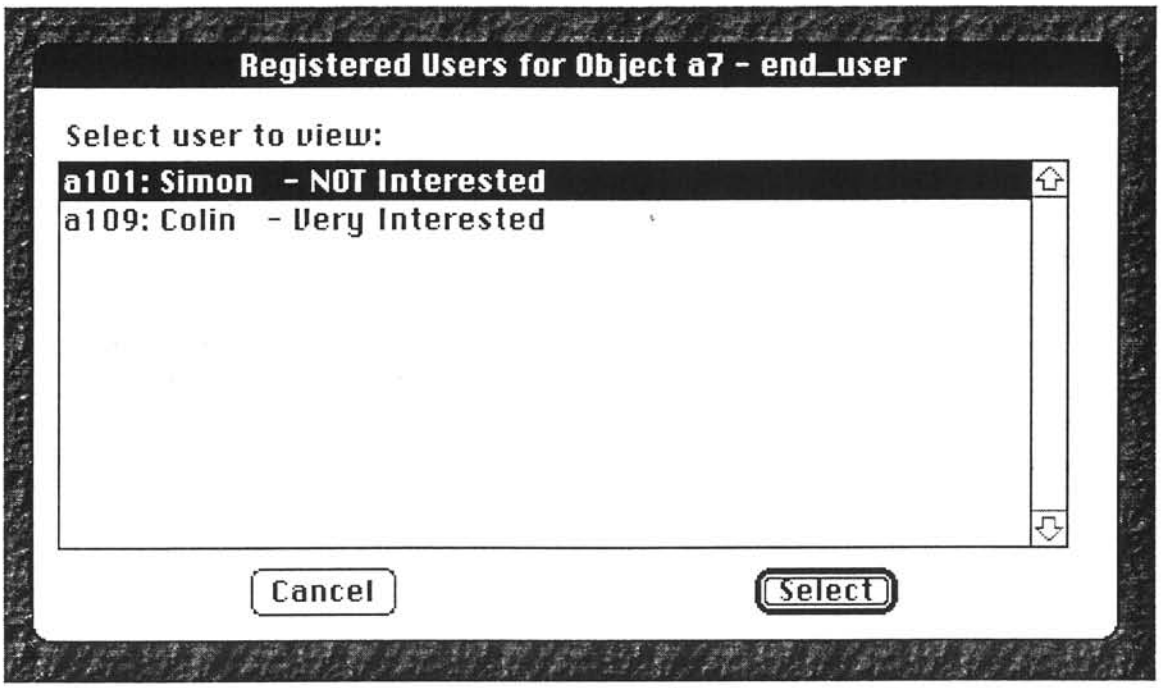

Figure 3.18 : Show All Registered Notes

A search facility for Notes was explored, and a huge ugly impractical dialogue box that allowed complex searching constructed. The code to perform the queries has not been implemented. Ideally, the user would be able to graphically construct complex queries, and have the results continually updated.

3.5.8 Hyperlinks

Hyperlinks were attempted, but proved to be too cumbersome to implement. It was intended that the user could click on the name of an object in the text of a note, and the object would be launched into edit/view mode.

Instead, a basic function was implemented that allowed the user to highlight text that referred to an object and 'edit' this object. This was only implemented for MV Notes and MV Note Users, however could assumingly be adapted for any MViews component.

3.6 MV Note Users

An MV Note User is a user who has access to use MV Notes. When an application using MV Notes is started up the user must log in. This ensures that the user is given appropriate access rights to Notes.

Associated with each user is a list of all the objects they are registered in (see section 3.5.1), a list of the notes they have read, a list of how they intend to be notified about new notes (see section 3.6.1), the access rights they have (see section 3.5.2), and a To do list (see section 3.6.3).

MV Note Users also have the ability to have multiple views. An MV Note User view may contain MV Note Users or views of these components. It is intended that MV Note User Views are used to create user groups.

It is intended that each MV Note User have an icon to represent them (this may exist within one or more user groups). To 'send' a note to a user, all that is required is to attach it to the user's icon. If the icon appears in multiple views it may be attached to any of these icons, since these are different views of the same user.

3.6.1 User Notification

Each user can determine how they wish to be notified about incoming/updated objects. The current method of doing this allows users to classify this action depending on their level of interest in the object and the priority concerned. The objects are distinguished as 'MV Notes' and 'System Objects'. A 'System Object' is any object that is not an 'MV Note'

Actions available are:

Popup Box	Pops up a dialogue box informing the user of the modification. The user then has the option of viewing it, adding it to their To do list, or continuing on with their work.
To do List	Puts the note into the users To do List. The user is not informed about its arrival.
Beep	Displays a banner at the top of the screen, saying that something of interest has been modified. The name and id of the object are displayed. The banner then disappears automatically. The user may check the 'Unread Notes' dialogue to view it. It was initially intended that the 'Beep' action would actually beep as well, but LPA MacProlog32's beep function fails every time it is used!
None	The user is not informed at all. If it was a note that was modified or created, the note is still marked as unread.

The dialogue box used to choose the actions has been criticised to be cumbersome. Other suggestions however have not given the user as much flexibility in notification of changes.

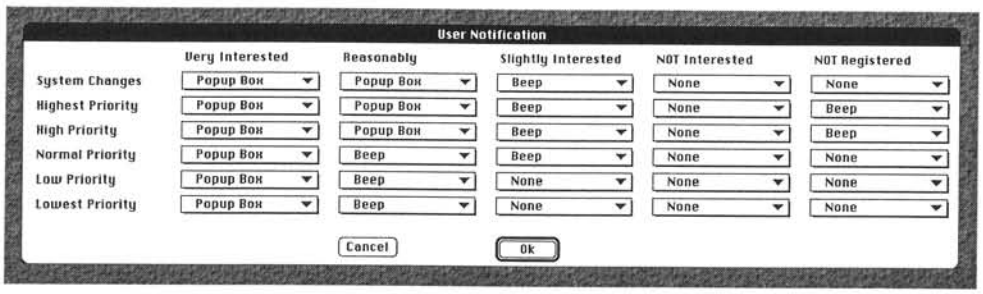

Figure 3.19 : User Notification

Since this dialogue box would be used very infrequently, it would possibly be sufficient as it is. Originally User Notification was fully dependent on how the receiver registered interest in an object, and the sender's priority was not taken into account. For this reason, the cumbersome dialogue box was created. This allowed the user a much greater flexibility level. Visual approaches were suggested, however the models required to contain the same information were even more complex than the dialogue box.

3.6.2 Visual Representations

MV Note User Icons are much simpler than MV Note Icons. An MV Note User Icon contains the name of the user this note refers to at the base of the icon. Double clicking on this icon edits the user information.

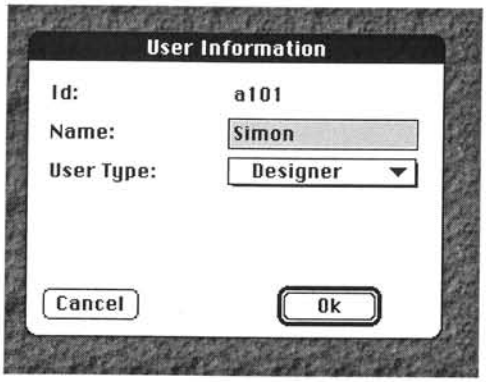

Figure 3.20 : User Information

If the note contains any other views a diagonal line is displayed in the top left corner of the icon (see figure 3.21). It is intended that Note User views are used to indicate groups.

Figure 3.21 : MV Note User Icon

Note Users can be graphically linked to other MViews objects, and even to objects in other views.

3.6.3 To do List

A To do List contains a list of notes. The user can add to, delete, and view notes in this list. It is intended as a temporary storage location for notes.

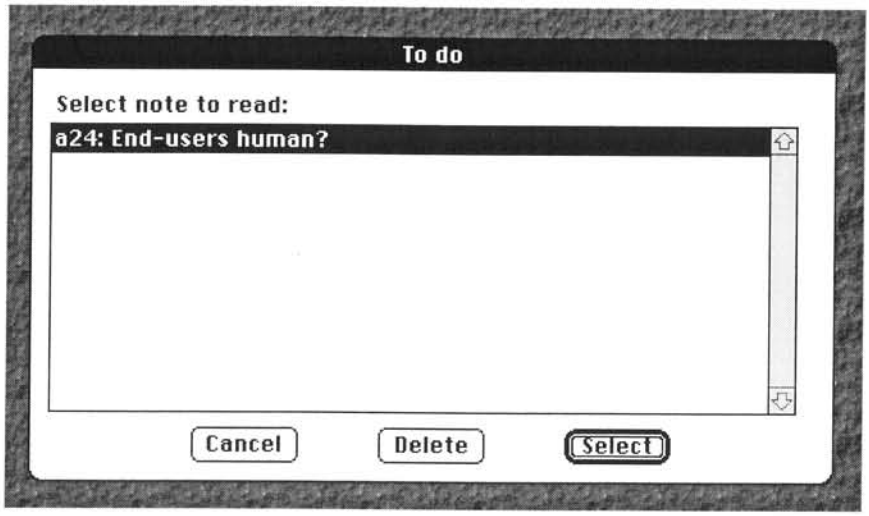

Figure 3.22 : To do List

3.7 Prototype Advantages and Disadvantages

No CSCW support is currently available for MViews, therefore the full extent of MV Notes could not be explored. MV Notes would be used to best advantage on a system with CSCW support.

MV Notes is not completely independent of MViews ER. Ideally it should be part of MViews, thereby ensuring greater independence.

Speed is lacking in the prototype. This is largely due to the amount of data currently getting stored. This could be sped up with more efficient code.

Updates are only processed when the user has finished the task they are doing. This is very beneficial since it could be disrupting if notes arrive while the user is doing other work (e.g. dragging an object across the screen). Most dialogue boxes are updated automatically at the end of each editing cycle (some boxes do not currently support automatic updating).

3.8 Hardware and Software Technicalities

Patches were required to fix some bugs encountered with LPA MacProlog32. Lack of memory was the main problem. If a procedure is given a large number of parameters it sometimes crashes LPA MacProlog 32 completely. Dialogue boxes take up lots of memory, for some unknown reason, thereby inhibiting the number of boxes that can be displayed simultaneously on the screen.

4 Comparisons

4.1 Creating

MV Notes are generally created in isolation from the objects, and are later linked up to relevant objects. They are created by using the "add note" tool and clicking on the graphical display at an empty location. There are however two exceptions to this. If a note is being linked to another note or an MV Note User then it can be done directly with the link note tool by dragging from the icon to attach it to. The creation of MV Notes is restricted to graphical views.

In Muer, notes can be created in a number of ways, but the most common method is to click on the "New Note" button on the "Main View" screen and then click on the object that you wish to attach a note to (the view itself is a valid object). Alternatively, the user can press the buttons marked "New Note" or "Reply" in the "Object Information" and "View All Notes" windows. Both methods bring up the "New Note" window, allowing the user to change the default note settings.

There are advantages and disadvantages to both interfaces. In Muer it is easy to forget you are in "New Note" mode and create new notes by accident. It is tedious to send a note to objects that are not on the screen, such as users, as the user must create a note to some visible object and use the "Add" and "Remove" buttons to choose the appropriate target. With LPA MacProlog32, the enter key is bound to the "Ok" button within dialogue boxes. With MV Notes, this meant that users often used the enter key when they didn't want to exit the dialogue box. Users requested that replies have the text of the old note inserted automatically (and quoted) in the reply.

4.2 Views

Since MViews is a platform that allows multiple views MV Notes can also be displayed and linked between multiple views. An MV Note can have graphical and/or textual views. A graphical view of an MV Note does allow the operations permissible in a MViews ER view. The operations have been restricted so that an MV Note View has a special meaning. It is intended that this view is for replying to notes. Likewise, an MV Note User View only allows MV Note Users and links between them. This view is used to define user groups. A textual view of an MV Note contains a Relational Database Schema (RDS) for the note. The user can modify and compile this schema (assuming they have the access rights), and all graphical views will be updated accordingly.

In Muer, there is only one view of the diagram, called the "Main View". This displays the diagram in it's entirety. Notes can be attached to the main view, and are displayed as an icon on the top left corner of the view. Notes can also be viewed in lists in the "Object Information", "View all Notes" and "View all Objects" windows, and in the "Read Note" window described below.

Most of the graphical views, in both interfaces, were easily understood. The MV Notes views tended to get very cluttered when people didn't use "MV Note View" and "MV Note User" windows (and these windows were not obvious).

4.2.1 Icons

MV Note Icons are a view of an MV Note. They were originally displayed as big ugly graphical objects, though this has been changed to a newer and less bulky form, as described above. The appearance of an MV Note Icon is determined by the attributes of the note it represents and whether it is linked to objects in other views. The icon can be hidden from the screen either via filtering or using the "hide tool".

The Muer interface is deceptive in that notes are not really viewed as icons at all. When a note is sent to an object, that object's icon is altered to show that there are notes attached. The note does not have an icon of it's own. The "Show" menu can be used to control what kind of flags are visible on the screen.

The Muer note icons were not intuitive and suggested the wrong usage to most users. The MV Notes icons were far more intuitive, but were very large and quickly cluttered the screen. Both systems used colour to present extra information about the note, though this could be a problem for colour-blind users. MV Notes icons showed part of the note's name on the icon: this can be confusing with long names. On the other extreme, the icons representing the presence of notes in Muer held very little information, and users suggested that some indication of how many of each note were represented would be advantageous.

4.2.2 Filtering

Within MV Notes, Filtering is only applied to graphical views. Filtering can be done based on the kind of message, the priority, whether interest has been registered in it, and whether it is unread. Muer, on the other hand, has a very crude filtering system, only allowing filtering on the basis of urgency and whether the note has been read.

The filtering in Muer was not used extensively, and in MV Notes users tended to hide individual notes rather than filtering the entire set of notes.

4.2.3 Reading

MV Notes can be read by clicking on the note icon while the "note tool" is selected, or double clicking on them with the "pointer tool". They can also be viewed from all queries by double clicking on them. MV Notes can also be read from a textual view. An MV Note can be modified by editing the text within it. All changes will be recorded noting who made the change and when.

In Muer, notes are read by double-clicking on them in a list view. Generally, this is the list that appears in the "Object Information" window. Muer's "View Note" window shows all the information about a single note in a window. Unlike the similar window in MViews, the only ways that the note can be altered are to delete it, change it's urgency flag (a change that is not propagated to the other users), or to change the objects it is linked to.

Users complained that it took too many use actions to read a note in Muer, as opposed to MV Notes, where a single double-click loads the note. Both systems had very similar note windows, both clearly based on e-mail layouts (i.e.: subject and other details at the top, the note text at the bottom). The users were all familiar with and understood this layout.

4.3 Links

MV Notes can be graphically linked to any MViews graphical object in the same view by using the "note link tool". This includes the links between objects (i.e. a note can be linked to the link between other notes). They can also be linked to MV Notes and MV Note Users in other views via a clumsy menu option. Each graphical view contains an object with a name of the view the top left hand corner. MV Notes can be attached to a view by linking the note to the graphical view object.

Muer allows the user to link notes to objects when they are created, and any user may re-link the note at any time. This is done with the "Add" and "Remove" buttons in the "New Note" and "Read Note" windows respectively.

A drawback of Muer's textual methods for linking of notes is that it can be hard to tell exactly which entity is which in the listing without the icon as a reminder. Further, large lists proved unmanageable: they were too long to fit on the screen, and difficult to search. Linking of MV Notes to objects in the same graphical view could clutter up the screen quite successfully, seeming to make the notes more important than the actual work being attempted.

4.3.1 Threads

Threads are notes written in reply to other notes. In MV Notes a graphical view of a note can be created to contain a thread, and the note icon is altered to indicate the presence of any replies. Threads can be nested within threads, if necessary.

Muer handles related notes by defining any note that is not in reply to any other to be in a new thread, and any replies to a note as being part of the thread as the note they are in reply to. Threads have a hierarchical structure that can be seen in any list of notes.

MV Notes has the a graphical interface for creating threads, however it is a complex interface. Muer is a lot simpler--the user need not even be aware of why notes are being added to the thread--but lacks a graphical representation of the structure of the replies. Also, there is some ambiguity over whether notes can belong to more than one thread, as a note can be in reply to several threads but only belong to one of them.

4.4 Deleting

MV Notes can be deleted by selecting the "delete tool" and clicking on the note to delete. The user will be confirmed whether they really do want to delete it. All links associated with the note will also be deleted. In Muer, the only way to delete notes is from a window where the notes are listed and a delete button is present.

The main view of Muer also has a delete tool, and many users have tried to delete notes by selecting it and clicking on a note icon. Unfortunately, as note icons are only additions to other entity icons, this results in the user accidentally deleting the note. Understandably, these users were a little upset. Deleting worked fine in MV Notes but some references to the note were often left dangling around.

4.5 Types

An MV Note can be one of four types; Message, Explanation, Question, or Example. There is one other type that can be used in addition to those above, and that is the Private type. A Private note can only be viewed by its owner. The type of a note is determined by the creator. Muer has a much more complex type structure, described in section 2.4.5.

The drawbacks of Muer and MV Notes complex structure are that users may not be sure exactly what type of note they are sending, and have complained that they wasted time worrying about it when, in fact, it is absolutely unimportant. The only difference it makes in MV Notes is in the appearance of the icon.

4.6 Priorities

MV Notes supports five different levels of priority; Lowest, Low, Normal, High and Highest. The priority of a note is determined by the creator. Muer notes only have two real priority

levels: urgent and not urgent. However, Muer attempts to let the sender show how important a note is by putting a "Response time" on it, and by optionally interrupting the recipient's work.

Users commented that Muer should have a greater range of priority settings.

4.7 Registration of Interest

MV Notes allows users to register interest in any MViews object (including other notes, and even links between notes). There are four different levels of registration available; Very Interested, Reasonably Interested, Slightly Interested and NOT Interested. The latter is used to filter out annoying change messages.

As with the Priority issue, Muer attempts to make a users level of interest in an object specific. Rather than assign a level of interest to an object, the user must decide from a list of options exactly what response is appropriate to an incoming note. The options include variations on interrupting the user, setting flags, and ignoring the note. This effectively combines the MV Notes "Registration of Interest" and "User Notification" dialogue, but in a simpler fashion.

4.8 User Notification

With MV Notes, each user can determine via a huge pop-up box, how they wish to be notified about incoming messages. The user can classify the action depending on their level of interest in the object and the priority that it has. Actions available are; Pop-up box, Put note in To do List, Beep and None. In addition to this the "Notes Unread" dialogue box would be automatically updated (Muer has a similar feature). In Muer the user was notified by the appearance of icons on the screen, entries in lists and pop-up windows.

The huge pop-up box in MV Notes invoked criticism by many. A macro language was suggested to get around using a huge pop-up box. Users thought that continually updating lists (c.f. Eudora) would be a good form of notification.

4.9 Searching

Searching in MV Notes could be done via the "Find - Advanced" dialogue (never implemented). It was intended that queries created with this be automatically updated. The use of several menu options provided a means of displaying a subset of notes e.g. "Show All Registered Notes" and "Show All Hidden Notes".

To let the user search for a specific note in Muer, the "View All Notes" dialogue was created. This has been through several versions. Originally, a very complex search engine was envisaged, similar to MV Note's' "Find - Advanced" dialogue. However, this was simplified (before it was even implemented) to a system of filtering. The user can choose from a list what types of note they would like to see, and can decide whether these should be chosen from all the notes, the new notes, the urgent notes, or even the deleted notes.

As was expected, users thought the "Find-Advanced" dialogue was ridiculously complex. A graphical form of querying may be more appropriate.

4.10 Update History

All changes made to MV Notes are recorded. Information recorded includes the change made, who made the change, and the time it was made. Changes can be undone and redone.

The usefulness of this feature is debatable, however it is useful to have as an Audit trail.

4.11 Note Access

MV Notes allows access to be restricted to certain users to perform certain tasks. Tasks that can be restricted are; Linking to a note, Deleting, Moving, Viewing, Editing, Adding a view, Hiding and Granting access rights.

Being able to restrict Moving was seen as completely impractical and frustrating.

4.12 To do List

Within MV Notes, a To do List contains a list of notes. The user can add to, delete, and view notes in this list. It is intended as a temporary storage location for notes.

4.13 Hyperlinks

These were attempted in MV Notes, but proved too cumbersome to implement. Instead, a basic function was implemented that allowed the user to highlight text that referred to an object and 'edit' this object. These objects however were allow implemented for MV Notes and MV Note Users.

4.14 Dialogue capture

MV Notes and Muer provided no support for dialogue capture.

4.15 Summary

Being able to add notes to text or objects is advantageous as long as the notes are not too cumbersome to use. Notes should not interfere with the task the user is trying to perform. Big bulky note icons (c.f. MV Notes) can get in the way. The icons used in Muer for notes were not as obtrusive but would have been better if they were separate from the object they were associated with. This would ensure that a note could be easily deleted without deleting the object that it is attached to! If the notes are too complex they will have a tendency to detract from the work actually done.

A note manager is useful as long as not too much is required of it, and should be mostly transparent to the user. It should be used to maintain notes and threads, perform notifications, and possibly permit simple searching. The use of threads is beneficial since it helps to organise notes. In practice threads should not get very large, but serve more as a reminder of discussions. Thread creation should happen automatically, requiring little (if any) user interaction to group replies.

5 A Simpler System

"Minimal Notes" was developed after Muer and MV Notes had been evaluated. It was developed to reflect a different style of note system. Many of the features of Muer and MV Notes were incorporated into the design, and many were omitted. The prototypes had shown that complex note systems could be counter-productive, therefore the objective of this system was to produce a simple (yet useful) note system.

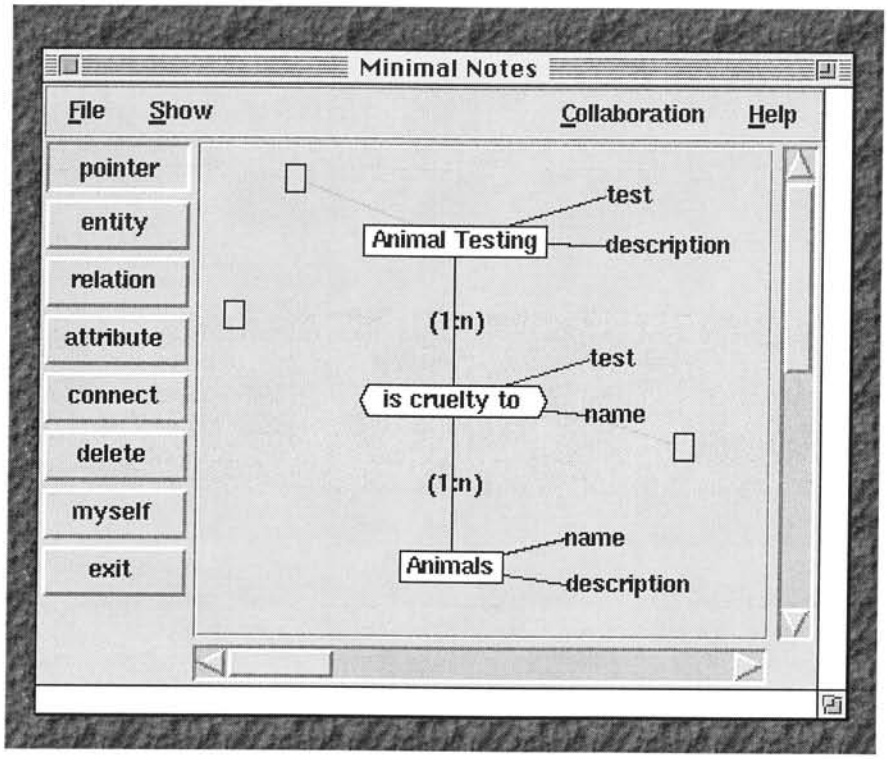

Figure 5.1 : Minimal Notes applied to Muer

Since both software developers had experience with Tcl/Tk [Ousterhout 1994], Minimal Notes was written using GroupKit [Roseman et al. 1995]. GroupKit provides CSCW support to Tcl/Tk applications.

Minimal Notes can be added to both text and canvas widgets in GroupKit. To add Minimal Notes to an existing GroupKit application, all that is required is two lines of code. The first line of code contains the source file for the notes application. The second line calls a procedure in the source file to start the note system. Figure 5.2 shows the full code for a text widget with CSCW support and the Minimal Notes extension.

```
gk_initConf $argv          #initialises GroupKit
pack [text .t]             #creates and displays a text widget

source notes.tcl           #loads Minimal Notes
start_notes .t             #applies Minimal Notes to the text widget
```

Figure 5.2: Full code for a text widget with CSCW support and Minimal Notes

The only change required to apply Minimal Notes to a canvas is to change the second line to pack [canvas .t].

5.1 Creating

Notes can be created by holding down the "Control" key and clicking on the location where the note is to be added. This same approach works for both canvas and text widgets. When a note is created a popup box (see Figure 5.3) will be displayed.

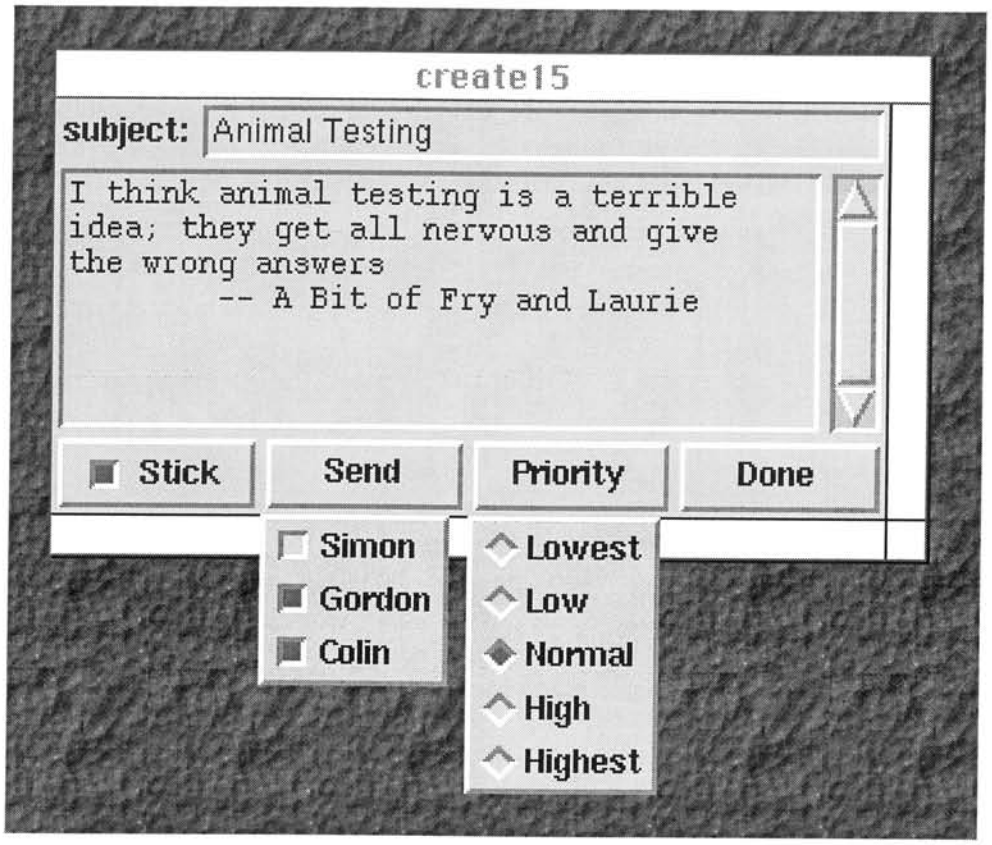

Figure 5.3 : Note Creation Popup Box

The field at the top of the box is used to enter a subject. If no subject is entered, this will be defaulted to "no subject". The area between the subject and the buttons is for the content of the note. Since this is a text widget, notes can also be added to any point in the note by Control-clicking on the text.

The buttons at the bottom of the note are used to determine what the note should do. The "Stick" checkbutton (defaulted to on) specifies whether the note is to be stuck to the canvas or text widget it was created on. If this is turned off the icon will not be displayed. Selecting the "Send" button allows the user to determine who to send the note to (if anybody). The "Priority" menu- is used to specify how important the note is. If the note is being sent to people, then this setting determines how they are notified. The "Done" button completes the note. If the user has selected anybody in the "Send" list then the note is sent to them, and if the "Stick" button is checked a note icon will be displayed.

5.2 Views

5.2.1 Icons

Note icons are displayed differently on canvas and text widgets. On text widgets the note is applied to the character closest to the cursor. This character will be marked with a yellow background and be slightly raised to denote that it can be clicked on. If the note is unread then the character will be displayed in red. If text is written around the note icon, then the icon will move with the character it is attached to.

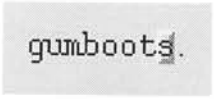

Figure 5.4 : Note icon on text

On canvas widgets the note is attached at the current cursor position. The note has a yellow background. If it is unread then it has a red border, otherwise the border is black.

Figure 5.5 : Note icon on canvas

For Muer, if a Note is attached to a location that contains an existing Muer object (eg. entity), then the note is attached to that entity via a yellow sticky line. There is no limit as to how many notes can be attached to any object. Whenever the Muer object moves, all notes attached to the object move with it.

Figure 5.6 : Note attached to Attribute

5.2.2 Reading

Double clicking on a note icon displays a popup box with the note in it (see Figure 5.7).

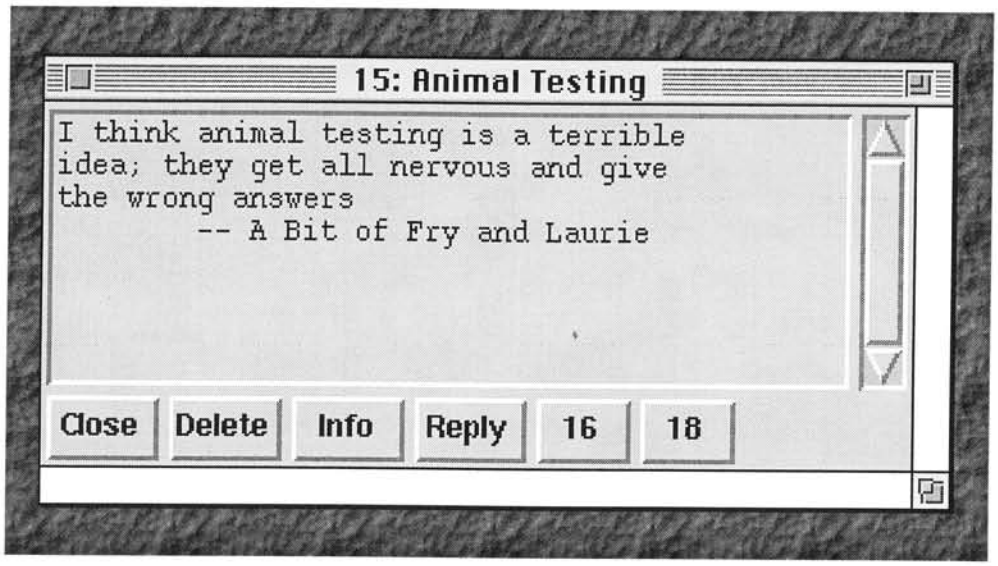

Figure 5.7 : Note Reading Popup Box

The id and subject of the note is diploid in the title bar of the note. The user cannot modify text in the note, however they may add additional notes to it by Control-clicking on the notes text. "Close" quits reading the note. "Delete" deletes the current note. All replies and notes to the note will be lost. "Info" displays additional information about the note (see Figure 5.8). The "parent" field of the Info section contains a hyperlink to the note that this note is a reply to (if there is one). Pressing the "Info" button again closes down the Info section. The "Reply" button can be used to add a reply to a note. Depressing this button pops up the same window used to create a note. The priority of the note being sent is defaulted to the priority of the note being replied to, as is the note subject. By default the note is sent to all the original recipients and the sender. The note being commented on will also be included in a quoted form. Any replies to the current note are shown to the right of the "Reply" button. These are currently displayed as buttons with id of the note on them.

Replies can be viewed by clicking once on the button of the note to view. This extends the current window to include the reply. Likewise replies to replies can be viewed by clicking on their icon (see Figure 5.8). The "Launch" button displays note in their own window.

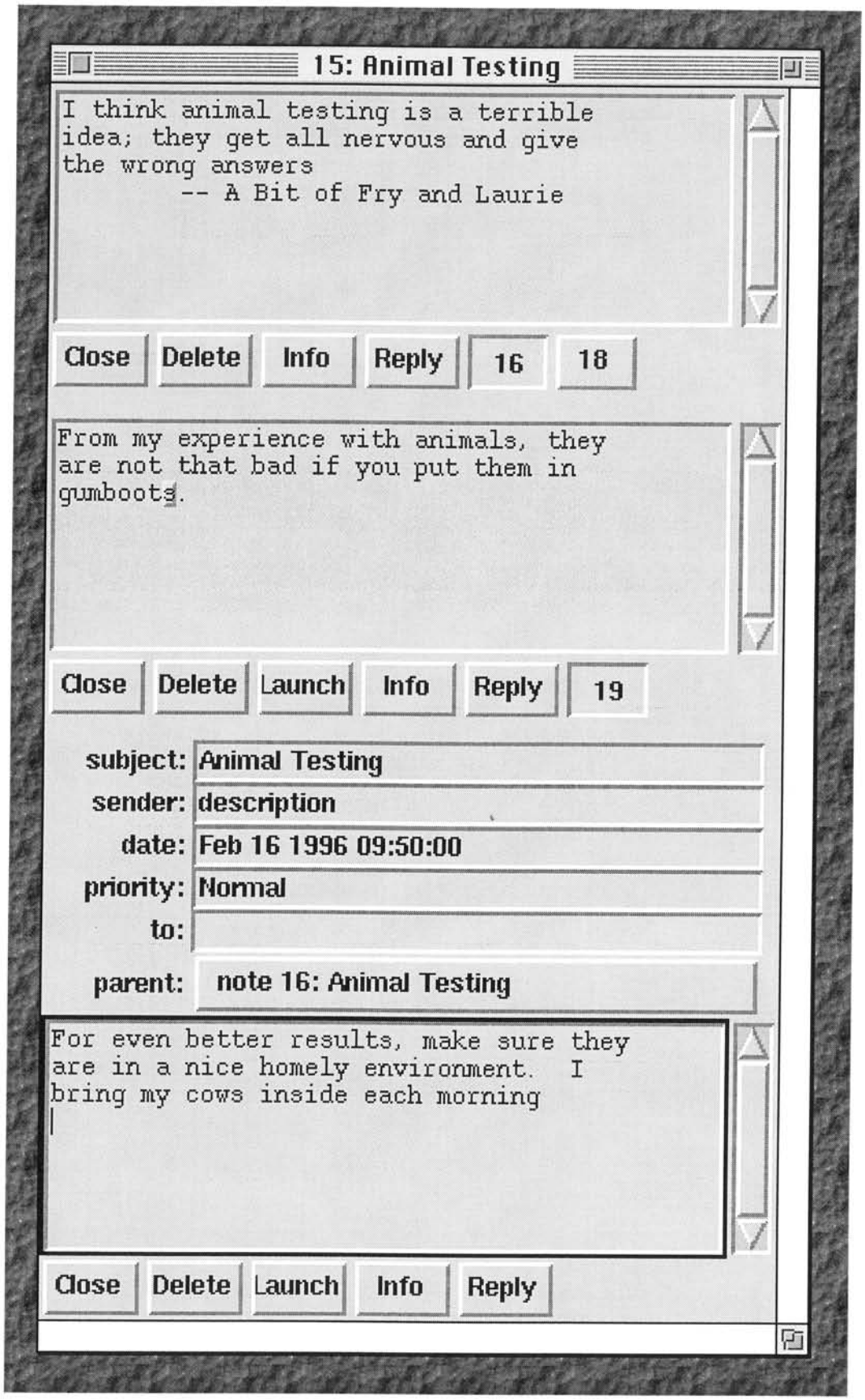

Figure 5.8 : Example of replies to notes

5.2.3 Show Menu

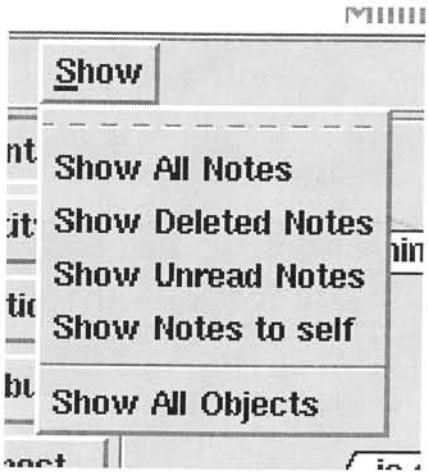

Figure 5.9 : Show Menu

The "Show" menu allows the user to display various note and object information. The "Show All Notes" option displays a list of all notes (see Figure 5.10). Notes can be read from this list by double clicking on the note to read. If a "d" is shown to the left of a note, then this indicates that the note has been deleted.

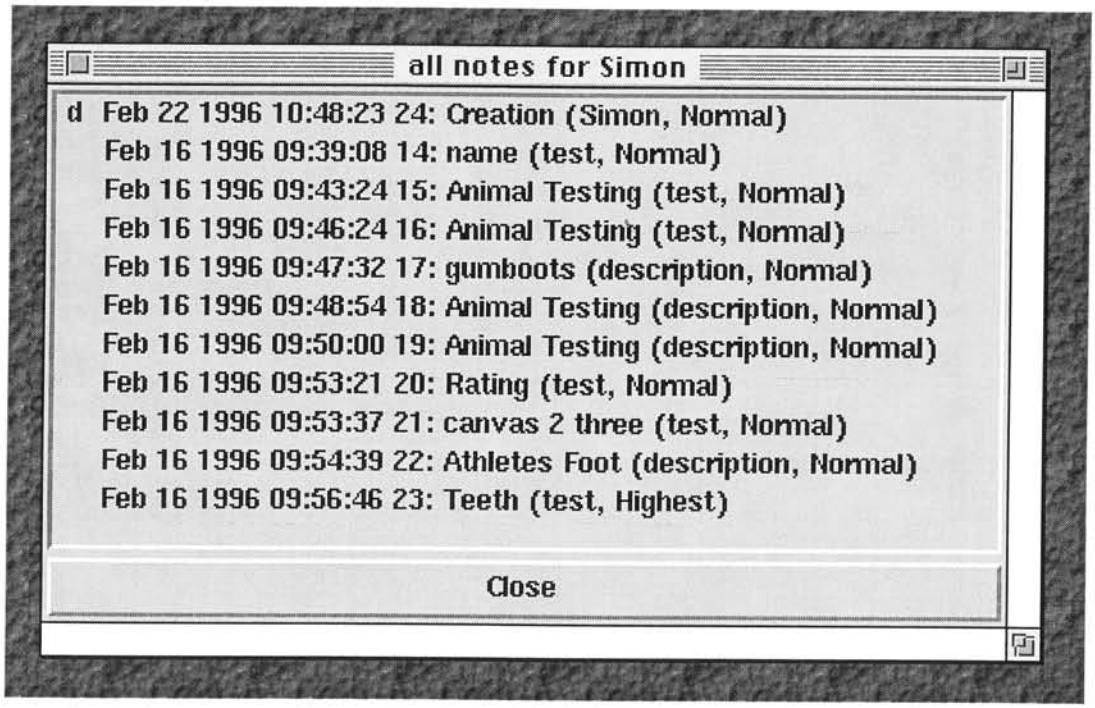

Figure 5.10 : Show All Notes

The "Show Deleted Notes" menu option is similar to the "Show All Notes", but only shows deleted notes. Notes can be undeleted by selecting them and pressing the "Undelete" button.

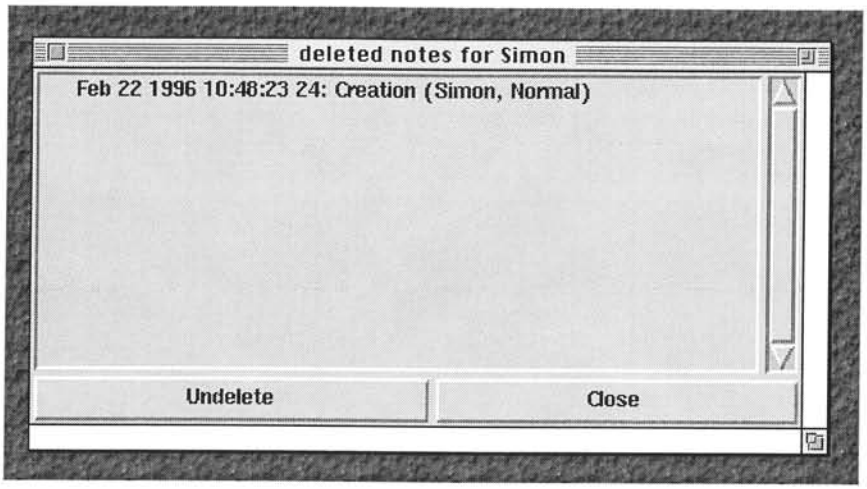

Figure 5.11 : Show Deleted Notes

"Show Unread Notes" displays only notes that the user has not read yet.
"Show Notes to self" displays all notes that have been addressed specifically to the current user (see the User Notification section for more details).
"Show All Objects" contains information about all objects used in Muer including People (see Figure 5.12). Objects can be deleted, undeleted and completely removed from the database. Once something in completely removed from the database it cannot be undeleted.

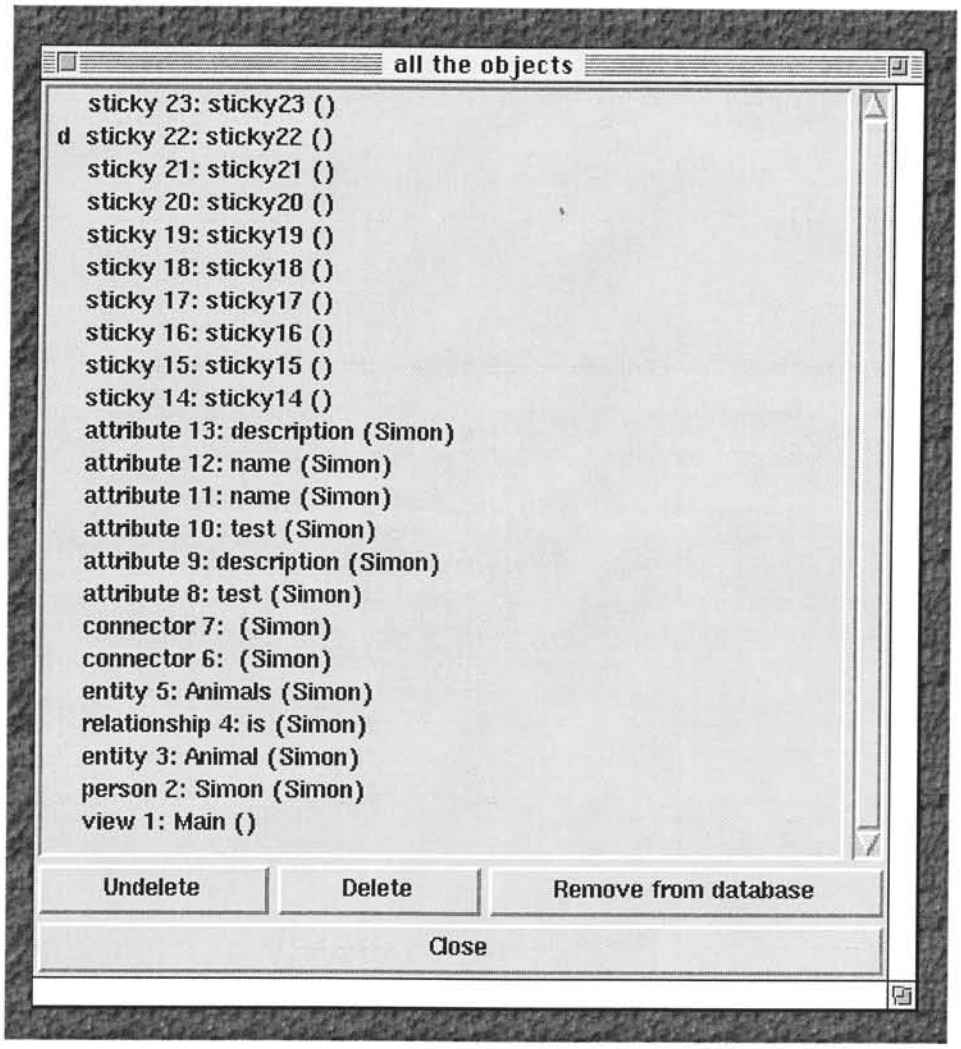

Figure 5.12 : Show All Objects

5.3 User Notification

Notes (more accurately, Messages) sent to a User are displayed in a popup box (see Figure 5.13). An "n" will be displayed to the left of a note if has not been read yet. The date and time the note was sent is shown to the left of this followed by the id of the note, and the notes subject. The sender of the note and the priority assigned to the note are displayed in brackets after the subject. The note can be deleted from this list by selecting the note to delete and pressing the "Delete" button.

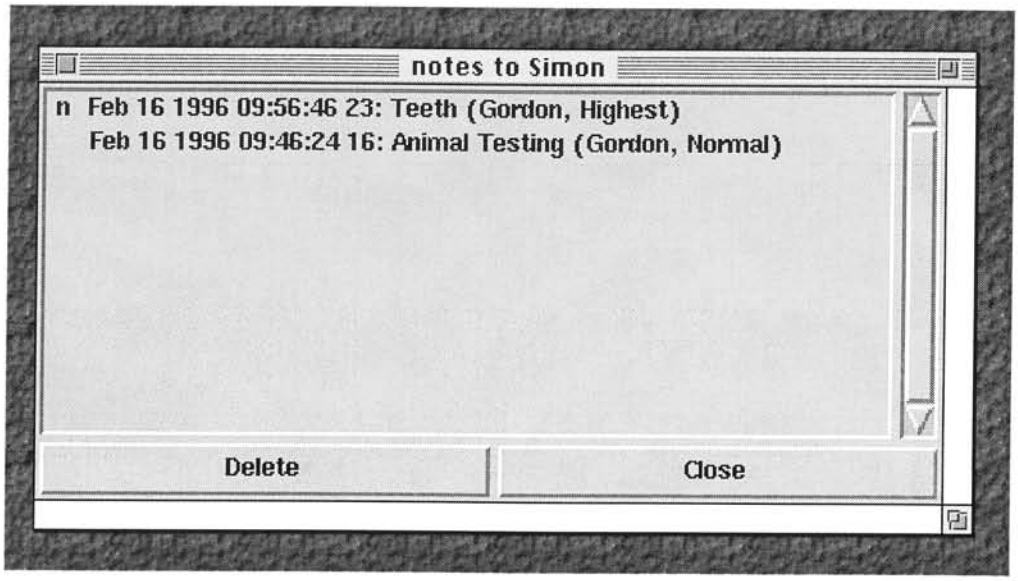

Figure 5.13 : Incoming Messages Box

The user can be notified of new notes in four different ways, depending on the priority of the note.

Priority	Action
Lowest	Put in incoming box
Low	Put in incoming box
Normal	Put in incoming box, Beep Once
High	Put in incoming box, Beep Twice
Highest	Put in incoming box, Beep Three Times

If the user does not have their "incoming message box" open when a message arrives, it will be opened automatically for the user.

Telepointers are another form of group awareness that is supported by Minimal Notes. The Collaboration Menu (see Figure 5.14) contains a checkbox to allow the user to turn telepointers on and off for other users screens. The telepointer displays the location of others cursors on the diagram. It can be useful to see the context they are working in.

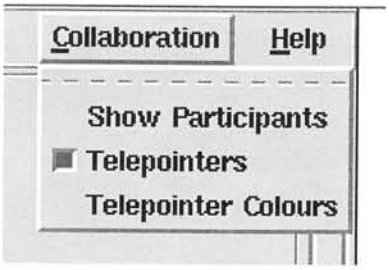

Figure 5.14 : Collaboration Menu

The colour of other users telepointers is defaulted to that users default colour. To differentiate between telepointers, each user can specify what colour they want other users telepointers to be displayed on their screen (see Figure 5.15).

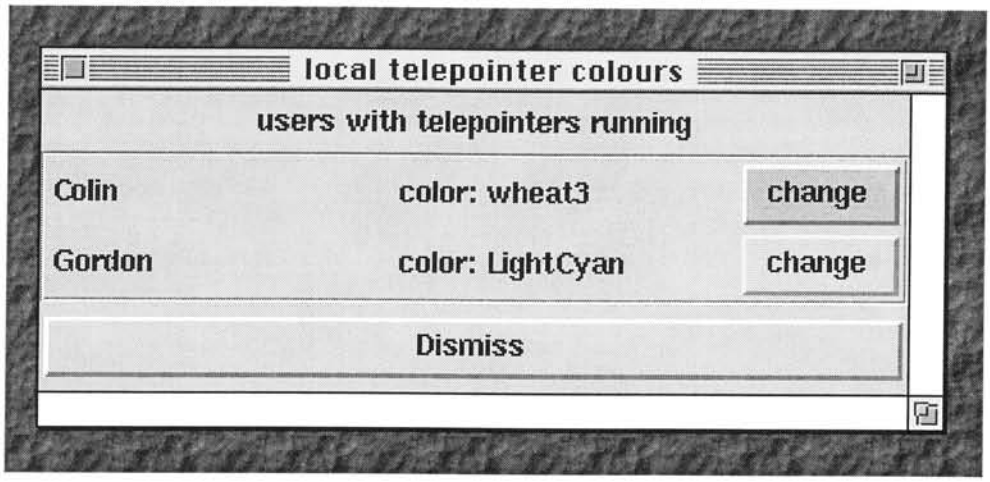

Figure 5.15 : Telepointer Colours

The other option in the Collaboration Menu, "Show Participants", displays information about the other users currently connected to the current conference.

5.4 Logging in

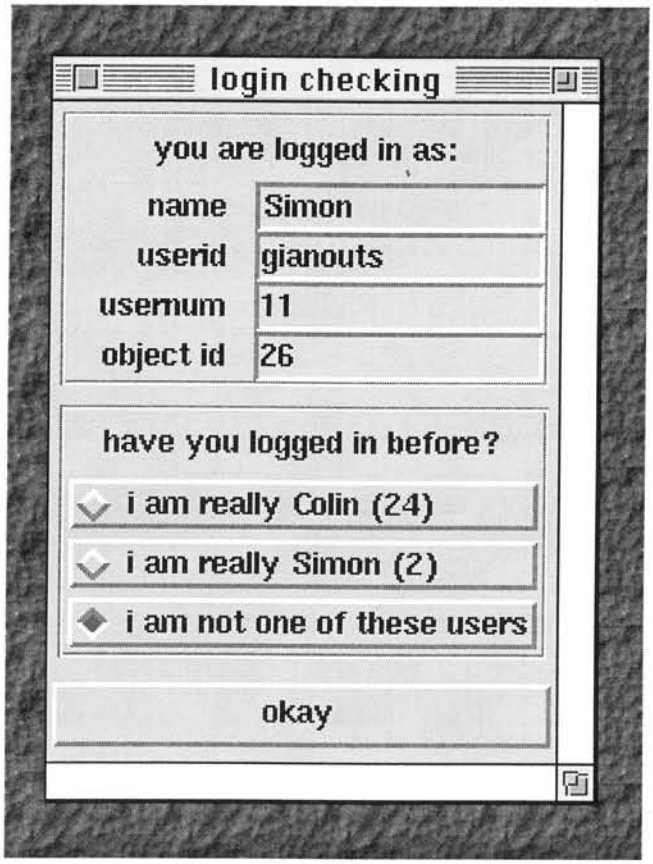

Figure 5.16 : Login Checking

If other users have joined and left the conference, information is still retained about them. So that users are informed of changes made in their absence, and can receive notes sent to them, login checking has been implemented. Login checking involves prompting the user to choose who they are from a list. If they are not on the list, they can connect as a new user.

6 References

[Grundy 1993] Grundy, J. (1993) "Multiple Textual and Graphical Views for Interactive Software Development Environments" PhD thesis, Department of Computer Science, University of Auckland.

[LPA 1994] Logic Programming Associates Ltd. (1994) "LPA MacProlog32 User Guide" London, England.

[Mugridge 1994] Mugridge, R. (1994) "Snart: a Mixed-Paradigm Language" Working Paper, Department of Computer Science, University of Auckland, September 1994.

[Ousterhout 1994] Ousterhout, J.K. (1994) "Tcl and the Tk toolkit", Addison-Wesley, 1994.

[Roseman et al. 1995] Roseman, M. and Greenberg, S. (1995) "Building Real Time Groupware with GroupKit, A Groupware Toolkit", Department of Computer Science, University of Calgary, Calgary, Canada, July 1995.

[Smith et al. 1993] Smith, B.C., Rowe, L.A. and Yen, S.C. (1993) "Tcl Distributed Processing" in Proceedings of the Tcl/Tk Workshop, Berkeley, California, U.S.A.

[Venable and Grundy 1995] Venable, J, Grundy, J.C. "Integrating and Supporting Entity Relationship and Object Role Models" 14th OO/ER Conference, Brisbane, Australia; December.