

Groupware design:
principles, prototypes, and systems

Andrew Jeremy Gavin Cockburn

Department of Computing Science and Mathematics

University of Stirling

Submitted in partial fulfilment of
the degree of Doctor of Philosophy

April 1993

Abstract

Computers are valuable tools for a wide range of work tasks. A substantial limitation on their value, however, is the predominant focus on enhancing the work of individuals. This fails to account for the issues of collaboration that affect almost all work. Research into computer supported cooperative work (CSCW) aims to eliminate this deficiency, but the promise of computer systems for group work has not been met.

This thesis presents four design principles that promote the development of successful groupware. The principles identify the particular problems encountered by groupware, and provide guidelines and strategies to avoid, overcome, or minimise their impact. Derived from several sources, the major influence on the principles development is an investigation into the relationship between factors affecting groupware failure. They are stimulated by observations of groupware use, and by design insights arising from the development of two groupware applications and their prototypes: *Mona* and TELEFREEK.

Mona provides conversation-based email management. Several groupware applications allow similar functionality, but the design principles result in *Mona* using different mechanisms to achieve its user-support.

TELEFREEK provides a platform for accessing computer-supported communication and collaboration facilities. It attends to the problems of initiating interaction, and supports an adaptable and extendible set of "social awareness" assistants. TELEFREEK offers a broader range of facilities than other groupware, and avoids the use of prohibitively high-bandwidth communication networks. TELEFREEK demonstrates that much can be achieved through current and widely accessible technology.

Together, *Mona* and TELEFREEK forcefully demonstrate the use of the design principles, and substantiate the claim of their utility.

Contents

1	Introduction	1
1.1	Undefining CSCW	2
1.2	Defining groupware	3
1.3	Principles for groupware design	4
1.3.1	Origin of the principles	6
1.3.2	Scope of the principles	7
1.4	Thesis structure	7
1.5	Appendices	9
1.6	Summary of thesis contribution	10
2	Overview of groupware systems and CSCW research	11
2.1	Introduction	11
2.1.1	Concerns on groupware classification	11
2.1.2	Structure of the overview	14
2.2	Messaging systems	15
2.2.1	Filtering systems	17
2.2.2	Passive conversation systems	23
2.2.3	Active coordination systems	27
2.2.4	Toolkits for messaging systems	33
2.3	Co-authoring systems	35
2.4	Meeting Environments	36
2.4.1	Low-technology computer conferencing systems	38
2.4.2	High(er)-technology conferencing systems	40
2.4.3	Meeting room environments	45
2.5	Seamless interaction and social presence	51
2.5.1	Background	52
2.5.2	Seamless interaction environments	53
2.5.3	Social presence systems	58
2.6	Summary of the overview	61
3	Problems in user-acceptance of groupware	63
3.1	Introduction	63
3.1.1	Chapter overview	64
3.2	Grudin's observations on groupware failure	64
3.3	The vicious circle confounding groupware's success	66
3.3.1	User-effort: the key determinant	69

3.4	Effort inherent in collaboration in general	70
3.5	Effort inherent in use of telecommunication channels	70
3.6	Effort imposed by systems	71
3.7	Explicit system requirements	71
3.7.1	Filtering Schemes	72
3.7.2	Passive conversation support	72
3.7.3	Active coordination support	73
3.7.4	Guidance dependence in asynchronous messaging systems	74
3.7.5	Requirements in synchronous groupware	78
3.8	Lack of flexibility	78
3.9	Lack of integration between systems	79
3.10	Summary	80
4	Four principles for groupware design	82
4.1	Introduction	82
4.1.1	Chapter overview	82
4.2	From groupware problems to groupware principles	83
4.2.1	System oriented view of user-effort	83
4.2.2	Introducing the principles	86
4.3	Maximise personal acceptance	87
4.3.1	Catchpenny systems	87
4.3.2	The “Reflexive Perspective” of CSCW	88
4.3.3	Champions and encouragement	90
4.3.4	Participatory design	90
4.3.5	Implementing personal acceptance	91
4.4	Minimise requirements	91
4.4.1	Avoid dependence on user actions	92
4.4.2	Use information that is available “for free”	93
4.4.3	Equal opportunity: enable shifts in cost and benefit	94
4.4.4	Strategies for minimal requirements in conjunction	95
4.4.5	Implementing minimise requirements	98
4.5	Minimise constraints	99
4.5.1	Be aware of the two level perspective of technology	100
4.5.2	Beware of explicit models and theories of collaboration	101
4.5.3	Open, unconstrained enhancement	101
4.5.4	Implementing minimal constraints	102
4.6	Maximise external system integration	102
4.6.1	Video fusion	103
4.6.2	Heterogeneous environments	104
4.6.3	Minimise dependence on structure and format	104
4.6.4	Implementation platforms	105
4.6.5	Implementing external integration	105
4.7	The principles’ effect on design and implementation	106
4.7.1	A note on evaluation	106
4.8	Relating the principles to the vicious circle	107
4.9	Summary	109

5	Mona: The principles and conversational email	110
5.1	Introduction	110
5.2	Prototypes and investigations	111
5.2.1	HyperCommitments	111
5.2.2	Personal <i>and</i> communicated reminders in <i>mnd</i>	114
5.2.3	Pre- <i>Mona</i> studies	116
5.3	<i>Mona</i>	116
5.3.1	<i>Mona</i> 's aims	117
5.3.2	System overview	118
5.3.3	Conversational context "for free"	120
5.3.4	Using and modifying conversational context	124
5.4	<i>Mona</i> and the principles	125
5.4.1	Maximising personal acceptance	125
5.4.2	Minimising requirements	127
5.4.3	Minimising constraints	128
5.4.4	Enabling external integration	129
5.5	Summary	130
6	TELEFREEK: Integrating collaboration support	132
6.1	Introduction	132
6.1.1	Chapter overview	133
6.2	Limitations of "social" and "seamless" groupware	133
6.2.1	Imitation of proximity or augmentation of collaboration?	133
6.2.2	Other problems	134
6.2.3	Synergy in collaboration support	136
6.3	Integrated collaboration support in TELEFREEK	137
6.3.1	System overview	138
6.3.2	Extendability, flexibility, and customisation	144
6.3.3	TELEFREEK as a prototype	146
6.4	Determining <i>who?</i> and <i>how?</i> in communication	146
6.4.1	Selecting the right people for communication	148
6.4.2	Selecting the right communication channel	151
6.5	Summary	154
7	Conclusions and further work	156
7.1	Introduction	156
7.2	An overview of CSCW and groupware	157
7.3	Problems in user-acceptance of groupware	157
7.4	Four principles for groupware design	158
7.4.1	Further work with the principles	160
7.5	Automatic conversational context in <i>Mona</i>	160
7.5.1	<i>Mona</i> : related further work	161
7.6	TELEFREEK: Integrating collaboration support	162
7.6.1	Further work with TELEFREEK	163
7.7	General Conclusions	165
	References	166

A	mond Manual page	184
B	Mona: User guide	187
B.1	Introduction	187
B.2	Installing <i>Mona</i>	187
B.3	Getting started	188
B.3.1	<i>Mona's</i> inbox	188
B.3.2	Sizing window fields	189
B.3.3	Iconifying the inbox	189
B.4	Reading mail	189
B.4.1	Header information	190
B.4.2	Saving a message	190
B.4.3	Printing a message	191
B.4.4	Closing a message	191
B.4.5	Archiving messages	191
B.5	Sending messages	191
B.5.1	Addressing conventions	192
B.5.2	Message body	193
B.5.3	Dispatching (multiple) messages	193
B.5.4	Replying	193
B.5.5	Forwarding	193
B.5.6	Sending and including files	193
B.5.7	Spell checking	194
B.5.8	User-preferences when sending	194
B.6	Reminders in <i>Mona</i>	195
B.6.1	Posting a reminder	195
B.6.2	Reviewing (upcoming) reminders	196
B.6.3	Converting normal mail into reminders	197
B.6.4	Changing reminders	198
B.7	Mail management	198
B.7.1	Mail in context	198
B.7.2	Conversational webs	201
B.7.3	Modifying the web	203
B.7.4	Sharing web views	205
B.7.5	Changing the inbox file	205
B.7.6	Searching the archive	205
B.7.7	Moving mail to the inbox	206
B.7.8	Deleting messages	207
B.7.9	Editing messages in the archive	208
B.8	Avoiding window clutter	209
B.8.1	Closing and iconizing windows	209
B.8.2	Raising and hiding windows	209
B.9	Quitting <i>Mona</i>	210
B.10	Common problems	210
B.10.1	My typing doesn't appear anywhere	210
B.10.2	The mail-window doesn't appear when I request it	210
B.10.3	Saving and reading mail messages doesn't work	210

B.10.4	Absent file error	210
B.11	<i>Mona's</i> files and directories	210
B.12	<i>Mona</i> text editing functions	212
C	TELEFREEK: User guide	214
C.1	Introduction	214
C.2	Installing TELEFREEK	214
C.3	Getting started	215
C.3.1	TELEFREEK's main window	215
C.3.2	Iconifying TELEFREEK	216
C.3.3	Quitting TELEFREEK	216
C.4	Awareness of who is about	217
C.4.1	User Filters	217
C.4.2	Tell me when he/she/they arrive	217
C.5	Queries about people	218
C.5.1	Information about those on the community list	219
C.5.2	Information queries through the button field	219
C.6	Establishing communications	221
C.7	Extending and customising TELEFREEK	222
C.7.1	User Filters	222
C.7.2	Button utilities	223
C.7.3	Event monitors	227
C.8	TELEFREEK files and variables	229
C.9	TELEFREEK text editing functions	231

List of Figures

2.1	The “standard” CSCW taxonomy.	12
2.2	An <i>Information Lens</i> “Message announcement” semi-structured message template and pop-up menus associated with template fields.	20
2.3	Hierarchy of message templates in the <i>Information Lens</i>	21
2.4	An example rule/action pair for autonomous message filing.	22
2.5	State transition diagram of conversational moves in an IBIS.	26
2.6	State transition diagram of a conversation for action in which speaker A requests action from speaker B (bold nodes represent terminating states).	32
2.7	Interaction possibilities in low-technology conferencing systems.	39
2.8	A view-sharing architecture allowing dynamic registration, heterogeneous terminals, turn-taking, and meta-communication about the task/process.	42
2.9	Layout of the Capture Lab meeting, observation, and projection rooms.	50
2.10	SharedARK “Video Tunnel”.	56
2.11	Video fusion.	56
2.12	The architecture of ClearBoard-2.	57
3.1	The “vicious circle” of dependencies in groupware adoption.	67
3.2	Summary of effort in collaboration.	69
3.3	Subsets of communications and their formats.	75
3.4	Imposition of inconvenience at levels of cost/benefit disparity.	77
4.1	Transitions between the user’s ideal working methods and those supported by work support tools.	85
4.2	Comparative benefits with guidance-dependent and guidance-free strategies	97
4.3	Comparative benefits with the equal opportunity strategy	99
4.4	The “vicious circle” of dependencies in groupware adoption.	108
5.1	<i>HyperCommitments</i> file management level.	113
5.2	<i>Mona</i> ’s inbox.	117
5.3	A standard mail item in <i>Mona</i>	118
5.4	A reminder mail item.	119
5.5	<i>Mona</i> ’s search template.	120
5.6	An automatically inferred conversational web.	121
5.7	User interface to the Unix mail command.	126
6.1	The Hypercard prototype of TELEFREEK.	139
6.2	The main TELEFREEK window.	140
6.3	Modifying the filters menu.	142

6.4	Editing the filters form modifies the filters menu.	142
6.5	TELEFREEK icons.	143
6.6	Adding a “call library” function to TELEFREEK’s media form.	146
B.1	The main <i>Mona</i> window.	188
B.2	<i>Mona</i> ’s inbox icons.	189
B.3	A mail message window.	190
B.4	The Save message dialog box.	191
B.5	Selection from a hierarchical menu.	192
B.6	Spelling correction window.	194
B.7	A reminder message in <i>Mona</i>	196
B.8	Selecting All reminders.	197
B.9	The reminder period-entry window.	197
B.10	Assigning an activation date to a normal mail item.	198
B.11	An automatically inferred conversational web.	199
B.12	A web node.	201
B.13	Web-node menus.	202
B.14	Requesting a named conversation.	203
B.15	The web modification “spider-cursor”.	204
B.16	<i>Mona</i> ’s search template.	206
B.17	A text-edit window.	208
B.18	<i>Mona</i> ’s window icons.	209
C.1	The main TELEFREEK window	215
C.2	TELEFREEK’s icons	216
C.3	A typical user filter menu: releasing the mouse button at this point would select the C Experts group	217
C.4	The ambush users pop-up window	218
C.5	The Named User pop-up dialogue box	219
C.6	Editing the <code>.mailias</code> file	221
C.7	The <code>.filters_file</code> edit window.	222
C.8	Modifying the filters menu	223
C.9	Adding a “call library” function to TELEFREEK’s <code>.button_utils</code> file	224

List of Tables

2.1	Systems demonstrating (but not necessarily limited to) message filtering. . . .	18
2.2	Message linking, argumentation, and passive conversation systems.	25
2.3	Active coordination support groupware and toolkits—continued next page. . .	29
2.4	Active coordination support groupware and toolkits	30
2.5	A selection of collaborative writing applications	37
2.6	Selection of real-time conferencing systems and toolkits—continued next page.	43
2.7	Selection of real-time conferencing systems and toolkits.	44
2.8	Selection of computer-supported meeting room environments, and a review of Group Decision Support Systems.	46
2.9	Summary of “seamless interaction” groupware that use video technology. . . .	54
5.1	Previous by same rule.	122
5.2	Next by same rule.	122
5.3	Cause heuristic.	123
5.4	Response heuristic.	123
6.1	An example extension to the functionality of TELEFREEK.	145
6.2	A typical “synchronicity scenario”.	147
6.3	Human factors affecting selection of people and mechanisms in communication.	148
7.1	Outline of a scheme for extending TELEFREEK to the Internet community. . . .	164

Declaration

The need for groupware design principles was recognised during collaboration with Steve Jones, a fellow research student at the University of Stirling. The general aims of the principles, and three of the four principle titles, developed from our joint efforts in which it would be impossible to distinguish each individual's input. The research in chapter 3, forming the foundation for the principles, was carried out solely by myself, and presentation of the principles, strategies, and examples in chapter 4 is entirely my own work. It should, however, be noted that I consulted extensively with Jones while writing a paper presenting these principles (Cockburn & Jones, 1992). A further paper, co-authored with Jones, is in press (Jones & Cockburn, 1993); it describes the principles' use in the development of MILO, a collaborative authoring system implemented by Jones (Jones, 1992a; Jones, 1992b).

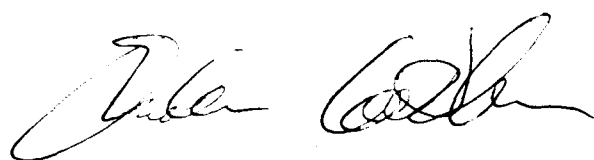
Work on the "Reflexive Perspective" of CSCW, described in chapter 4, pursues a similar line to that of Thimbleby *et al* (1990). My work on "Reflexive CSCW" has been published in (Cockburn, 1991c), and in a paper co-authored with my academic supervisor (Cockburn & Thimbleby, 1991).

Mona, the system described in chapter 5, and its prototypes were developed solely by myself. Its description in chapter 5 is original, but there will be overlaps with other publications describing various aspects of the system and its development (Cockburn & Thimbleby, 1992a; Cockburn & Thimbleby, 1992b; Cockburn & Thimbleby, 1993).

The HyperCard prototype of TELEFREEK, described in chapter 6, was developed and demonstrated by myself at the Department of Computer Science and at the Knowledge Science Institute, both at the University of Calgary, Alberta. The initial motivation for TELEFREEK arose from discussions with Dr Greenberg at Calgary. The work of chapter 6 has been used to produce a co-authored technical report at Calgary (Cockburn & Greenberg, 1993).

The HyperCard prototyping experiences described in chapters 5 and 6 led to my involvement in a co-authored paper (Thimbleby *et al.*, 1992) with HCI practitioners who also found HyperCard rewarding and *frustrating*.

Appendices B and C, the user guides to *Mona* and TELEFREEK, are available as technical reports at the University of Stirling (Cockburn, 1992; Cockburn, 1993).



Acknowledgements

Many people have contributed towards the content and existence of this thesis in many ways. I can only mention a few here; my apologies to those who are omitted.

The Isle of Man Education Department has presided over my academic career for more than two decades. I am grateful to schools and organisations on the Isle of Man for their support.

My fellow computer science postgraduate students at the University of Stirling have been excellent companions. A more supportive group would be hard to encounter. Of this group, Steve Jones deserves particular mention: our efforts at collaborative writing and our investigations into meeting environments have taken us from pub, to coffee bar, to Surfers' Paradise!

Dr Saul Greenberg at the University of Calgary deserves special thanks. During my two month visit to Calgary his enthusiasm, energy, and creativity stimulated an entirely new direction in my research. I do, however, have reservations about his sense of humour: rolling visiting research students down glaciers is not my idea of "funny."

The greatest academic influence throughout my University career has been the inspiration provided by my academic supervisor, Professor Harold Thimbleby. I feel extremely fortunate to have benefited from his inventiveness, ideas, and motivation.

Not all of my studentship has been spent pondering CSCW and groupware. Breaks in mountain regions have renewed research motivation when it has flagged. Iain Small has tolerated and spurred even the most bizarre of climbing escapades, and has kept us safe through them... Cheers Iain.

Others that I must thank include the computer officers, Catherine, Graham, and Sam, for enduring my multifarious pleas. I am deeply indebted to Dr Tom Kane for his highly constructive comments on an early thesis draft, and for his all-round support. Friends and family have, naturally, been supportive, and in particular I must thank Daniel, Eleanor, Poss, and Tom for tolerating my seclusion "up North". The one person above all others who has encouraged all my projects is Mum. It would be impossible to sufficiently thank her for the love and support.

Chapter 1

Introduction

“Let those who wish to communicate any matter of pressing importance to each other by fire-signals prepare two earthenware vessels of exactly equal size both as to diameter and depth. Let the depth be three cubits, the diameter one...”

The Histories of Polybius, 2nd Century B.C.¹

Throughout history humans have applied their available resources and technology to the purpose of communication. Today’s technology is no exception. Worldwide telecommunication networks have reduced the possible turnaround time for intercontinental information exchange from weeks, at the turn of the century, to milliseconds—distance is no longer a substantial barrier to rapid interaction. Computer technology allows people to store and review their thoughts of the previous day, week, month or year—the susceptibility of individual human memories to the passing of time has been reduced.

Computers, then, can allow communication with our own thoughts across time, and networks enable virtually instant access to millions worldwide. By combining these abilities physical separation, time-zones, forgetting, and a plethora of associated communication problems need no longer impede human capabilities for working together: colleagues in separate countries could discuss business decisions in real time; international time-zones could be used to advantage, allowing twenty-four hour work during normal office hours. The potential is bounded by our imagination. Such are the aims of researchers in the field of Computer Supported Cooperative Work (CSCW).

¹Improvements to communication by fire-signals proposed by Aeneas Tactitus, extracted from “The Histories of Polybius”, Book X, Chapter 44. (Shuckburgh, 1889).

Computer support for teams of people working together is not a new idea (Engelbart, 1963) (reprinted in Greif, 1988a); indeed the currently prevalent computer supported working environment, the Personal Computer, emerged to largely override multi-user computing communities established on mainframe machines in the 1960s and 1970s. In the middle of the 1980s, deficiencies in support for collaborative work (within which Personal Computing is executed) were addressed by an increasing number of computer scientists. They adopted the term “groupware” to describe computer systems specifically tackling the problems of people working together. The banner term “Computer Supported Cooperative Work” broadens the scope of groupware research, and includes studies of social, psychological, and other aspects of collaborative work that are relevant to the enhancement of collaborative work through modern technology.

The first CSCW conference took place in Austin, Texas in 1986. In the six intervening years since then, CSCW has become an established and substantial research domain with academic journals, conferences, books, and research grants dedicated to it, and many experimental systems have been developed with varying degrees of optimism for marketplace success. Yet, while research interest has flourished, there has been a notable lack of success in bringing groupware out of the research lab.

This thesis is concerned with promoting the success of groupware systems. It notes the problems groupware encounters, provides design principles and practical strategies to guide designers round these pitfalls, and demonstrates the principles use in two groupware applications.

1.1 Undefining CSCW

It would be reasonable to expect a thesis on CSCW to begin with a detailed definition of the letters making up the acronym. To attempt to do so consumes thousands of words (Jones, 1990; Bannon & Schmidt, 1989), and although thought provoking it is a largely self-serving, subjective, and temporally relevant exercise that risks confusing the issue (Bowers, 1992).

The research domain of CSCW is highly malleable and dynamic: essentially it is an umbrella term, constantly adjusting to encapsulate the work executed under its auspices. However, its theme may be summarised as:

the study and support of human activities (typically) using modern technology.

This apparently excessively general statement is as precise as possible without knowingly excluding some CSCW research. Now to dissect it:

the study — several disparate research disciplines contribute to CSCW, including computer scientists, sociologists, ethnographers, linguists, and psychologists. Not all are directly concerned with the role of modern technology (hence the word “typically” in brackets), instead they wish to better understand how people work, and work together. Their findings might, however, be applied to modern technology;

and support — a dangerously specific word that would be better replaced by “augmentation, enhancement, assistance, and support”. The aim of systems resulting from CSCW research (groupware) is not only to *support* existing human activities, but also to improve them, widen their scope, provide new facilities, and to assist in their execution;

of human activities — by avoiding a statement on plurality of those involved it is possible to circumvent early assumptions about group or individual facilitation.² “Activities” similarly avoids constraining the types of “support” offered;

using — the mechanisms employed by CSCW systems not only mediate human activities, they are also *used* for collective benefit. Facilities enabled by computer processing are exploited in the enhancement of group tasks;

modern technology — this phrase overcomes restrictions to computer capabilities. Various new forms of modern technology are drawn on by CSCW in support of human activities, such as multi-media, video, and active badges (see section 2.5.3). Although computer processing is typically required to enable these technologies, its role in the overall functionality may be a subsidiary one.

1.2 Defining groupware

Groupware is the primary concern of this thesis. Its relationship to CSCW is that of a more tangible sibling.

²A recurring theme in this thesis is the avoidance of tunnel vision and pre-conceived notions of what CSCW and groupware do—some of which are derived from taxonomies and classifications of CSCW (see section 2.1).

The term *groupware* first appeared in (Johnson-Lentz & Johnson-Lentz, 1982) in which it described computer systems *and* their related social group processes; as such, the terms *groupware* and *CSCW* are similar. A less broad definition is currently prevalent; it restricts groupware research to actual systems. Ellis *et al* (1991) suggest that:

“groupware be viewed as the class of applications, for small groups and for organisations, arising from the merging of computers and large information bases and communication technology.”, page-39.

While this perspective of groupware is a useful introduction, they go on to provide a specific definition of groupware:

“computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.”, page-40.

Although summarising *almost* all groupware, this (second) definition excludes some applications that are accepted as being “groupware”. It is therefore unsuitable as an *a priori* definition.

The groupware definition required at this stage is the lowest common denominator. As groupware is frequently the systemic result of CSCW research, it is natural that its definition should be a system oriented variant of the permissive CSCW definition above:

the support of human activities using modern technology.

The components of this broad definition are discussed above. It must, again, be stressed that this definition, like that of CSCW, is intentionally wide-ranging. Much of this thesis is dedicated to overcoming entrenched opinions that have been consolidated by *a priori* assumptions about groupware, CSCW, and their ambitions.

1.3 Principles for groupware design

In 1983, Donald Norman (reprinted in Baecker and Buxton, 1987) argued for “more fundamental approaches to the study of human-computer technology”, page 501. He alerted human-computer interaction (HCI) researchers to the “tar pits and sirens of technology”, referring to the temptations of system development, and the self-serving enticements of new technology.

His sentiments—that fundamental principles should be used to “broaden our views, sharpen our methods, and avoid temptation”—echo those of the development principles for the Xerox Star user interface (Smith *et al.*, 1982).³

User interface design is still a combination of art, science, and engineering (Thimbleby & Greenberg, 1991; Rettig, 1992), but the level of formality available in its execution, and the awareness of its developmental pitfalls have been increased by research and experience since Norman’s advice.

CSCW research is now in a similar situation to that of HCI when Norman argued for fundamental HCI principles. CSCW can be viewed as research territory into which exploration has only just begun, rife with uncharted tar pits and sirens—the problems of HCI’s youth are being re-visited. Technology driven research focuses attention on communication network facilities that are, as yet, largely unavailable and arguably inappropriate. The next-system trap draws designers back into implementation, and the lessons from their experiences are unrecorded and consequently unavailable for the benefit of future developers. CSCW developers “in the know” have the benefit of folklore-like design issues (Dourish, 1992b) which permeate through local research communities. However, excitement in the research field and developmental pressures leave little time for consideration and recording of widely applicable design principles that would broaden access to the lessons of design.

The principles presented in this thesis provide system designers with a guiding “chart”, noting the relevant pitfalls, problems, and barriers to the development of successful cooperative work support tools. Personal intuitions, and a “feel for the right way to do things” remain a major part of design (and the key factor distinguishing good designers from bad ones), but intuitions, regardless of their foundations, are fallible (Smith *et al.*, 1982; Erickson, 1989). These principles alert designers to groupware’s problems, and to the mis-guided intuitions encountered (some repeatedly) in collaboration support.

Naturally, like navigational charts, the refinement of principles is a continual process. The groupware design guidance presented here necessarily includes influences from the author’s own intuitions—drawn upon when other, less subjective, information sources were unavailable. Some of this (minimised) intuitive input will be disputable. This should not discredit the value

³Xerox Star was the direct fore-runner to the Apple Macintosh computer series, the success of which is largely attributable to its “ease of use.”

of the principles as platforms for system development for the following reason. Evaluation of groupware is extremely difficult, time consuming, and unreliable—beyond the most elementary metrics, such as whether people “liked” a system or not, it is exceptionally difficult to establish objectively the positive and negative system aspects, and to trace them back to design issues. Design principles, however, allow a reverse-engineering approach to iterative improvement of collaborative support: if a system fails, the principles upon which it was developed are called into question. Using the navigation analogy once more, a ship sailing with an inaccurate chart can modify the chart through its discoveries (analogous to system evaluation); the benefits of the voyage then become available for future sea-farers. Setting sail with no chart is reserved for the first, the intrepid, and the foolish.

1.3.1 Origin of the principles

The groupware design principles forwarded in this thesis are derived from various information sources—including CSCW literature, research collaborations, and (appropriately) computer-mediated communication mechanisms—and from personal experiences in groupware development. The role of personal intuition and “feel for the right thing” has been kept to a minimum, but cannot be avoided entirely. First-hand experiences in prototyping and system building are necessary to fully appreciate the subtlety and complexity of (groupware) design issues (Norman, 1983).

Mona and TELEFREEK (the systems described in chapters 5 and 6 of the thesis) were developed in conjunction with the principles. Superficially, it may seem improper to claim that they *demonstrate* the groupware design principles given that they contributed to the principles advancement. Such developmental and exploratory techniques are not uncommon; indeed, Descartes in the 17th Century advocated such an approach: “...each truth discovered was a rule available in the discovery of subsequent ones”, translated in (Descartes, 1927). By prototyping the systems, and by encouraging user-feedback (participatory design), an iterative development cycle enhanced the evolutionary maturity of the systems *and* their motivating principles.

1.3.2 Scope of the principles

Researchers will continue to engage in ambitious projects, intentionally exploring new domains where existing guidelines are largely irrelevant. They must maintain the highest levels of professionalism to ensure that problems encountered and lessons learned are recorded for the benefit of others.

Existing technology and established communication networks, however, have the capacity to improve group work *now*. The design principles presented in this thesis provide a foundation for the development of successful groupware built on currently available technologies; they also accept our currently limited understanding of human collaborative processes.

The four principles impart advice and design questions that must be considered by all groupware designers. Within each principle, practical strategies for implementing its aims are provided.

While researchers carry out visionary experimentation for tomorrow's technology, some of the lessons must be laid down to guide those building groupware today. This thesis is largely dedicated to that purpose.

1.4 Thesis structure

This thesis provides a single cohesive argument for, and demonstration of, a particular approach to groupware development. There is also a degree of chapter independence: it is possible to read, for example, chapter 5 (about the groupware system *Mona*) without having progressed through chapters 1 to 4. Chapter independence enables efficient access to information relevant to CSCW practitioners with specific interests. The chapters with the strongest interdependence are chapters 3 (concerning the particular problems encountered by groupware) and chapter 4 (detailing the groupware design principles).

Chapter 2 reviews groupware and CSCW research, establishing a platform and context for the thesis. Concentrating on systemic aspects of CSCW, the review examines the range of groupware applications, their design, and implementation strategies. It explicitly notes the potential dangers of categorising groupware. Several tables are used to summarise and contrast the diverse range of groupware and the implementation strategies they employ. The chapter includes a review of recent work on “seamless interaction”

and “social presence” systems.

Chapter 3 examines the problems encountered by groupware. Many papers in CSCW literature have noted the lack of successful or popular groupware (indeed several have noted that users *hate* it), but few have given more than cursory attention to analysing the causes of failure—the most notable exception being (Grudin, 1988).

In this chapter, groupware’s lack of popularity is investigated, primarily in terms of the user’s perceived costs and benefits. Although initially applicable to *any* computer system, single-user or groupware, the analysis extends to illuminate severe problems encountered specifically by group support tools. The cost/benefit observations are demonstrated with respect to a large class of groupware applications, this being enhanced asynchronous messaging systems. *Mona*, the groupware system described in chapter 5, provides enhanced asynchronous messaging support while avoiding the failings observed and demonstrated in this chapter.

Chapter 4 draws on the observations made in chapter 3, and describes four groupware design principles. The principles promote the development of collaboration support systems with maximised potential for success. At a general level, they raise the questions that must be addressed by all groupware developers, and at a specific level, they provide practical strategies for achieving successful implementations. The first principle addresses the importance of personal appeal in groupware; the second attends to system-imposed user-requirements; the third examines the constraints imposed on users; and the fourth principle investigates groupware’s role within the wider collaborative work environment. Existing groupware systems exemplifying desired features are extracted from CSCW literature to demonstrate the principles application.

Chapter 5 describes *Mona*, an enhanced electronic mail (email) groupware system that demonstrates the groupware design principles. It provides, among other augmented facilities, automatic conversational context of incoming and outgoing email messages. The prototypes and investigations leading up to *Mona*’s development in the X Windows environment are described. It is contrasted with other conversation-based email applications, and *Mona*’s fundamentally different mechanisms for achieving user-support is attributed to the four groupware design principles.

Chapter 6 describes TELEFREEK, another groupware application, also developed under the guidance of the principles, but providing an entirely different type of collaborative support to that of *Mona*. TELEFREEK is concerned with users' requirements and desires in collaborative work. It offers an integrated, extensible, and customisable interface platform to a variety of communication resources, media, and information. Emphasising the use and potential of base-entry-level technology, TELEFREEK supports "social awareness" facilities for which previous groupware systems have required video technology and high-bandwidth networks. Like *Mona*, TELEFREEK was prototyped prior to development in X Windows, and the HyperCard prototype is described.

Insights gained during TELEFREEK's implementation, are combined with observations of related research, and are used to record a set of human factors affecting collaboration. These factors form a design foundation for further work with TELEFREEK and for other groupware applications providing a platform for initiating collaboration.

Chapter 7 summarises the thesis, draws general conclusions from the research, and provides directions for future work with the principles, *Mona*, and TELEFREEK.

1.5 Appendices

Appendix A is a Unix-style manual page for the `mind` reminder utility that was developed as a prototype for *Mona*. `Mnd`, and its forerunning prototype *HyperCommitments*, are both described in chapter 5.

Appendix B is a user-guide for *Mona*.

Appendix C is a user-guide for TELEFREEK.

The user-guides for *Mona* and TELEFREEK are written in accordance with Carroll's "Minimal Manual" recommendations (Carroll, 1990; Carroll *et al.*, 1987/1988; Draper & Oatley, 1990), consequently they focus on actual user tasks within the systems.

1.6 Summary of thesis contribution

The systems described in the thesis provide current research contributions to CSCW on their own merit. *Mona's* primary research contribution is derived from the alternative mechanisms for user-support that it demonstrates. It achieves user-support comparable to that of other groupware applications, but in contrast, it does so without a dependence on explicit user-requirements for additional structured information. *Mona's* attainment of “free” user-benefits should be the base-platform from which all enhanced email systems extend.⁴ TELEFREEK demonstrates that current network facilities are capable of supporting many of the social awareness facilities for which other systems have assumed dedicated high-bandwidth audio and video channels.

The focus of the thesis is on assisting groupware designers to produce successful systems. The four groupware design principles (providing this assistance) are developed from an investigation into the problems encountered by groupware users. This examination clarifies the relationship between factors affecting groupware's success, and extends earlier work on the causes of groupware failure.

The principles raise the questions that designers must address, they provide practical advice on overcoming groupware's problems, and they use examples drawn from the extensive research literature on groupware design experiences. Extracting design lessons from the diverse literature is, in itself, a noteworthy research contribution: many publications primarily describe a single system, but some additionally contribute significant items of development knowledge. Newcomers to the research field, and aspiring groupware developers are unlikely to survey the entire (and, to the uninitiated, overwhelming) array of literature prior to engaging in development. Condensing and honing groupware development experiences in design principles eases access to the lessons of research experience.

This thesis explicitly records developmental experiences with *Mona* and TELEFREEK (and their prototypes) and combines these lessons with observations that are extracted, adapted, and developed from research literature. The primary research role of this thesis is to generate accessible guidance for *successful* groupware design.

⁴*Mona* has been classified with research on intelligent user interfaces (Cockburn & Thimbleby, 1993), but the author has reservations about this use of the word “intelligent”.

Chapter 2

Overview of groupware systems and CSCW research

2.1 Introduction

Prior to examining the causes of groupware's lack of success, and proposing principles for improved design, it is essential to understand what groupware is, what work has been done, and to place it within its parent research field of Computer Supported Cooperative Work. The aim of this chapter is to impart such understanding.

This thesis addresses systemic issues—design, use, acceptance, and so on—consequently, this overview focuses on the systemic work of CSCW, usually termed “groupware”. The overview provides a foundation for the development of generic groupware design principles. To assist the reader, systems within each groupware category are summarised in tables.

2.1.1 Concerns on groupware classification

Reviewing the entire interdisciplinary range of CSCW research would be an exhausting and interminable undertaking, for both author and reader.¹ The primary concern of this thesis is

¹Those wishing to access *all* aspects of the research will find excellent introductions in (Greif, 1988a), (Greenberg, 1991b), and chapters 13 and 14 of (Dix *et al.*, 1993); the ACM SIGCHI/SIGOIS-sponsored CSCW conferences provide bi-annual definitions of the state of the art (ACM Press, 1988; ACM Press, 1990; ACM Press, 1992). A variety of other conferences, journals, and publications also relate relevant work with varying degrees of dedication to CSCW, these include: the bi-annual European Community conference on CSCW (EC-CSCW, 1989; Bannon *et al.*, 1991; EC-CSCW, 1993), the annual ACM conference on Human Factors on

groupware and its design, therefore, the overview presented in this chapter, concentrates on systems, their mechanisms, and design.

The majority of groupware systems developed to date are derived from, motivated by, or iterative improvements on a small set of groupware systems which first identified an area for user-support, or an un-tapped potential use of new technology. Consequently, there is a small set of groupware domains, each containing several systems (however, the distinction between these domains is becoming indistinct, particularly with the recent advent of research into integrated work environments, see section 2.5).

The “standard” classification dichotomies of CSCW

A common classification of groupware uses two dichotomies, shown in figure 2.1 (Ellis *et al.*, 1991; Rodden, 1991; Dix, 1992a). The first is based on the group’s physical proximity, whether they are co-located or dispersed. The second distinguishes between the nature of the communication channels, either synchronous or asynchronous.

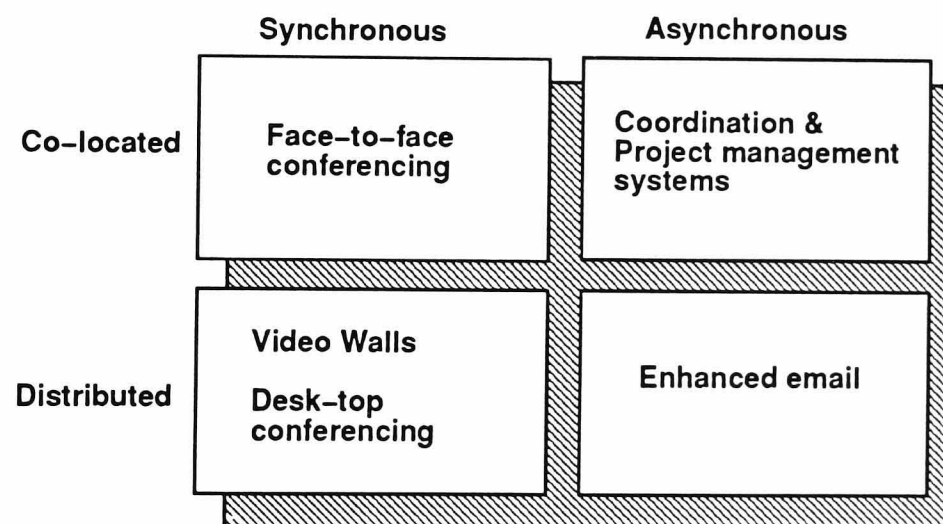


Figure 2.1: The “standard” CSCW taxonomy.

Although these distinctions accurately categorise the support provided by most groupware systems, they give no insight to the actual communication requirements of system users (Rhyne & Wolf, 1992).² Channels intended for synchronous use are frequently applied in an

Computing Systems (Robertson *et al.*, 1991; Bauersfeld *et al.*, 1992), and journals such as the new “Computer Supported Cooperative Work: An introductory Journal” published by Kluwer Academic. A extensive list of CSCW literature sources is given in (Greenberg, 1991a).

²In writing reviews of CSCW, author’s may be forced into adopting less than ideal schemes to adequately

asynchronous manner (telephone tag is a frustrating example), furthermore, the asynchronous capabilities of synchronous media are sometimes explicitly sought: for example, telephone answering machines (consider the message “I know you’re not there but when you get in remember to...”). In contrast to synchronous mechanisms being used in an asynchronous manner, asynchronous mechanisms can be used in a synchronous style: as displayed when email messages are rapidly exchanged. The proximity distinction in groupware is also suspect. It should be possible, at least in principle, to apply remote conferencing systems (such as group editors) in support of co-located people.

The danger of these distinctions lie in promoting the notion that systems necessarily support only one of these communication styles: one quadrant of the CSCW taxonomy, figure 2.1. Systems founded on this assumption are unlikely to allow the flexibility and integrability that is needed when group requirements shift between quadrants. Yet computers and modern telecommunication facilities have greatly increased flexibility to choose the pace of interaction: asynchronous media such as email can be transferred locally in seconds and globally within minutes. Often it is not the ability to support truly synchronous communication that influences the selection of particular telecommunication facilities, rather it is whether the message will be buffered on arrival (Sylvia Wilbur, private communication).³

Although it is true that groupware has almost exclusively enabled either synchronous or asynchronous mechanisms, there is a risk that designers will believe this is necessarily so, and fail to satisfy actual interaction requirements. Designers must be aware that their system will be used in ways they had not foreseen, and supporting or easing transitions between communication methods will enhance a system’s perceived value. Indeed, the better the system, the more likely it is to be used outside its planned domain.⁴ Ellis *et al* (1991) support this view with their statement “A comprehensive groupware system might best serve the needs of all the quadrants.”

describe the work of others. In using the time/space matrix shown in figure 2.1, Rodden (1991) states “While the division of cooperation into either synchronous or asynchronous is useful it still reflects a primarily technological perspective...”

³In a series of papers, Dix examines many related issues: the synchronous/asynchronous dichotomy in CSCW (Dix, 1992a), the communication and information requirements of distributed workers (Dix & Miles, 1992; Dix & Beale, 1992), and the relationship between communication channels and the pace of interaction (Dix, 1992c).

⁴This phenomenon, related to homeostasis, is discussed in Thimbleby (1990a).

As integrated approaches to CSCW proliferate, it seems likely that future reviews of CSCW research will focus on communication requirements rather than technological properties of the support offered: for an example, see (Bannon & Schmidt, 1989).

While classification schemes are useful (for comparisons, and for simplifying description) they have a drawback in that they can obscure alternatives, and inhibit creativity: they can establish technical or arbitrary distinctions where, preferably, none should exist.

2.1.2 Structure of the overview

As a consequence of the concerns on taxonomies raised above, it is with trepidation that a classification of groupware is presented here. The problem in reviewing groupware is describing systems that were built within classification structures without reinforcing the structure boundaries. However, as Rodden noted, classification schemes yield benefits (of clarity and conciseness) that outweigh their costs in terms of potential blindness to alternatives. In this overview, the intention is to establish a CSCW and groupware foundation, which, by the thesis conclusions, will be catechised by the groupware design principles, and by the alternatives demonstrated by the groupware applications described!

Four categories of groupware are discussed in this chapter. In general, they provide collaboration support through progressively more sophisticated communication technologies. This progression does not imply that latter categories are improvements on earlier ones: just that they are different.

Section 2.2 discusses the diverse class of systems that provide, and are based upon, computer messaging services. Typically these system use low-bandwidth channels, and are often restricted to textual communication. They can, in theory, provide extremely powerful facilities for cooperation and coordination.

The hypertext techniques used by some messaging systems lead to collaborative authoring systems, detailed in section 2.3. Many co-writing applications are founded on messaging systems and require prolonged turnaround times between the author's contributions; others aim to support the dynamic requirements of writers who work concurrently on their documents.

The concurrency problems encountered by real-time co-authoring groupware are also addressed by groupware meeting environments, described in section 2.4. Although much of the research on meeting environments has been on computer enhanced face-to-face meetings,

there is an increasing trend towards the inclusion of remote-proximity colleagues through multi-media.

Finally, “seamless interaction” environments and “social presence” groupware are examined in section 2.5. Seamless interaction systems, like many remote meeting environments, use high-bandwidth communication capabilities to integrate the tools and techniques used in everyday work. Social presence systems promote awareness of the whereabouts and availability of others for a variety of purposes, ranging from social browsing for casual interaction, to constantly tracking individual’s to ensure they are where they should be!

Other observations of groupware design, experiences, and development (for instance, describing particular groupware toolkits) are made where appropriate.

2.2 Messaging systems

Messaging systems have evolved from rudimentary electronic mail (email) facilities that allowed text messages to be sent between users on the same multi-user machine. Wide area networks (Quarterman, 1989), and the advent of international messaging standards, such as RFC822 (Crocker, 1982) and more recently X.400 (CCITT, 1987; Chilton, 1989; Kille, 1991), have matured and stabilised sufficiently for messages to be reliably sent nationally in minutes, and globally in little longer.

Within this discussion, groupware messaging systems are viewed as those extending or using email facilities for user-support: they display a common property in that messages are sent person to person(s). Although there may be sophisticated mechanisms for addressing, filtering, copying, processing, and selecting messages, when reaching their destination messages are deposited in personal mailboxes. Many computer conferencing systems also use computer messaging facilities, but these typically use a single shared information store rather than personal distribution of messages. Computer conferencing systems will be discussed with meeting environments in section 2.4.

In it’s ubiquitous form, email allows users with network access to send textual messages to one another. Any message may be sent to any number of recipients, and aliases and groups can be set up allowing messages that are addressed to a single name to be widely distributed (for example, the UK CSCW interest group, named `uk-csw`). Although most prolific in academic networks, email is becoming more common in the business world, and

many PC and Apple Macintosh mail systems are becoming available (Collin *et al.*, 1992). Email's restriction to textual information exchange is also diminishing: protocols such as X.400 explicitly allow messages to include multi-media (graphics, audio, Fax, and so on), as demonstrated by applications such as NeXTmail (NeXT Computer, 1990).

Naturally, as more people use email the number of messages increases, potentially exponentially as each sender can address numerous recipients. Email is therefore prone to overloading its users with information (Palme, 1984; Hiltz & Turoff, 1985; Holleran & Haller, 1990). The first, of four, messaging systems sub-classes described below answers the call "Who will save the receivers from drowning in the rising tide of information...?" (Denning, 1982), page-164. These "Message filtering" systems (described in section 2.2.1) address the inequality of email's assistance for those sending messages, and those receiving them. By returning some control to the receivers, and by restricting the distribution of messages, these systems reduce information overload.

The second message system sub-class improves email management in an augmentative way (rather than the curative approach of filtering systems). These "passive conversation" systems, described in section 2.2.2 enhance email's use as a medium for collaborative and coordinated work. Much of the work in this category is on presenting relationships between messages: the problem is summarized by Flores *et al.*, (1988)

"The range and quality of information readily accessible via the computer appears to have temporarily outpaced the growth of new roles and institutions for handling information. ... The management of information becomes an additional task—a burden, not a support", page-159.

"Passive conversation" and message relationship systems improve information access in an attempt to keep pace with expanding data storage. Their facilities are typically directed at the capture of decision deliberation.

"Active coordination" systems, the third sub-class of messaging systems, are examined in section 2.2.3. They attempt to monitor and drive the advancement of coordinated work projects.

The technique of "semi-structuring" messages, used to support most of the systems in the categories above, was found to be more powerful than expected: this unanticipated value is reflected in the title of the paper "Semi-structured messages are *surprisingly* useful ... "

(my emphasis) (Malone *et al.*, 1988). The fourth, and final, sub-class use semi-structured mechanisms as toolkits for the development of a wide range of message based groupware applications.

This review of messaging systems focuses on recent work, carried out under the banners of “CSCW” and “groupware”. For an review of the early work in messaging systems, see (Wilson, 1987; Wilson, 1988).

2.2.1 Filtering systems

The problems of information overload, and the paucity of email management were briefly introduced above. Many systems have provided message management facilities such as filtering, prioritizing, and archiving in an attempt to reduce these difficulties, and several are comparatively summarised in table 2.1.

In this section, the focus is on the *Information Lens* (Malone *et al.*, 1988; Crowston & Malone, 1988; Mackay *et al.*, 1989; Malone *et al.*, 1987), and its related research at MIT’s Sloane School of Management. It provides a clear demonstration of the techniques typically used in message filtering. Describing *Information Lens* also serves as a basis for the discussion of its second and third generation implementations (the Object Lens and OVAL), described in section 2.2.4.

Having described the *Information Lens* in detail, several alternative filtering schemes that are adopted by other message handling systems are detailed.

The Information Lens

Prior to and during the development of several systems that enhance email management, Malone and colleagues at MIT carried out studies on email use (Mackay, 1988; Brobst *et al.*, 1986), and on general organisational design issues (Malone, 1983; Crowston *et al.*, 1987–1988). Mackay’s study of frequent email users finds that, far from being just a communication system, email is used for a variety of information, time, and task management activities. Three extreme categories of email users are identified:

1. Prioritizers—who use email as a time management tool. They want to assign priorities to items such as “urgent”, “when I’ve got time”, “delete”, and so on.

System	Technique	Notes
Information Lens (Malone <i>et al.</i> , 1987; Malone <i>et al.</i> , 1988)	Semi-structured templates. Rule/Action pairs.	Message filtering, automatic responses, and response suggestions. See the text.
Andrew Message System (Borenstein & Thyberg, 1991; Ogura & Gillespie, 1991; Borenstein & Thyberg, 1988)	Template-based.	Wide ranging email and bulletin board enhancements. The system was modified to include automatic filtering capabilities having observed users carrying out manual filtering.
ISCREEN (Pollock, 1988)	Parsing “envelope” and message content. Rule/Action pairs.	Similar to Info. Lens, but avoids template structure. Instead it uses a flat rule structure, organisational hierarchy variables (eg To: MY-MANAGER), and rule conflict detection through user driven “What If” scenarios. See the text.
MAFIA (Lutz <i>et al.</i> , 1990)	Natural language parsing; keyword searches; type hierarchies	An alternative message filtering approach. Notes template mechanisms dependence on similar type hierarches. Automatically computes a semantic message representation.
LSI (Foltz, 1990; Foltz & Dumais, 1993)	Latent Semantic Indexing	Specifically directed at filtering NetNews articles. Automatically generates a “latent” semantic representation of items. Can retrieve similar or related items, or prioritize according to user’s estimation of the value of previous articles.
Answer Garden (Ackerman & Malone, 1990)	Semi-structured templates. AI techniques.	Query messages on specific topics are autonomously answered when the system has previously encountered them. Otherwise messages are routed to appropriate experts, and subsequent answers are added to the database.
(Kraut & Streeter, 1990)	Study of coordination in software development.	Examines facilities that are useful for people’s requirements in addressing their communications. Experimental systems provide features such as “find me an expert”, “does anybody know”, etc.

Table 2.1: Systems demonstrating (but not necessarily limited to) message filtering.

2. Archivers—who use email as an information management tool, and store information “just in case” it becomes relevant.
3. Manager/secretary teams—who use email for task management, the manager requesting the secretary to (independently) perform actions, such as “reply to” or “remind me”.

Mackay acknowledges that these extremes do not identify “typical” users, but notes that they demonstrate the flexibility and diversity of facilities required by enhanced email systems.

While Mackay’s email investigation is a general one, the study made by Brobst *et al* (1986) is specifically focused on problems of message *routing*: that is, ensuring that each message reaches all persons to whom it is valuable, and yet does not disturb or overload others. Their examination of information sharing in organisations identifies three prevalent techniques for establishing the value of information:

Cognitive filtering — information value is established by scanning its *content* for such things as keywords. This implies that automated filtering systems require a knowledge of each user’s interests and preferences.

Social filtering — depends on the recipient’s knowledge of the sender or author. For instance, messages from Joe Smith might be important because he is a manager. Another important variant of social filtering is provided by recommendations—information referred by a trusted colleague is likely to be relevant. Unmodified email systems attach the sender’s name to messages, but to autonomously exploit this information, filtering systems would have to record value ratings for numerous colleagues.

Economic filtering — estimates the value of information based on a cost/benefit analysis. Factors under scrutiny include the length of the message, current work pressure, the number of message recipients (widely distributed messages are usually of low personal value), and the cognitive and social estimations of value, detailed above.

The *Information Lens* is designed to support the flexibility and diverse functionality identified by Mackay, and to enhance email management through filtering schemes resembling those observed by Brobst *et al*. To do so it uses two primary components: a hierarchy of semi-structured message templates (or *frames*), and rule/action pairs.

Frames increase the amount of computer processable information carried in each message. When sending email, *Information Lens* users are encouraged to select a frame closely matching the topic of their message, and then to fill in fields contained in the frame: figure 2.2, taken from (Crowston & Malone, 1988) page-268, shows a typical message frame. The system's ability to execute appropriate actions for the message receivers' increases when sender's select highly specific frame, and when most or all the template's fields are accurately filled in: figure 2.3, taken from (Malone *et al.*, 1988), page-323 illustrates a hierarchy of semi-structured message templates.

Deliver		Cancel
Meeting Announcement		
To: Anyone		
From: Malone		
cc:		
Subject:		
Topic:		
Day:		
Meeting Date:		
Time:		
Place:		
Text:		

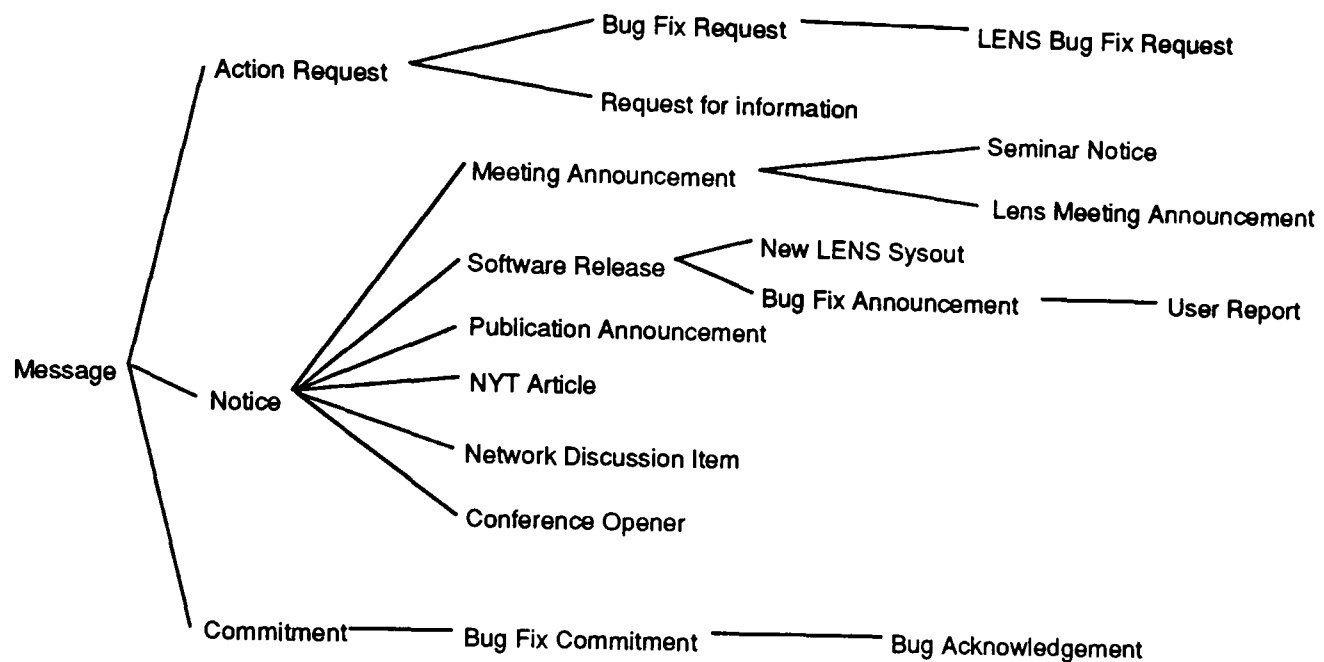
Place:
E53-301
E40-298
E52-598
Faculty club
Default
Explanation
Alternatives

Figure 2.2: An *Information Lens* "Message announcement" semi-structured message template and pop-up menus associated with template fields.

Information Lens allows users' to define rule/action pairs to customise their email management. The structural properties of incoming messages (template type, and field contents) are examined by each receiver's rules, established as IF condition THEN action pairs: see figure 2.4, taken from (Malone *et al.*, 1988), page-319. When rule conditions are met, the stated actions are executed. Actions supported by the *Information Lens* are:

1. MOVE TO: folder-name — moving the message to a specified folder. For example, a rule/action pair might be


```
IF Message type: Action request
Action deadline: Today, tomorrow
THEN Move To: Urgent;
```

Most generic templates are at the left of the figure. Progressively more specific to the right. Template fields are inherited through the hierarchy.

Figure 2.3: Hierarchy of message templates in the *Information Lens*.

2. **DELETE** — deletes the message;
3. **RESEND: user-name** — forwards the message to the named user(s);
4. **SHOW** — forwards the message from the central “Anyone” server to the user’s personal mailbox (see below);
5. **SET** — allows fields within the message template to be set to specified values, possibly triggering further rule/action pairs;
6. **LISP code** — allows more sophisticated actions to be executed through named LISP functions.

To supplement the facilities available through personalised filtering agents, *Information Lens* also supports an “Anyone” mail server. Messages sent to “Anyone” are collected in a central server, from where they can be selected by personal filters and forwarded to personal mailboxes. For instance, rule/action filters might be used to scan for messages on specialist topics. If found, the messages are forwarded to personal mailbox using the **SEND** action, described above.

Save		Cancel
Rule Editor		
Name		

IF		
Subject:	To:	
Default	From:	
Explanation	cc:	Subject: CISR Lunch
Alternatives	Subject:	Topic:
	Message type:	Text:
	Ignore After:	Day:
	Meeting Date:	Time:
	Place:	Characteristic:.....
	THEN	
	Move To:	CISR Lunch

Figure 2.4: An example rule/action pair for autonomous message filing.

Finally, *Information Lens* can “intelligently” suggest suitable template types for replying to messages; furthermore, it can suggest appropriate actions. For example, when answering a **Bug Fix Request**, the system will offer three alternative message types: **Bug Fix Commitment**, **Request for Information**, and **Other**.

Comparative filtering approaches

The problems of information overload, and the deficiencies of email management are well known (Hiltz & Turoff, 1985; Malone *et al.*, 1987; Mackay, 1988). The potential of semi-structured schemes for easing these problems is reflected in the numerous systems adopting similar techniques for filtering and for more ambitious forms of user support (second and third generation *Lens* systems, called *Object Lens* and *OVAL*, are described in section 2.2.4).

An in depth study of the difficulties encountered by email management systems will be made in chapters 3 and 4. It is, however, worthwhile briefly discussing some alternatives to the *Information Lens*'s filtering mechanisms.

ISCREEN (Pollock, 1988) provides similar email management to that of *Information Lens*, but does so through markedly different techniques. Pollock notes that semi-structured template techniques fail when messages arrive from sources using different or no structuring mechanisms. ISCREEN's rule/action pairs are therefore triggered by information sources inherently transferred in email communication: these being the email “envelope” (or header), and the

message text itself, which is searched for keywords. Another fundamental difference between ISCREEN and *Information Lens* is ISCREEN's flat, rather than hierarchical, rule structure. ISCREEN's target user-base includes non-regular computer users (such as senior managers), and it was believed that a hierarchical rule structure would be "too complex for the user group" (Pollock, 1988), page-234. Although the flat structure is potentially prone to conflicting rule definition, ISCREEN maintains consistency through a "Conflict Detection and Explanation" mechanism. Having specified a new set of rules users can simulate a set of "What If" scenarios to experiment with, and gain confidence in, the rules' operation. MAFIA (Lutz *et al.*, 1990) similarly notes the negative affects of semi-structuring, and achieves message filtering through natural language parsing and keyword searches of message text. Other systems supporting filtering schemes are summarised in table 2.1.

Deterministic rule/action pairs (as displayed by *Information Lens*) fail to account for many of the economic filtering requirements discussed in (Brobst *et al.*, 1986), and mentioned above. When work pressure is low, users' may want to browse material that would normally be filtered out. Establishing deterministic rule/action pairs to achieve the multi-level priorities required for economic filtering, although possible, would be complex and time-consuming. A related issue is addressed in a learning interface for scheduling meetings (Kozierok & Maes, 1993) that automatically assigns confidence weightings to the inferences it makes: at certain thresholds, which the user can alter, the system can autonomously execute actions (when very confident), suggest actions (when quite confident), or remain silent until suggestions are requested (when unsure). This scheme could be adapted to assign priority weightings to email; for example, during periods of low pressure a Show Me threshold could be set at a low value, consequently increasing the number of messages displayed.

2.2.2 Passive conversation systems

Message filtering provides facilities that are essentially curative: it aims to overcome existing problems in information overload. The systems described under "passive conversation" have a subtly different approach to email management—they focus on augmenting and enhancing email use through a variety of schemes for meaningfully relating information. In describing his HyperMail system, Belew and Rentzepis (1990) underlines this intention: "to treat Email as a form of *literature*, worthy of the same preservation and augmentation that is typical of

traditional printed media.”, page-48.

A summary of message linking and conversation representation systems is given in table 2.2.

It must be stressed that there is a close relationship between many of the systems discussed under the separate “Messaging Systems” categories: “conversation” systems support filtering; “filtering” systems support active coordination (described in the next section), and so on. In this overview, each system’s categorisation largely depends on the emphasis of the literature describing it.

Passive conversation systems typically use an underlying formalism or language to represent the allowable state transitions in processes such as policy decisions, arguments, and conversations. The system described in greatest detail here is gIBIS. Several systems adopt similar approaches, and a brief comparison follows the description of gIBIS.

gIBIS

The capture and representation of design deliberation is the goal of gIBIS (Conklin & Begeman, 1988a; Conklin & Begeman, 1988b; Yakemovic & Conklin, 1990). It semi-autonomously records design problems, and the conversations around them: the alternative solutions proposed; the arguments and trade-off analysis between alternatives; the commitments made during issue resolution. A graphical representation of this process allows conversations to be browsed in order to guide future decisions, to determine outstanding commitments, and to recover the motivation and reasoning behind the resolution of previous issues.

gIBIS is founded on a model of design deliberation called *Issue Based Information Systems* (IBIS). Developed by Rittel (Kunz & Rittel, 1970), IBIS represents the conversation-like process used in the solution of highly complex problems. When resolving *Issues*, stakeholders take *Positions* which they support or object to through their *Arguments*. Figure 2.5, taken from (Conklin & Begeman, 1988a) page-305, shows the set of legal rhetorical moves in an IBIS.

gIBIS (*graphical* IBIS) supports the IBIS model through semi-structured message templates conceptually similar to those of the *Information Lens* (described above). Messages are intercepted by the system which records its IBIS type and context, allowing conversation (or

System	Technique	Notes
HyperMail (Belew & Rentzepis, 1990)	User-specified links.	User establishes links between messages for future use.
gIBIS (Conklin & Begeman, 1988a; Conklin & Begeman, 1988b; Yakemovic & Conklin, 1990)	State transition diagrams (Issue Based Information Systems). Semi-structured message templates.	Focuses on capturing, indexing, retrieving, and graphically representing informal design decisions. See the text.
Strudel (Shepherd <i>et al.</i> , 1990)	Semi-structured messages, production rules, and conversation management paradigms (state-transition diagrams).	Graphical representation of conversational progression similar to gIBIS, but using flexible conversation models.
WHAT (Hashim, 1991)	IBIS based.	Using IBIS representation schemes for argumentative writing.
SIBYL (Lee, 1990)	Decision representation language (state-transition diagram). Semi-structured templates.	Similar to gIBIS but claims a greater knowledge-based orientation.
COKES (Kaye & Karam, 1987)	Semi-structured templates. Rule/Action pairs. State transition diagrams.	A "knowledge-based office assistant". Aims to combine automation of routine tasks and assistance with complex ones.
<i>Mona</i> (Cockburn & Thimbleby, 1992a; Cockburn & Thimbleby, 1993). See chapter 5.	Conversation inferencing heuristics, and user-specified linking.	Infers conversational relationships independent of information explicitly provided by users. Graphical representation of conversation.
SYNVIEW (Lowe, 1985)	Explicit structures for representation of argument. User voting for ranking information value.	Aims to assist decisions and overcome information overload.

Table 2.2: Message linking, argumentation, and passive conversation systems.

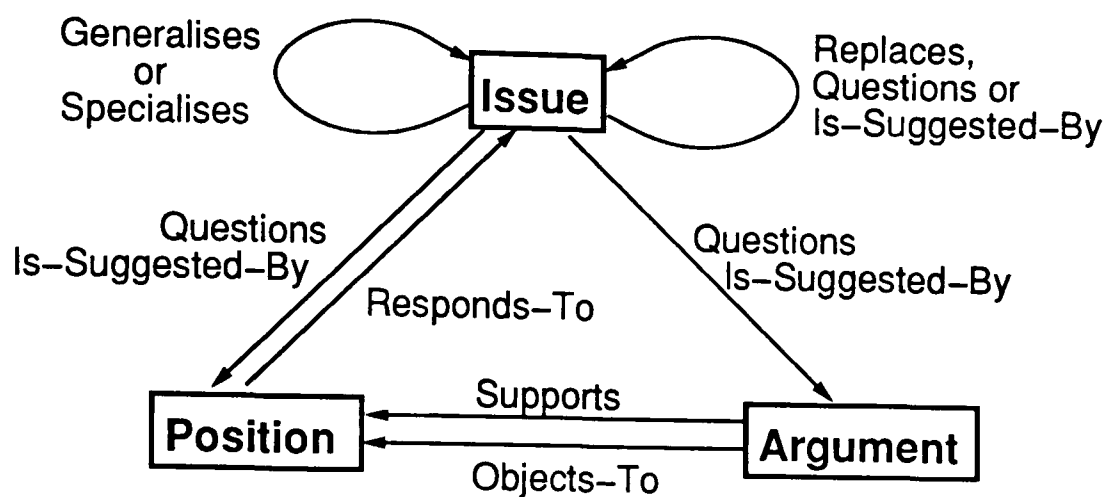


Figure 2.5: State transition diagram of conversational moves in an IBIS.

Issue) progress to be captured and represented in a Hypertext display.⁵ Users can simultaneously access the Hypertext representation, and gIBIS controls the locking schemes necessary to avoid concurrent updates. Only a single, central, view of the conversation is supported, forcing all users to a consensus; although such restrictions might be advantageous within gIBIS's domain (design decision support), the lack of personal views is likely to be unpopular in others (Greenberg, 1990b).

Other message linking systems

Several systems adopt similar approaches to gIBIS, using semi-structured templates, knowledge-bases, and simple argumentation models to capture message relationships (for a summary of these systems, see table 2.2). The ability of these systems to assess the context relating messages is dependent on message sender's selecting an appropriate template, and filling in relevant semi-structured information fields. These dependencies are a serious hindrance to their successful use, and are discussed in the following chapter.

Belew and Rentzepis (1990) note related problems in message filtering, observing that people are unlikely to be able to state beforehand what constitutes an interesting message. And yet, this *a priori* categorisation is precisely what users are required to explicitly encode in their message filtering rules. They therefore contend that the *a priori* knowledge recorded

⁵For an overview of Hypertext, see (Nielson, 1990b; Conklin, 1988; Nielson, 1990a), and for a discussion on the Hypertext problems encountered by systems such as gIBIS, see (Nielson, 1990a; Utting & Yankelovich, 1989).

in rule/action pairs should be supplemented with *post hoc* browsing aids. HyperMail (Belew & Rentzepis, 1990) supports user-explicit linking between messages, and allows a variety of conversational or keyword links to be constructed between messages. It unfortunately lacks a graphical browser which would enable multiple threads of conversation and interest to be viewed simultaneously.

Mona (Cockburn & Thimbleby, 1992a; Cockburn & Thimbleby, 1993; Cockburn, 1992), described in detail in chapter 5, uses fixed conversational heuristics, and user customisation facilities to enable conversation browsing without dependence on semi-structured techniques.

2.2.3 Active coordination systems

To varying degrees, many of the systems described above predict and suggest user actions: for instance, *Information Lens* and gIBIS recommend particular message types for response. The systems categorised under “active coordination” extend this “predict and suggest” role to “predict and execute”. Passive systems capture message relationships, present them, and allow users to manipulate, use, or ignore them as they see fit. In contrast, active systems can additionally track and maintain a *knowledge* of the status of commitments, prompt user actions through reminders, and, controversially, attribute blame to commitment defaulters.

The boundary between passive and active systems is indistinct. Both typically rely on semi-structured message techniques and a theory or model of conversation/coordination. The essential difference is the degree to which the model is used to act on behalf of users.

As these systems actively support users, they are critically dependent on correct use: it may be inconvenient if a passive system fails to autonomously capture a message’s conversational context, but it will be annoying if a similar failure in an active system causes redundant or mis-timed reminders, and worse if it blames the wrong people for commitment defaults.

The best known system of this category (perhaps of all groupware) is *The Coordinator* (Winograd, 1987; Winograd & Flores, 1986; Flores *et al.*, 1988). One of the earliest groupware systems, and certainly one of the earliest commercially released groupware applications (Action-Technologies, 1987; Winograd, 1988; Opper, 1988), *The Coordinator* has been the subject of many evaluative exercises (Carasik & Grantham, 1988; Bullen & Bennet, 1990; Grehan *et al.*, 1991) which unanimously condemned it. The causes and contributory factors in its lack of success are discussed in chapter 3.

The following section examines *The Coordinator's* mechanisms for active coordination and management of projects. A selection of systems providing active support are summarised in tables 2.3 and 2.4. For a further review, including the Chaos (De Cindio *et al.*, 1986) and COSMOS (Dollimore & Wilbur, 1991) projects, see (Wilson, 1988).

System	Technique	Notes
MONSTR (Cashman & Holt, 1980)	Protocol-driven. State-transition diagram of communication paths. Email.	Early forerunner to <i>The Coordinator</i> . Commitment capture. Highly constrains the possible communication paths, and aims to: “support the software maintenance process, rather than emphasising capabilities, tools or static structures” (Cashman & Holt, 1980), page-15.
<i>The Coordinator</i> (Winograd, 1987; Winograd & Flores, 1986; Action-Technologies, 1987; Flores <i>et al.</i> , 1988; Winograd, 1988)	Speech-Act Theory (Language/Action perspective). Semi-structured templates.	An early groupware system, widely studied, and providing MANY useful findings on the importance of social aspects in group work support. See the text.
E-MERGE (Leadbetter & Seeley, 1992)	As for <i>The Coordinator</i> .	Commitment based message tracking in an Apple Mac environment. Concentrates on interface issues, and additional email facilities.
CHAOS (De Cindio <i>et al.</i> , 1986)	As for <i>The Coordinator</i> . Additional organisational knowledge.	Commitment based message tracking. Collaborators are assigned roles in particular tasks to govern their permissions, status, and tasks.
SACT (Woo, 1990)	Refined Speech-Act Theory and knowledge-based “MOAP”’s (Multi-Organization Activity Processors).	Focus on <i>automating</i> organisational communication. Conceptual system description—not implemented.
COMTRAC (Koo, 1988)	Commitment-based protocols generated from a context free grammar.	Recording and reminding of inter-agent commitments. Agents need not be human. Similar aims to the Coordinator.
MailTrays (Rodden & Sommerville, 1991)	Object-oriented variation of <i>The Coordinator</i> ’s theme. Interaction mechanisms encoded into objects. Semi-structured email templates.	“Mailtray” metaphor for objects. Trays have associated guards (filters) and rule/action pairs for incoming messages, and similar out-going actions/guards. Graphical “conversation editor” shows tray (interaction) paths.
SAMPO (Auramaki <i>et al.</i> , 1988)	Speech-Act-based office Modelling aPprOach. Study of Speech-Act Theory.	Theory, examination, clarification, and representation schemes for Speech-Act-Theory.

Table 2.3: Active coordination support groupware and toolkits—continued next page.

System	Technique	Notes
Various calendaring and scheduling systems (Kincaid <i>et al.</i> , 1985; Kaplan <i>et al.</i> , 1990; Murphy, 1990; Greif & Sarin, 1987; Grehan <i>et al.</i> , 1991; Sathi <i>et al.</i> , 1988; Kozierok & Maes, 1993)	Various, but typically semi-structured messages.	Many systems, notably RTCAL (Greif & Sarin, 1987) and the “learning interface agent for scheduling meetings” (Kozierok & Maes, 1993). Several provide active coordination assistance in meeting schedulers.
Object Lens alias OVAL (Malone & Lai, 1988; Crowston & Malone, 1988; Lee & Malone, 1990; Malone <i>et al.</i> , 1992)	Object-oriented semi-structured templates. Rule/Action pairs. Hypertext Links.	A toolkit for the development of a variety of collaboration and coordination systems (discussed in section 2.2.4).

Table 2.4: Active coordination support groupware and toolkits

The Coordinator

The Coordinator project offers an alternative to the decision-making orientation of the two major branches of management systems. These branches are, first, *management information systems*, which tend to saturate users with information, and provide no evidence of improving resultant decisions (Kiesler *et al.*, 1985). Second, *decision support* (and expert) systems, which tend to induce early decisions before adequately considering all alternatives. Flores *et al.* (1988) observe that both these system types are dependent on a rationalistic approach to problem solving, in which office issues can be objectively observed and fully characterised. They contend that this approach is rarely appropriate in practice as management problems are usually ill-defined.

The Coordinator's perspective of cooperative work support contrasts directly with that usually maintained by office technology in which functions, tools, and artifacts are the core of collaboration support (Flores *et al.*, 1988; Winograd & Flores, 1986). Its alternative management support philosophy, founded on speech-act theory, is that language mediates action.

Speech-Act Theory. *The Coordinator's* “Language/Action perspective” is derived from Searle’s work on Speech-Act Theory (Searle, 1969; Searle, 1979)⁶ in which “speech acts”

⁶Searle’s work extends and clarifies that of Austin (1962), which was rife with self doubt. Austin admits of

are the basic units of communication. Although Speech-Act Theory provides categories for *all* communicative utterances, the essential speech-act in supporting cooperative work is the *illocutionary* act, that communicates intentions.⁷ Searle (1979) forwards a five point taxonomy of illocutionary acts:

1. assertives, state a condition (past, present, or future);
2. directives, attempt to get the recipient (hearer, or receiver) to do something;
3. commissives, commit the person making the statement to a future course of action;
4. expressives, communicate a psychological state about the situation (for example thanking, apologising, welcoming);
5. declaratives, change a state in reality (for example, declaring termination of employment).

A subsidiary use of explicit speech-act communication is the promotion of clear and succinct communications, as noted by MONSTR, a forerunner to *The Coordinator*:

“MONSTR addresses the problem of determining organizational responsibility for an activity by requiring that a person explicitly tell MONSTR that s/he accepts responsibility...”, (Cashman & Holt, 1980), page-11.

Several groupware projects have explicitly adopted speech-act theory to formalise the processes of work coordination, these include: Chaos (De Cindio *et al.*, 1986), E-MERGE (Leadbetter & Seeley, 1992), COSMOS (Dollimore & Wilbur, 1991), COMTRAC (Koo, 1988), and the system examined here, *The Coordinator*.

The Coordinator project generalises speech-act theory into *conversations for action*: “...the central coordinating structure for human organizations”, (Winograd, 1987), page-10. The finite state-transition network of *conversations for action* that is explicitly embodied in *The Coordinator* is shown in figure 2.6, taken from (Winograd & Flores, 1986), page-65. In addition to conversations for action, *The Coordinator* also supports explicit models for *conversations for clarification*, *conversations for possibilities*, and *conversations for orientation*.

some aspects “...I have not succeeded in making clear even to myself”, page-151.

⁷A recent and thorough review of Speech-Act Theory can be found in (Auramaki *et al.*, 1988).

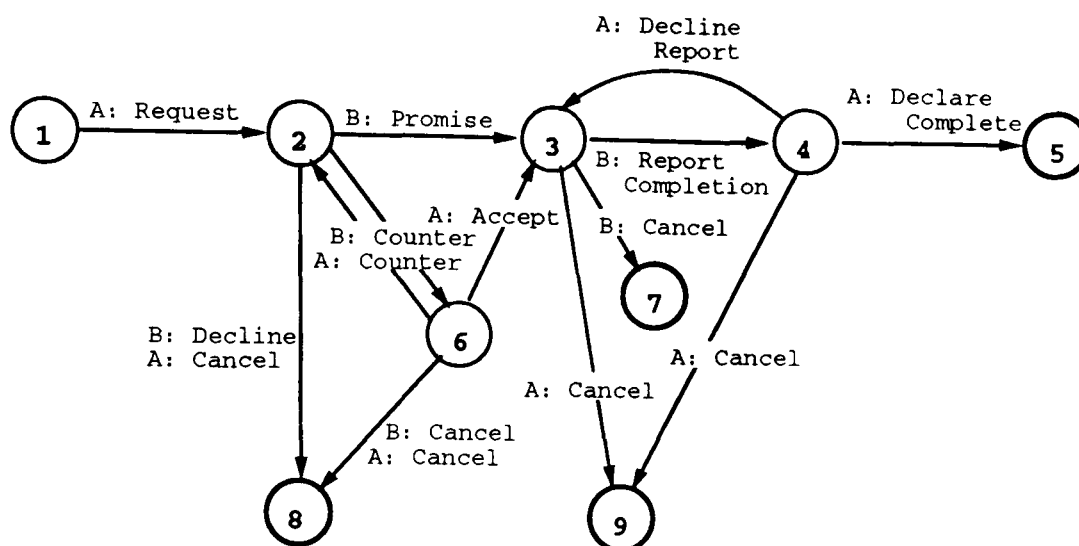


Figure 2.6: State transition diagram of a conversation for action in which speaker A requests action from speaker B (bold nodes represent terminating states).

The Coordinator's conversations for action not only capture conversational commitments, they are also “interpreted centrally to *drive* the conversation described”, (Rodden & Sommerville, 1991), page-165 (my emphasis). Messages are tracked, and their affect on the status of commitments is automatically noted: thus, if A commits to complete a project by a particular date, the system can provide timely reminders, observe who is responsible for delays, and (controversially) attribute blame.

The Coordinator combines tools for email, calendaring, scheduling, and word-processing, within its underlying conversation and management support. Literature advertising *The Coordinator* (Action-Technologies, 1987) claims widespread commercial use, improved work capacity and effectiveness, and enhanced recognition of group involvement.

The Coordinator has been the subject of many evaluative studies (Erickson, 1989; Carasik & Grantham, 1988; Henninger, 1991; Bullen & Bennet, 1990; Grehan *et al.*, 1991). Many faults are cited including its lack of flexibility and poor user interface, but its fundamental problem is maintaining a current knowledge of commitments. Although *The Coordinator* could manage project commitments if all communications were transmitted in a format accessible to it, this cannot be achieved without a great deal of additional work in structuring, reformatting, and duplicating communications. *The Coordinator* therefore encounters difficulties in providing sufficient personal benefit to warrant the costs (additional work).

The Coordinator is further criticised for ignoring and alienating the very social aspects of work that motivated its design: the designers state

“We design in a fundamental domain of social interaction”, (Flores *et al.*, 1988), page-171;

and yet evaluators note

“the Coordinator falls down, not because it has a formalised (textual) dimension, but because it has excluded, marginalised, and even illegitimised the ‘cultural’ dimension of conversation.”, (Robinson, 1989), cited in (Bannon & Schmidt, 1989), page-366.

The severe problems encountered by active coordination systems are investigated in chapter 3.

2.2.4 Toolkits for messaging systems

Many of the systems above are not restricted to the categories in which they are described: for example, Strudel (Shepherd *et al.*, 1990) and COSMOS (Dollimore & Wilbur, 1991) are configurable systems, enabling their support to be tailored for specific organisational requirements.

The system best exemplifying the power of message-based cooperation and coordination toolkits is the second generation *Information Lens*, called the Object Lens (Malone & Lai, 1988; Crowston & Malone, 1988), and recently renamed OVAL (Malone *et al.*, 1992). It supports four kinds of building blocks for the creation of cooperation applications:

1. Semi-structured **Objects**—representing organisational artifacts such as people, tasks, messages, and meetings. Objects are manipulated through a template-based interface, similar to the semi-structured templates of the *Information Lens* (see figure 2.2).
2. User customisable **Views**—summarise collections of objects, and allow object representation to be tailored. Views supported include tables, networks (hypertext), and calendars.
3. Rule-based **Agents**—perform autonomous tasks for users. Similar to the rule-action pairs of the *Information Lens* (see figure 2.4).

4. Links—representing relationships between objects. Links can be browsed through hypertext tables (Views).

The 1988 paper on the Object Lens (Malone & Lai, 1988) expounded its potential as a generic toolkit for cooperative work applications. Their 1992 paper (Malone *et al.*, 1992), though changing the name of the system to OVAL, substantiates this claim by demonstrating OVAL-based implementations of gIBIS, *The Coordinator* (although the OVAL implementation does not support the full range of *The Coordinator's* functionality), Lotus Notes (Lotus, 1989), and the *Information Lens*.

This demonstration of the OVAL toolkit's flexibility is extremely appealing. Some concerns, however, are worth noting.

- No mention is made of the amount of design effort required to replicate the restricted implementations in OVAL. OVAL's rule-action pairs (or agents) allow explicit sections of LISP code to be called when rule conditions are satisfied, and it must be assumed that this facility was used to replicate systems. No mention is made of the quantity of LISP code required.
- Implementation with OVAL will constrain designers to the facilities supported by the toolkit. This constraint is minimised by allowing explicit LISP code, but OVAL's limited interface components (or views) are likely to restrict interaction facilities.

Although these limitations may be substantial for some groupware applications, Malone *et al* note that toolkits such as OVAL can yield substantial advantages. Groupware implemented in OVAL will be highly integrable due to compatible object schemes and interface components: for example, messages processed by OVAL's replication of Lotus Notes could be autonomously reported to the OVAL-*Coordinator* through appropriate agents, and vice-versa. This capability holds potential for overcoming many of the compatibility problems that are encountered by conflicting semi-structured messaging schemes (Lee & Malone, 1990).

Toolkits such as OVAL enable, if nothing else, rapid prototyping of cooperation applications, admitting experimentation and easing studies into new areas of user support. Their potential is impressive, as is that of messaging systems in general. However, the semi-structured techniques almost unanimously adopted by messaging systems are dependent on explicit user-actions, and this dependence has severe implications for user-acceptance. The nature of these

implications are examined in the following chapter, and chapter 4 offers some alternatives. Chapter 5 describes *Mona*, a conversation-based email system that uses these alternatives to promote user-acceptance.

2.3 Co-authoring systems

Many of the systems described above improve information management through graphical representation schemes (for instance, showing networks of conversational progression). Typically, they link related information to ease browsing and directed searches. The term “hypertext” describes these schemes,⁸ and “hypermedia” broadens the scope of hypertext by incorporating diverse media including text, graphics, audio, and video.

Hypertext-like schemes for improving information management are not a new idea. In 1945 Vannevar Bush, reprinted in (Greif, 1988a), proposed the *Memex*, a system addressing the “massive task of making more accessible our bewildering store of knowledge.”, page-17. The conceptual *Memex* features similar functionality to current hypermedia applications, such as data trails, but did so using a micro-film database—Bush could not be expected to foresee the information storage revolution brought about by transistor technology.

Although hypertext is a useful interface mechanism for groupware, it does not, in itself, solve difficulties in collaboration or collaborative writing.⁹ Many of the issues addressed by co-authoring systems are applicable to a wide range of collaboration applications, they include:

1. **Access permissions and roles**—not all people involved in co-writing projects are of equal status: some will be authors who are ultimately responsible for document creation and modification; others may be referees who, although allowed to annotate and add to the document, may not be allowed to delete and change existing text. Quilt (Leland *et al.*, 1988), a co-authoring system, supports explicit user-roles of this nature.
2. **Concurrency**—co-authors may wish to modify their documents simultaneously, raising complications in maintaining “current” document versions. A variety of related issues arise:

⁸For an overview of Hypertext systems, see (Conklin, 1988).

⁹Unfortunately, hypertext often creates as many problems as it solves. Its rich information connections overwhelm users causing them to become “Lost in HyperSpace”. Some schemes for overcoming this limitation are discussed in (Nielson, 1990a; Thimbleby, 1991).

- Merging documents that have been independently modified off-line.¹⁰
 - Locking documents or document parts to avoid concurrent (on-line) access—for instance, see KMS (Yoder *et al.*, 1989).
 - Granularity (size) of the nodes (document sections) to be individually locked in avoiding concurrent access.
 - Schemes to maintain awareness of the actions of others, thus avoiding the necessity for locking schemes—see WYSIWIS in section 2.4.3, and GROVE (Ellis *et al.*, 1991).
3. **Views**—primary considerations are the mechanisms for representing the document, and whether document views are tailorable to individual preferences. MILO (Jones, 1992a; Jones, 1992b; Jones & Cockburn, 1993), like AUGMENT/NLS—Readings 2–5 of (Greif, 1988a), and (Engelbart & Lehtman, 1988)—encourages authors to focus on document structure rather than low-level word-processing facilities. VNS (Shipman III *et al.*, 1989) and Quilt (Leland *et al.*, 1988) ensure users are able to customise and tailor personal representations of documents while not restricting those available to others.
4. **Annotation**—annotation facilities are of particular importance in co-authoring systems. Many alternative annotation mechanisms have been examined, drawing extensively on various hypertext schemes. A recent addition to the plethora of metaphors investigated (including links, warm links, hot links, post-its, notecards, flags, and so on) is layers-of-acetate—see InterNote (Catlin *et al.*, 1989).

Selected systems supporting collaborative writing are shown in table 2.5. Further reading on hypertext systems (collaborative and personal) can be found in (Jones, 1990; Conklin, 1988).

2.4 Meeting Environments

There has already been substantial potential for overlap between the two groupware domains discussed: for example, WHAT (Hashim, 1991), described as a passive conversation messaging

¹⁰Liveware (Witten *et al.*, 1991; Jones, 1993) addresses these issues: “We use the term *Liveware* for the class of systems used to support asynchronous CSCW tasks subject to observational consistency.” (Thimbleby, 1990b), page 6/3.

System	Notes
InterNote (Catlin <i>et al.</i> , 1989)	Focuses on collaborative annotation of documents. Extends Hypertext linking beyond node traversal. Allows data transfer across links (called "warm" linking). Future work outlines a "layers of acetate" overlay model for annotation.
BLEND (Shackel, 1991; Shackel, 1987)	A base-communication-technology project initiated before the arrival of ubiquitous email. Focuses on the use of electronic communication networks as an aid to the many stages of writing (creating documents, submitting, storage, refereeing, and related communication issues).
MILO (Jones, 1992a; Jones, 1992b; Jones & Cockburn, 1993)	Attends to user requirements in writing (individual and collaborative). Promotes attention on document structure rather than low-level document manipulation. Minimally constrains collaborating writers, allowing social protocols to govern the writing process.
VNS (Shipman III <i>et al.</i> , 1989)	An electronic equivalent of a scientists notebook: a data repository of notes and ideas, enhancing information sharing. Uses a single central store of notebook pages, but allows links between pages to be personalised. Pages can be simultaneously accessed. Links to notebook pages contain contextual information: thus, prior to traversing links, users can review the linked information.
KMS (Yoder <i>et al.</i> , 1989)	Interlinked screen-sized workspaces (<i>frames</i>), containing combinations of text, graphics, and images. Stresses the importance of reducing user effort to increase likelihood of collaboration: efficient browsing, easy link creation, sophisticated word searches. Notes small granularity of hypertext nodes reduces effort and frustration of "turn taking" due to concurrent access.
GROVE (Ellis <i>et al.</i> , 1991)	Real-time co-writing support. Explicitly avoids the use of locking schemes, allowing WYSIWIS (section 2.4.3) and social protocols to avoid concurrent editing.
Quilt (Leland <i>et al.</i> , 1988)	Co-writing groupware that includes many work coordination features found in messaging systems (uses email) such as automatic logging and notification of commitments. It supports automatic hypertext annotation through email. Quilt uses explicit mechanisms for supporting each writer's role within document production. These mechanisms restrict read, write, and annotation permissions for each collaborator.
Wang FreeStyle (Francik <i>et al.</i> , 1991)	Multi-media dynamic annotation of screen images using pointing, drawing, writing and, speaking. Images can be scanned into the system: for instance paper documents, or bitmaps of computer displays.

Table 2.5: A selection of collaborative writing applications

system, actually supports document production, and Quilt's (Leland *et al.*, 1988) ability to capture and notify events could classify it as an active coordination system, rather than a co-authoring one.

In this section, a new form of overlap is introduced: that of the shared workspace. Many of the systems described above afford collaborative facilities through shared workspaces. These may be accessed simultaneously and/or asynchronously, be stored locally and/or distributedly, and may enforce consensus views and/or allow personal interpretations.

Systems categorised here as "meeting environments" particularly emphasise the merging and stimulation of ideas within a shared workspace. Words commonly used in association with these systems include *brainstorming*, *participation*, *gesture*, *emphasis*, *voting*, *group cohesion*, and *involvement*.

Three classes of meeting environments are discussed in this section. The first is the oldest and most accessible form of computer conferencing, and is closely related to messaging systems. The second describes conferencing environments that have been enabled by advances in network-bandwidth capabilities: they particularly focus on support for shared drawing surfaces. The third examines computer enhanced meeting room environments. A fourth, closely related, system class will be described later (section 2.5): it focuses on integrated work environments and enhancing awareness of social presence.

As groupware systems and computer networks evolve, the somewhat artificial and technological distinctions between groupware categories presented here, and in other classifications, will become increasingly nebulous.

2.4.1 Low-technology computer conferencing systems

Low-technology computer conferencing systems have been developed since the 1970s, and many currently support large user communities: examples include HICOM *Communications and information for the HCI community* (Newman *et al.*, 1990), and a wide variety of local-site bulletin-board systems.

They typically comprise of a central database of text discussions on a set of topics, a register of users, the topics each user is interested in, and a record of the items they have seen. Users log onto the central machine and can read and donate items under particular topics. Usually, items under each topic are accessed through a linear list. A moderator

is often required to manage donations (pruning and grouping), and to keep conversations flowing.

The communication infrastructure of these systems lends itself to messaging facilities, consequently, person to person email is frequently available: the interaction structure of these systems is shown in figure 2.7, extracted from (Rodden, 1991), page-330.

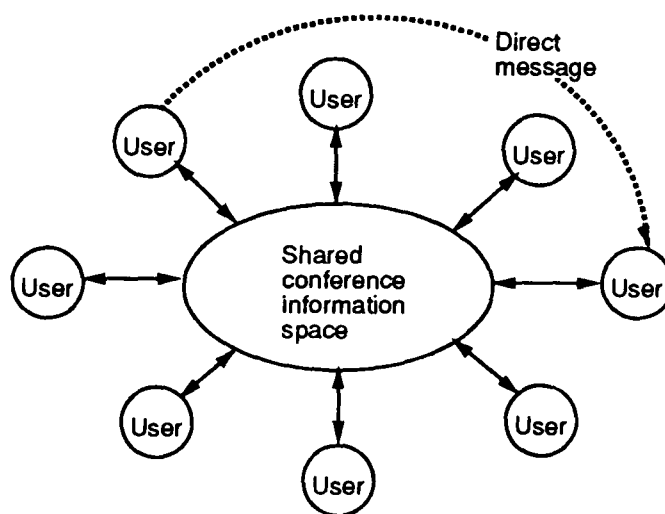


Figure 2.7: Interaction possibilities in low-technology conferencing systems.

Low-technology computer conferencing systems are primarily used as central information repositories, accessed by individuals at their leisure to catch up on interesting or related work during periods of low work pressure.¹¹ The turnaround time of conversational issues (the period between each individual's contribution) is typically days if not weeks. Although such ponderous progression has disadvantages, it allows contributions from highly remote users, and facilitates greater consideration of issues under discussion. Gordon *et al* (1985) experimented with explicitly "staged" conferences, intentionally slowing progression to ensure all participants have the opportunity to contribute.

Investigations into the use of low-technology computer conferencing systems have found that the critical factors for success are not the technology, but rather the roles played by system moderators and other "champions" and "evangelists" who promote system use, maintain focused conversations, and revive flagging discussions (Eveland & Bikson, 1988; Francik *et al.*, 1991).

Low-technology conferencing systems are discussed in detail in (Hiltz, 1984) which includes

¹¹Frequent comments are of the nature "Sorry I haven't contributed for a while, I've been too busy to log-on."

a description of the EIES system also introduced in (Hiltz & Turoff, 1985); a study of user performance with EIES is provided in (Hiltz & Turoff, 1981). Reviews of related systems are available in (Wilson, 1988).

2.4.2 High(er)-technology conferencing systems

Low-technology conferencing systems increase and equalise access to computer-mediated discussions. However, their inability to support dynamic involvement renders them inappropriate for domains such as crisis management or highly interactive collaborative “brainstorming” sessions. This limitation is largely due to the telecommunication infrastructure on which they are developed.

Advancements in network bit-rates, reductions in communication latency, and ubiquitous desk-top workstations provide an infrastructure *through* which colleagues can engage in meetings in real-time, incorporating the sophisticated display capabilities of window based workstations. The fact that people collaborate *through* the technology provides the main distinction between systems discussed in this and the following section (meeting room environments) in which computers supplement face-to-face interaction.

Groupware for real-time meetings is of two main types. The first, *collaboration-aware* (Lauwers *et al.*, 1988), specifically accounts for simultaneous use by multiple users, noting the collaborative nature of tasks supported: explicitly providing multi-user interfaces; catering for the differing roles of participants; and often employing embedded models of group tasks and processes. Systems demonstrating this approach include the co-authoring application Quilt (Leland *et al.*, 1988) (see section 2.3), LIZA (Gibbs, 1989) (below) and CoLab (Stefik *et al.*, 1988; Tatar *et al.*, 1991) (discussed in section 2.4.3). Developing collaboration-aware groupware is extremely complex, not least because of our limited understanding of *how* people actually go about group work. Collaboration-aware systems also encounter problems in encouraging users to adopt them: users will be familiar with their single-user applications, and overcoming their inertia in maintaining current working methods is a substantial barrier (these and related issues are discussed in chapter 3).

The second approach in enabling real-time groupware is to share access to unmodified single-user applications (Ahuja *et al.*, 1988; Greenberg, 1990a; Greenberg, 1990b), producing *collaboration transparent* (or *shared view*) systems (Lauwers *et al.*, 1988; Greenberg, 1990b).

These systems manipulate single-user applications, manage multiple-user access to the single input stream (through schemes such as turn-taking and floor control), and multiplex the output to the multiple-users. Some collaboration transparent systems account for personal requirements, such as specific tailored views, or particular display hardware. Figure 2.8, adapted from (Greenberg, 1990b), shows the architecture of a shared-view system that handles heterogeneous terminals, dynamic registration of meeting participants, and powerful features for meta-level dialogue around the underlying single-user application (see below). The components commonly found in view-sharing systems, shown in the figure 2.8, are as follows:

Registrar — manages several aspects of the conference, including: its initial set-up and termination; dynamic reconfiguration when people enter or leave; access to the participatory roles of individuals (some people may be restricted to view-only access, while others will be full-contributors); feedback of the current conference status.

View manager — responsible for the representation of the conference on each user's display. May account for individual view preferences, particular display hardware, private and shared views, and so on.

Chair manager — maintains some form of floor control to ensure that access to the shared application's single input stream is appropriately managed. Rather than imposing a single form of floor control it will usually be preferable to support a variety, thus allowing users' to select protocols best serving their current interaction requirements. Alternative floor control schemes include ring-passing, pre-emptive take-over, time-slices, and moderated by a human facilitator.¹²

Meta-communication manager — meta-communication enables powerful facilities for collaborators to simultaneously talk around the underlying single-user application without accessing the application itself. The focus of meta-communication research is on transferring the "gestural" information (pointing, marking, emphasising, etc) that is a vital part of group interaction (Tang, 1991; Tang & Leifer, 1988). Several systems attending to meta-communication have been developed, mainly in the form of shared drawing boards (Greenberg & Bohnet, 1991; Greenberg *et al.*, 1992; Elrod *et al.*, 1992).

¹²For further information on turn-taking and facilitation see (Greenberg, 1990b; Dubs & Hayne, 1992).

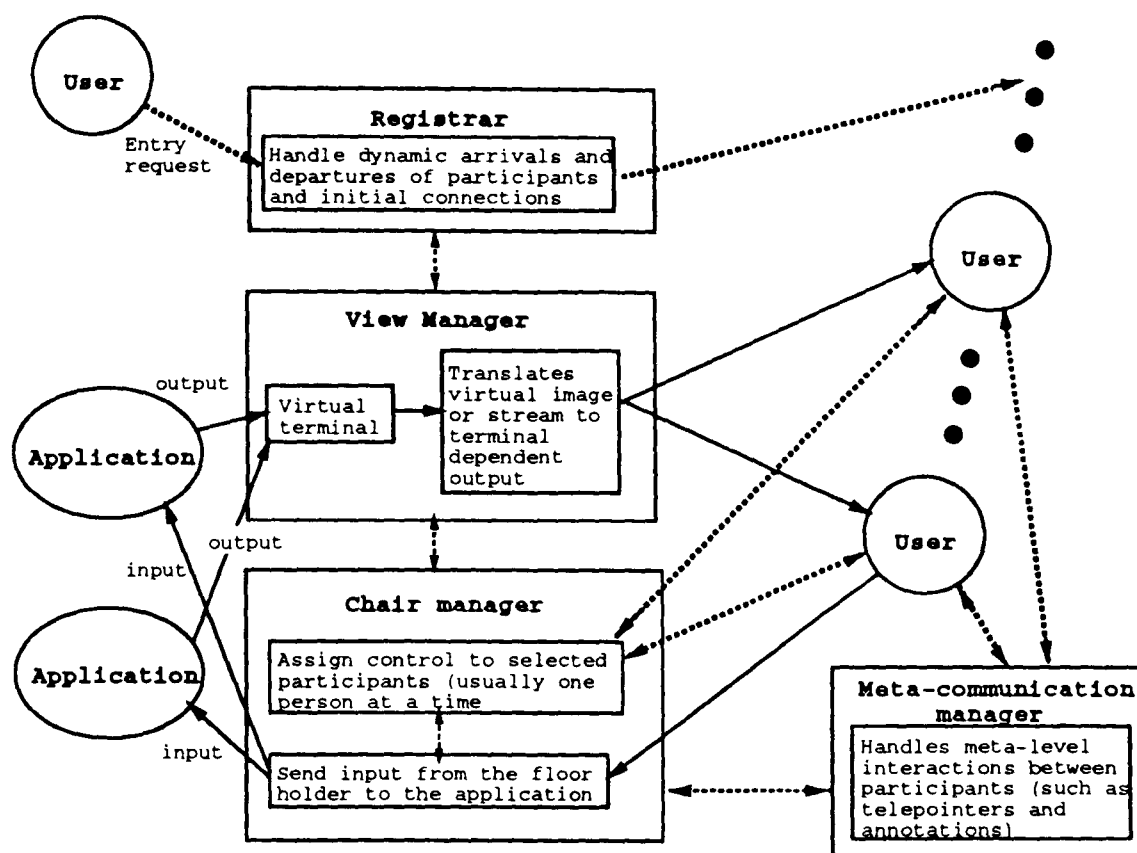


Figure 2.8: A view-sharing architecture allowing dynamic registration, heterogeneous terminals, turn-taking, and meta-communication about the task/process.

For a review of shared-view applications, and alternative ways of implementing them, see tables 2.6 and 2.6, and (Greenberg, 1990b; Ahuja *et al.*, 1988; Lauwers *et al.*, 1988). A set of closely related systems that integrate drawing surfaces with video and multi-media are reviewed in section 2.5.

System	Notes
GroupSketch (Greenberg & Bohnet, 1991)	Shared workspace for real-time meetings. Focuses on shared drawing activities (sketching, annotation, listing ideas), and on gesturing around drawn items. Ownership of pointers, pencils, and so forth, is conveyed by name-labels, and by orientation (pointing North-East, for example). System development motivated by observations of shared workspace activity (Tang & Leifer, 1988; Tang, 1991).
Commune (Minneman & Bly, 1991)	Shared workspace application for drawing, gesturing, and so on—similar to GroupSketch. Colours denote ownership of pointers, pencils, and annotations. Compared interactions between two and three party sessions, both with and without video channels. Found that although loss of video did not cause interaction difficulties, participants were less engaged in tasks executed without video.
Rapport (Ahuja <i>et al.</i> , 1988)	Shared workspace conferencing environment using a “meeting room” metaphor to provide a natural interface to conference registration procedures. Caters for a variety of display hardware. Screen sharing mechanisms allow various applications to be “brought into” meeting rooms and used as the meeting focus. Incorporates a person to person messaging facility. The paper discusses alternative architectures for screen and application sharing.
LIZA (Gibbs, 1989)	An extensible toolkit for the development of real-time groupware applications. LIZA tools are built from “active objects”. Five tools are mentioned in the paper: a room tool providing spatial representation of meeting participants; a graph tool for idea manipulation (linking, creating, deleting); a slide tool for displaying graphics; a meta-tool representing participants and tools for meeting facilitation (floor control policies, for instance); a voting tool for assistance in decisions.
Rendezvous (Hill <i>et al.</i> , 1993)	A toolkit for the construction of multi-user interfaces, particularly focusing on “conversational props”. “Props” are the things collaborators talk around, about, and through: for instance, whiteboards, models, mock-ups, and so on. Using a feature rich, constraint maintenance system designers can rapidly develop groupware for the X-Window system.

Table 2.6: Selection of real-time conferencing systems and toolkits—continued next page.

System	Notes
WeMet (Rhyne & Wolf, 1992)	A pen-based meeting support tool. Focuses on easing transitions between synchronous collaborative work, <i>and</i> asynchronous modification. Essentially, WeMet provides synchronous collaboration support, augmented with history and undo facilities.
GroupKit (Roseman & Greenberg, 1992)	A toolkit for shared workspace real-time conferences. Three main toolkit assets: an extensible object-oriented run-time architecture for managing processes and communication; transparent screen overlays enabling meta-communication around the underlying application (gesture, drawing, listing); open protocols that avoid constraints on the interaction mechanisms governing group work (for example, floor control, and conference registration). See the text.

Table 2.7: Selection of real-time conferencing systems and toolkits.

Research into shared-view systems and meta-communication screen overlays (allowing drawing, gesture, text entry, and so on) is becoming more accessible through toolkits for the development of real-time conferencing systems. Examples of such toolkits include GROUPKIT (Roseman & Greenberg, 1992), and LIZA (Gibbs, 1989). GROUPKIT *should*¹³ greatly reduce the effort required to develop conferencing systems. It provides design support for the fundamental components of collaboration-transparent (shared view) real-time conferencing systems: basic connectivity requirements; higher-level end user functions such as flexible floor-control and registration policies; and gestural and annotative facilities through screen-overlays. The practical value of GROUPKIT is augmented by the fact that it runs on any machine supporting X Windows and UNIX (through the INTERVIEWS toolkit), and by its availability through anonymous ftp.¹⁴ The use of a standard interface platform, and software availability through anonymous ftp is an impressive combination that more research groupware systems, particularly development toolkits, should aim for.

Current telecommunication capabilities make real-time conferencing systems possible (through Broadband ISDN networks), but as yet real-time desk-top conferencing is largely restricted to local sites. The systems described in this section therefore support *locally-remote* environments (for instance, a single organisation site), but these facilities are likely to become available in truly remote conferencing. They also facilitate research into the way people in-

¹³The author has no personal experience with GROUPKIT.

¹⁴From the Department of Computer Science at the University of Calgary (cpsc.ucalgary.ca).

teract through real-time computer conferencing, the work governing protocols they employ, and the tools they find useful.

2.4.3 Meeting room environments

The computer conferencing systems described above focus on supporting remote or locally-remote collaborators. They allow people to work together who, without such support, would be less able (or unable) to do so. In contrast, computerised meeting room environments, examined in this section, enhance and augment face-to-face meetings that could or would have occurred regardless of the computer's support.

To avoid alienating users who are unfamiliar with computer technology, groupware enhanced meeting rooms are usually similar to their non-computerised counterparts. Almost all have a large projected computer display visible to all meeting participants, which replaces (or supplements) the meeting room blackboard or paper flip-chart. Beyond these common factors, the diversity of meeting room groupware is pronounced: typically each participant has access to a keyboard; many systems supply every participant with a personal workstation; some restrict access to the large display to a single workstation; some use expert "facilitators" to control the software; others rely on social protocols to mediate the meeting; and so on. A summary of computerised meeting environments is provided in table 2.8, and in (Kraemer & King, 1988).

In this section, two computer supported meeting environments are examined: Colab, developed at Xerox PARC; and CMI's Capture Lab. These systems and their evaluation illustrate the subtlety of issues affecting groupware, and highlight the diversity of approaches in providing support for similar domains.

Colab

Colab (Stefik *et al.*, 1988) is designed to support face-to-face meetings involving between two and six participants. Its emphasis is on support for small design teams in which team members work in parallel. The environment consists of seven bit-mapped workstations (six mounted at the meeting table, one at a stand-up podium) connected by a local area network, and a large shared-screen display, emulating a chalkboard.

A fundamental requirement of all computer-supported meeting room environments is that

System	Notes
Colab (Stefik <i>et al.</i> , 1988; Stefik <i>et al.</i> , 1987; Tatar <i>et al.</i> , 1991)	Supports small working groups of between two and six people, each receiving a personal workstation that can display personal and shared workspaces. A large display (or electronic “chalkboard”) can be accessed by all participants simultaneously. The system maintains a relaxed implementation of WYSIWIS to allow personal workspaces (see the text for further details).
Capture Lab (Elwart-Keys <i>et al.</i> , 1990; Mantei, 1989)	Focuses on support for business people who may not be familiar with computers, and consequently requires a simpler interface than technically oriented systems such as Colab. Up to nine participants, each receiving a Macintosh computer that can support private and shared workspaces. The large central display, which can only be accessed by one user at a time, is accessed through a pre-emptive interrupt protocol. See the text for further details.
Project Nick (Cook <i>et al.</i> , 1987)	Support for highly structured meetings to establish the requirements of large scale distributed systems. System composed of a large shared display, networked computers (one for each participant), and software tools to “apply automated facilities to the process, conduct, and semantic capture of design meetings.”, page-132.
Amsterdam Conversation Environment (Dykstra & Carasik, 1991)	Support for group interaction in face-to-face meetings. Rather than supporting decision making or meeting processes, ACE is designed to support conversation and to promote and stimulate interaction among participants. “... we started to question what it was that we really wanted to support: processes or people?”, page-420. The paper concentrates on design concepts. The system itself is described as a series of prototyping exercises with the final version being a network of 21 Macs supporting private and shared windows, and Cut-Copy-Paste operations between them.
Group Decision Support Systems (GDSS) (Kraemer & King, 1988)	A wide variety of GDSS are reviewed. Noting alternative support schemes, and the use of human “facilitators”. Facilitators roles are examined, including: operate the technology, instruct participants on use of the decision model, coordinate the activity, and document the group’s work.

Table 2.8: Selection of computer-supported meeting room environments, and a review of Group Decision Support Systems.

participants receive the shared context available in normal meetings. To some extent, this is achieved by the shared electronic “chalkboards”. However, the decision in Colab (and many other systems) to provide all participants with individual workstations potentially detracts from the shared workspace. The provision of personal workspaces can isolate individuals, removing the shared context by requiring too much concentration to engage in the meeting *while* operating the computer (Mantei, 1988).

The Colab developers initially attempted to bypass this problem by enforcing the shared workspace within each personal workstation display. To do so, a variant of WYSIWYG (what you see is what you get) (Thimbleby, 1990a) was used. WYSIWIS (Stefik *et al.*, 1987), *what you see is what I see* (pronounced whizzy-whiz), ensures that all meeting participants share the same view, whether their attention is directed at personal workstation screens or the electronic chalkboard.

Rigidly implementing WYSIWIS in this manner requires all displays to be identical. This imposes limitations and usage difficulties, and necessitates heavy network traffic (see below). Strict WYSIWIS also disallows many of the benefits that motivate computerised meeting environments, such as the ability to separate personal and group displays: for example, displaying pre-prepared personal work, or working independently or in sub-groups during meetings. Furthermore, strict WYSIWIS requires all participant’s cursors to be constantly visible, which is likely to cause confusion and screen clutter. After initial experimentation, the Colab team implemented a version of WYSIWIS, relaxed across four dimensions (Stefik *et al.*, 1987):

1. Display space—rather than requiring entire screen displays to be shared, objects on the screen (such as specific windows and pointers) can be selected for sharing.
2. Time of display—strict WYSIWIS requires all user actions to be dynamically apparent to all users in real-time. Relaxing this constraint reduces excessive network transmission costs. Colab’s communication grain size is relaxed so that drawing events (for example) are transmitted only when completed.
3. Subgroup population—rather than requiring all activities to be shared by all participants, Colab allows sub-groups to manage separate workspaces.
4. Congruence of view—strict WYSIWIS requires all display views to be identical. However, users are likely to have personal (and sub-group) view preferences such as window

position, window layering, and window presence or absence.

Colab supports three collaboration-aware software tools for separate meeting processes: *Boardnoter*, *Cognoter*, and *Argnoter*. *Boardnoter* provides an electronic chalkboard using metaphors for chalk, an eraser, and (strangely for a chalkboard!) a typewriter: several systems providing similar support, such as GroupSketch, were discussed in section 2.4.2.

Cognoter, the most sophisticated Colab tool, employs three explicitly distinct stages (brainstorming, idea organisation, and idea evaluation) to support collaborative preparation of presentations. The brainstorming stage promotes synergy in idea generation, encouraging users to donate anonymous ideas, in the form of short phrases, to the shared screen. Ideas can be moved or edited, but not deleted. In the organisation phase the group attempts to establish an ordering between the ideas generated during brainstorming. The two basic operations used to assist idea organisation are linking ideas into presentation order, and grouping related ideas. The evaluation stage determines the final form of the presentation. Only at this, final, stage can ideas from the initial brainstorming session be deleted.

The third Colab tool, *Argnoter*, supports general argumentation around proposals, and uses three stages comparable to *Cognoter*.

The initial papers describing Colab (Stefik *et al.*, 1987; Stefik *et al.*, 1988) made limited observations of system use. Problems with *Cognoter* included its separation of task-phases which resulted in users adopting work-around strategies to avoid explicitly changing tools. Other problems were derived from the relaxations of WYSIWIS: for example, “window wars”, in which users would competitively scroll windows (one scrolling up while another scrolls down) resulted from the lack of congruence between window representations—comments such as “the top-left window” are redundant in non-congruent displays.

Four years after their initial descriptions of Colab, the team at Xerox PARC published the results of extensive observation of the lab’s use, and in particular, the use of *Cognoter* (Tatar *et al.*, 1991). They identify a major and fundamental flaw in the tool’s design, which they describe as a failure to account for the differences between human and computer-supported synchronous communication:

“...many of the serious problems in *Cognoter* stem from a culturally prevalent, easy-to-make assumption that communication consists of bits of verbal or textual material passed whole from person to person” (Tatar *et al.*, 1991), page-206.

Other problems observed by the Colab team included the lack of synchronisation between speech and action: verbal descriptions of actions were out of context to the majority of the group because they would not become visible until the action was completed—a result of the *time of display* relaxation of WYSIWIS.

The extreme level of discontent with the system is clearly stated,

“...they found it so frustrating that they put their heads in their hands, raised their voices, and ultimately threatened to walk out. They expressed astonishment that anyone would build such a tool” (Tatar *et al.*, 1991), page-190.

These observations are being used to develop a second generation *Cognoter*, called *Cnoter*, which aims to retain *Cognoter*'s positive features, and eliminate its flaws (Tatar *et al.*, 1991).

The Capture Lab

The Capture Lab meeting room developed at CMI (Elwart-Keys *et al.*, 1990) provides an interesting contrast to Colab. Colab focuses on meeting processes for computer-familiar teams using task specific and collaboration-aware software. The Capture Lab, in contrast, furnishes a meeting environment for non computer-familiar businessmen (without the use of a facilitator) through unmodified single-user applications in a collaboration-transparent manner. Much of the research on the Capture Lab is observation-based, establishing how computer supported meeting environments are used, and the facilities that are beneficial (Mantei, 1989).

The Capture Lab environment (shown in figure 2.9) consists of eight Macintosh computers with monitors recessed into the meeting table. A ninth Macintosh serves as the “public” computer, and its display is shown on a large electronic “blackboard”. Only one participant can control the public computer at a time, and access to it is mediated by a pre-emptive protocol: pressing the “ON” key of any keyboard establishes control, and disconnects the current holder. Although this pre-emptive protocol may appear to invite conflict, the design motivation was to allow social protocols to prevail, and to promote verbal interaction. The designers also note that current understanding of meeting processes is insufficient for work governing protocols to be explicitly encoded in systems.

The user that controls the public computer can transfer information bi-directionally between their private computer and the public one through cut, copy, and paste operations.

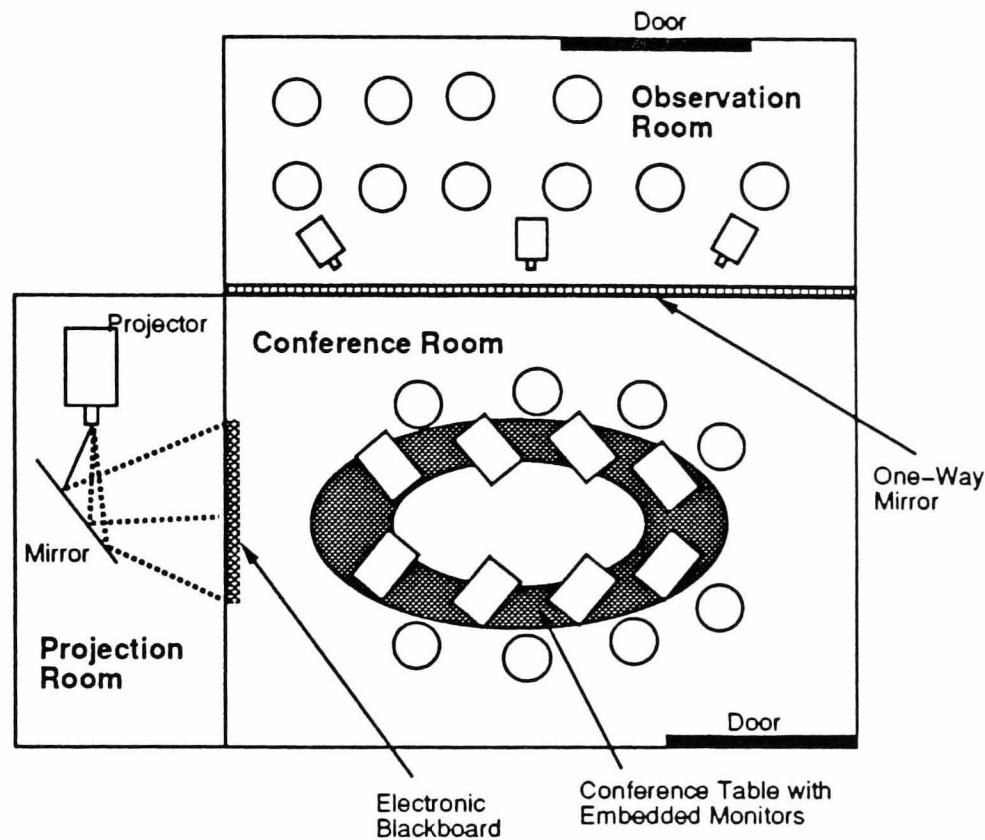


Figure 2.9: Layout of the Capture Lab meeting, observation, and projection rooms.

Six interface design requirements arose from their intention to support non computer-familiar businessmen:

1. Easy transition between personal and group work.
2. Privacy of personal workspaces—allowing users to work in parallel. Private workspaces also allow participants to prepare their contributions before displaying them on the shared screen. The design aim is to increase participation from those who might feel intimidated by the prospect of a public display of poor typing skills or lack of interface familiarity.
3. WYSIWIS—the Capture Lab achieves strict WYSIWIS (of a form) through the shared context of the single public display. By avoiding the maintenance of WYSIWIS across private workstations it overcomes many of Colab's problems in distributing and relaxing WYSIWIS.
4. Keeping pace with the conversation—Capture Lab's single shared display eases satisfaction of this requirement: actions on the shared screen can be accompanied by verbal

comments.

5. Support for protocols and social structures—allowing the group to select their working processes rather than having the system impose them.
6. Minimal training.

Mantei's (1989) observations of executives using the Capture Lab are fascinating. Rather than focusing on the technology, her observations concentrate on the physical layout of the meeting room. The design issues discussed include seating arrangements, and the field of view between participants, for example:

“To make the visual distance look less, the conference table was built with a light colored formica oval covering in its center and a dark circle of 1.5 feet around the edge”, page-159.

Her observations emphasise the subtlety of groupware design, and are echoed in the evaluation of *Cognoter*:

“[*Cognoter*] encountered serious difficulty because it did not recognize that it had entered a new arena. It slipped up on implicit aspects of the system, places where the designers didn't realize they were making choices.”, (Tatar *et al.*, 1991), page-207.

2.5 Seamless interaction and social presence

The systems examined in this chapter so far have been categorised according to the type of support they provide. This would appear to contradict section 2.1.1 which raised concerns on the affects brought about by classifying groupware. In this section, groupware beyond the standard (limiting) classification schemes (figure 2.1) is reviewed. These systems address some of the boundaries and dependencies imposed by groupware, and attend to general human needs in collaboration.

Although demonstrating improvements in support for collaboration, these systems (like those described above) impose specific dependencies, and fail to fully capitalise on their potential. The limitations of these (and other) systems are examined in detail in chapter 3.

These systems are revisited in 6, in which a groupware system, TELEFREEK, demonstrates an alternative approach for attaining integrated environments and social presence facilities.

2.5.1 Background

The task-specific, and channel-dependent, work environments supported by many groupware applications is artificial. It fails to account for the way people actually work and neglects their real communication needs (Bly *et al.*, 1993). It is unrealistic to expect that any single tool can be adapted to all tasks, and similarly, no single communication mechanism can best serve all communication needs. Yet such expectations and requirements are displayed by some applications (such as *The Coordinator*), which attempt to capture all communications (even those originating outside organisational boundaries). Real work environments draw on many different tools and a variety of media according to communication needs.

In supporting individuals, personal computers have reduced the user-effort derived from the use of multiple task-specific systems by promoting consistent “look and feel” across applications: for example, see the Apple Macintosh “User Interface Guidelines” (Apple Computer, 1988). The problem is more complex for groupware: not only must applications present *every* user with their preferred (and consistent) interface, but they must do so across several hardware platforms.¹⁵

Rather than concentrating on specific collaborative tasks, groupware in this section examines the (co-)work environment, and applies modern technology in its augmentation. The issues include:

1. the merging of multiple communication mechanisms and computing facilities;
2. social factors in collaborative work, and how computers can help satisfy them.

Extensions to this work, discussed in chapter 6 include:

3. the human factors that affect how people make contact and establish communications;
4. the merging of support for human needs in collaboration with access to communication channels, groupware tools, and related information sources.

¹⁵New applications that assist designers in overcoming such problems are discussed in chapter 4.

The following two sections discuss, firstly, groupware focusing on “seamless interaction” in which communication facilities are intermixed with computing tools, and secondly, systems attending to “social presence” in collaborative environments.

2.5.2 Seamless interaction environments

The importance of gesture and other subtle forms of communication were discussed in section 2.4.2. Shared-drawing applications (also discussed in section 2.4.2) demonstrate limited gestural capabilities: pointing, emphasising, out-lining. Kraut *et al* (1988b; 1988a; 1990) examine a broader range of issues affecting communication, extending the time-frame beyond individual collaboration sessions. They particularly study the formation and maintenance of collaborative relationships, noting three factors that critically influence the success of partnerships: the ability to engage in interactions that are **frequent, high-quality** (stimulating more than one sensory channel), and **low cost**. These three properties are most readily afforded by close physical proximity, and Kraut *et al* provide design recommendations for groupware’s replication of physical proximity.

Founded on these and similar observations, several groupware systems combine high-quality communication channels (typically video and multi-media) with computer technology. These facilities enable natural human communicative cues, such as facial expression, to augment computing tools. They create *seamless interaction environments* in which boundaries between tools and facilities for mediating (collaborative-)work are reduced.

As yet, much of the research in this area focuses on maximally exploiting the limited telecommunication bandwidth, for instance examining how video is best integrated with computing tools.

Several of these systems are summarised in table 2.9.

Video technology and groupware

Several research projects have examined the comparative effectiveness of assorted combinations of communication channels in supporting group work (Smith *et al.*, 1991; Scrivener *et al.*, 1992; Fish *et al.*, 1993; Egado, 1988; Gale, 1991).¹⁶ Generally, these studies conclude that video adds little to the end products of collaboration, but that users are more engaged in their

¹⁶For the original work of this nature see (Chapanis, 1975) referenced in (Greif, 1988a).

System	Notes
TeamWorkStation (Ishii & Miyake, 1991; Ohukubu & Ishii, 1990)	Video fusion of workstation images, physical desk-tops, and collaborators faces or hands to provide “seamless interaction between individual and groupwork.” Merges video signals from a variety of sources to integrate work environments (see the text).
ROCOCO Sketchpad (Scrivener <i>et al.</i> , 1992)	Shared drawing surface. Investigates video channels as supplements to audio communication in support of shared drawing activities. Describes an experiment involving real-time collaborative work between Australia and the UK. Users gauged the support more worthwhile in widely dispersed collaborations (presumably because it would otherwise be impossible).
SharedARK (Smith <i>et al.</i> , 1991)	A real-time, multi-user, Alternative Reality Kit. Experimental system for investigating competing combinations of communication media. Video communication is enhanced with eye-to-eye contact through a “video tunnel” (see figure 2.10).
ClearBoard (Ishii <i>et al.</i> , 1992; Ishii & Kobayashi, 1992)	Shared drawing application giving the visual effect of “looking through and drawing on a big glass board.” Extends the communicative value of eye-to-eye contact by providing “gaze-awareness”, in which the direction of eye-sight yields contextual information of the objects and items under discussion. Uses video fusion techniques developed from design experiences with TeamWorkStation. See the text.
Marcel (Newman & Wellner, 1992)	A paradigm shift for the desk-top metaphor: real-world desk-top objects are augmented rather than simulated. The ultimate goal of the research is a reduction in incompatibilities between paper and electronic domains. The real-world desk is scanned by computer image processors, and additional information from the computer can be projected onto the desk. Two sample applications are described: a desk-top (projected) calculator, and a French-English translation assistant. Interesting and visionary research.

Table 2.9: Summary of “seamless interaction” groupware that use video technology.

work, and more inclined to future interaction when video is available: video improves social aspects of collaborative work. The areas in which video is most beneficial are summarised in (Fish *et al.*, 1993):

1. increasing the spontaneity and frequency of communication;
2. supporting social relationships;
3. coping with complex problems;
4. support for research and development.

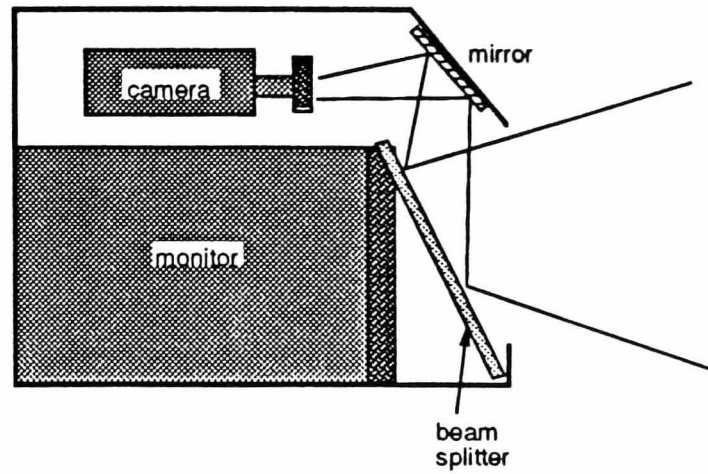
Motivated by these observations, video and multi-media research strives for optimal use of available bandwidth. Two schemes supported by groupware which improve interaction through video are eye-to-eye contact (and *gaze awareness*), and the fusion of real-world and computer workspaces. The system best exemplifying *both* these approaches is *ClearBoard* (Ishii & Kobayashi, 1992; Ishii *et al.*, 1992) which is described below.

Eye-to-eye contact enhances the sense of engagement provided by video communication. During development of *ClearBoard*, Ishii and Kobayashi (1992) observed:

“Lack of *eye contact* has been another problem of existing desk-top video conferencing systems. People feel it difficult to communicate when they cannot tell if the partner is looking at him or her... ‘eyes are as eloquent as the tongue.’”, (Ishii & Kobayashi, 1992), page-526.

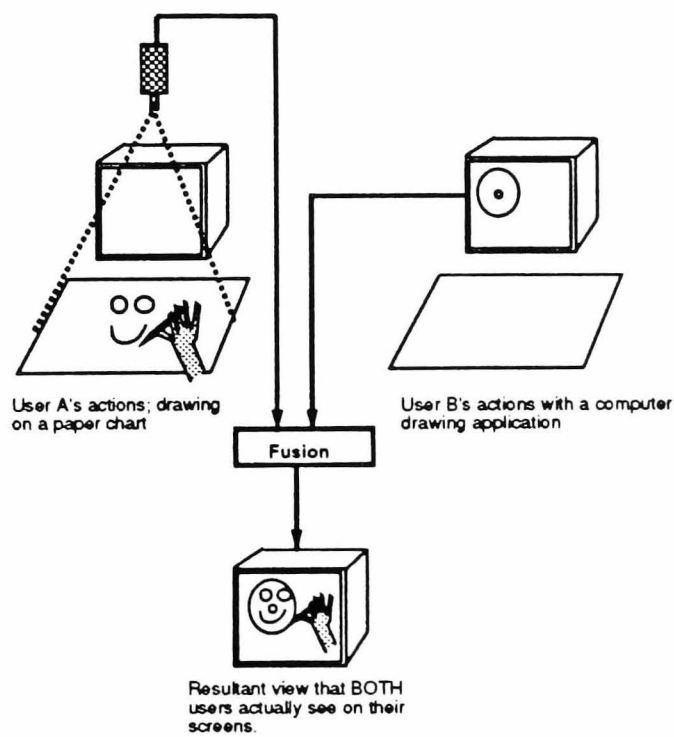
The “video tunnel” of SharedARK (Smith *et al.*, 1991) demonstrates a simple mechanism for achieving eye-to-eye contact through video: see figure 2.10, adapted from page-35 of (Smith *et al.*, 1991).

Video fusion extends the transparent overlay metaphor (used by systems such as *Commune*, *GroupSketch*, and *GroupKit*, see tables 2.6 and 2.7) that allows users to gesture, annotate, and scribble on transparent windows *placed over* shared applications. Video fusion allows transparent overlays to convey full-motion video of anything users place before the live cameras, for example: shared computer applications, real-world desk-tops and notepads, real-time facial expressions, hand gestures, and so on. Video fusion techniques are best exemplified by the *TeamWorkStation* (Ishii & Miyake, 1991), see figure 2.11.



The beam-splitter and mirror provide a screen-centre perspective for the camera. Consequently, when both users look at the screen they have eye-to-eye contact.

Figure 2.10: SharedARK "Video Tunnel".



User A's actions (on a drawing surface), and user B's actions (with a computer drawing application) are merged. The resultant display shown on both user's monitors is shown at the bottom.

Figure 2.11: Video fusion.

ClearBoard

Evaluating the *TeamWorkStation*, Ishii and Kobayashi (1992) recognised the importance of smooth transition between face-to-face communication (through video) and work activities. To ease this transition, in ClearBoard, they pursue *gaze awareness*: an extension to eye-to-eye contact. Through gaze awareness the engagement of eye-to-eye contact is maintained, and in addition collaborators share a common visual frame of reference, allowing each user to see where their co-worker's attention lies: comments such as "this one" are placed in context by the direction of sight.

ClearBoard's user support is a metaphor of a stand-up glass sheet. Users work on opposite sides of the sheet: drawing on the glass, maintaining eye-to-eye contact through the pane, and keeping a common frame of reference on the glass. A prototype, ClearBoard-1, did not incorporate computer applications for user support. ClearBoard-2, however, supports a shared drawing application "TeamPaint". The architecture of ClearBoard-2 is shown in figure 2.12, adapted from (Ishii *et al.*, 1992), page-37.

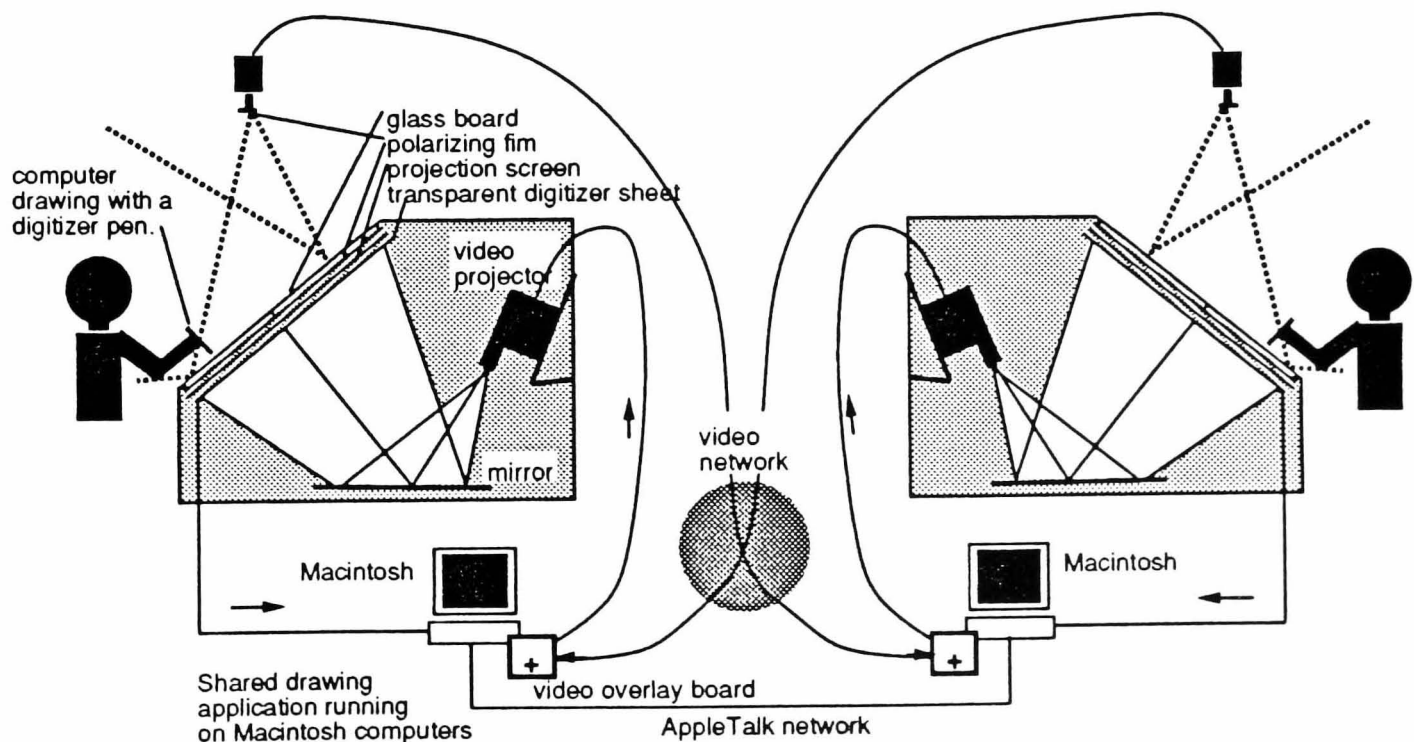


Figure 2.12: The architecture of ClearBoard-2.

2.5.3 Social presence systems

Social presence groupware represents a change in emphasis from support for *tasks* to support for general and social *human needs* in collaboration. Seamless interaction groupware used observations on collaborative partnerships (Kraut *et al.*, 1988a; Kraut *et al.*, 1988b; Kraut & Dumais, 1990) to motivate improvements in communication richness. In contrast, social presence systems, use the overall observation of Kraut *et al* that physical proximity best serves communication and collaboration requirements. These systems encourage, ease, and assist access to frequent, high-quality, low-cost interactions, and many adopt metaphors that simulate close physical proximity.

A summary of three types of social presence systems is given in the following sections, these (overlapping) types are: virtual hallways, social browsing systems for casual interaction, and directed encounter systems for specific needs in communication.

Virtual hallways and the Xerox “Video Wall”

Between 1984 and 1988 a constant video connection was maintained between Xerox PARC in Palo Alto, California and an associated research lab eight hundred miles away in Portland, Oregon. At each end of the connection, large-format video screens and cameras provided continuously present “virtual hallways”, called *Video Walls* (Goodman & Abel, 1987). Over four years the connections were extended to include limited point-to-point connectivity. In its final state, the facilities created a *Media Space* (Bly *et al.*, 1993) that consisted of several “virtual common rooms”, eight personal video stations in offices at Palo Alto, and six in Portland.

The *Media Space*’s ability to mediate specific collaborative tasks was limited by network capabilities. However, rather than supporting specific tasks, its aim was “to recreate in a working group separated geographically the sense of embeddedness that we found in working together locally” (Bly *et al.*, 1993), page–33. The system was used extensively for casual interaction, social browsing, and social awareness functions. As such it was a success, and although terminated due to lab closure, it stimulated much further research within Xerox sites and elsewhere.

Social browsing and the CRUISER

Several systems and facilities have been developed to promote and support social interaction over a network. The most rudimentary of such facilities are network browsing commands (such as *who*, and its variants, on Unix) that provide a list of users on some (or all) of the local network machines.

Groupware for social browsing almost without exception achieves its support through video channels. Many experiment with proximity metaphors (virtual hallways, corridors, and offices), and a variety of prompts are used to stimulate casual interaction. CRUISER (Root, 1988; Fish *et al.*, 1992; Fish *et al.*, 1993) provides “virtual proximity” through audio and video metaphors that simulate close physical proximity. CRUISER’s support for social awareness goes beyond that supplied by the shared common rooms of the Xerox *Media Space*; it actively searches for and initiates social interaction.

CRUISER users can “drop into” the offices of others, initiating audio and video conferences between sites that could (in theory) be widely distributed.¹⁷ People who do not want interruptions can apply various levels of prohibition on the ability of others to establish contact, for instance, users can draw “blinds” across their cameras, and can switch off audio connections. Three methods for encouraging spontaneous interaction are supported:

Cruises — in which the user initiates a series of one or more calls (audio/visual links) to either a list of names, or randomly selected individuals.

Autocruises — analogous to bumping into someone in the corridor. CRUISER can initiate calls at random intervals.

Glances — allowing users to peek into the offices of others.

CRUISER’s evaluation (Fish *et al.*, 1992) yields important insights into user requirements in communication. A primary, and unexpected, use of video channels was to “ambush” colleagues as they entered their offices. A continuously open video channel to a colleague’s empty office would be maintained, allowing the “waiter” to continue working while peripheral visual awareness of the video link notifies the colleague’s arrival. Typically, once aware that the colleague had arrived, the waiter would terminate the video connection and walk to the

¹⁷Although networks are capable of achieving the necessary transmission rates, they are typically restricted to local sites or tightly connected distant sites.

colleague's office. Pragmatically, this use of high-bandwidth channels to transmit redundant information is both wasteful and expensive. Given that CRUISER included a "directory listing the availability status of all other users" (Fish *et al.*, 1992), page-39, it would be surprising if low-transmission-cost "ambush" facilities are not added to the system to avoid the wasted use of video channels. TELEFREEK, described in chapter 6, allows equivalent functionality through base-technology facilities.

Directed whereabouts systems and Xerox EuroPARC

A natural compliment to social browsing facilities are systems that enhance awareness of who is about, what they are doing, and whether they are available for conversation. Although directed-encounter facilities overlap with those of social browsing systems, the primary intent is to support and assist users who have a particular need for interaction with a specific individual or group.

Simple facilities for directed encounters include Unix utilities such as `finger`. When invoked with a user name (login-name, or part of the real name) `finger` returns information about the user: the login and real names; home directory; idle time; the time mail was last read (disabled at many sites); and the contents of the user's `.plan` file (typically, a "funny" personal description). Unix utilities also allow people to look up addresses beyond the local network: for instance, `hosts` assists internet addressing by providing access to i-net address codes through aliases and symbolic names. The X.500 protocol supports a worldwide directory service to places, organisation, and individuals. The relevance of X.500 to groupware is examined in (Prinz & Pennelli, 1991), and a review of interfaces to X.500 is given in (Iannella, 1992).

Whereabouts and social awareness has been a focus of research at Xerox EuroPARC, Cambridge. At EuroPARC, several systems draw on the information provided by the site's active badge technology (Harper *et al.*, 1992). "Active badges", worn on clothing, are location sensing devices that transmit an infra-red identification code every 15 seconds. Detectors are located in rooms, hallways, and stairwells. A central database collects identity codes, location, and time information, which can be used for several purposes:

- Establishing the current location of any individual, and enabling point to point video or audio links to be initiated by name rather than address.
- Any badge-wearers' location can be displayed on a computer displayed floor plan.

- A personal diary of activities can be automatically generated, noting events such as meetings, their location, and time. For example, PEPYS (Newman *et al.*, 1991) generates diaries that aim to aid human memory through “activity-based information retrieval” (Lamming & Newman, 1991).
- Events (past, present, and future) can be browsed, and “event daemons” can provide automatic notification of events within the database, such as meeting arrangements and other appointments: implemented in the *Khronika* system (Lovstrand, 1991)).

Not all social presence systems at EuroPARC use the active badge technology. *Portholes* (Dourish & Bly, 1992) uses low-bandwidth video technology to maintain awareness, or a “sense of community”, between distributed sites (EuroPARC and PARC, California). *Portholes* goes some way towards integrating access to information about colleagues with access to various communication channels (email and, in some cases, a “listen” device), but the designers note that its major use is in locating colleagues.

Finally *RAVE* (Gaver *et al.*, 1992) integrates many of the EuroPARC systems to “Realize A Video Environment”, allowing:

“seamless transition between support for synchronous collaboration and systems which support semi-synchronous awareness over long distances and of planned and electronic events”, page-27.

2.6 Summary of the overview

Four categories of groupware have been presented: messaging systems, co-authoring tools, meeting environments, and the recently emerging category, seamless interaction and social presence systems. The perspective applied in this categorisation is, wherever possible, that of the services provided to users.

Whatever the perspective, classifying groupware is a risky business: showing *the way things are* does not necessarily reflect *the way things should be*. Recently, several projects have noted groupware’s tendency to forward and enforce work environments within falsely isolated applications or work-domains. This tendency (which may be partially attributed to technology-oriented classifications) is addressed by the fourth, and most recent, groupware application domain, *seamless interaction and social presence systems*.

Many areas of CSCW research have been intentionally omitted in this overview. The contributions of social scientists, ethnographers, psychologists, and others have been cited only when *directly* impacting on the focus of this thesis: the design and implementation of groupware.

In the following chapter some of the systems described above are revisited. Having established a foundation for understanding groupware, its ambitions, and techniques, groupware's success and the problems it encounters are investigated, with specific examples, in chapter 3.

Chapter 4 describes four design principles for improving groupware's success. As these principles are intended for application in *all* groupware domains it was essential that *all* domains, and their variants, be described in this overview.

Chapter 3

Problems in user-acceptance of groupware

3.1 Introduction

The number, diversity, and dates of the systems detailed in the preceding chapter demonstrate that interest in CSCW and groupware has increased dramatically in the last decade. This increase is motivated partly by a recognition that computers could better support the inherently collaborative nature of work, and partly by new technologies and network capabilities. Yet, despite substantial investment of time, money, and research effort groupware successes have been small-scale and restricted to local sites or tightly-connected distant sites.

This chapter investigates the problems that cause the failure of groupware and examines the relationship between issues affecting groupware's acceptance: the costs incurred in its use; the benefits it offers; the personal satisfaction it imparts; and its establishment of a "critical mass" of users. The observations made are used to develop groupware design principles in chapter 4.

3.1.1 Chapter overview

The problems encountered by users as a direct result of groupware is the primary focus of this chapter. In section 3.2 the literature on groupware failure is examined, focusing on Grudin's (1988) identification of three particular causes. The cost/benefit disparity observed by Grudin is examined and extended, and the costs (or undesired aspects) of groupware are viewed as the overheads of *user-effort* beyond that required to execute personal work tasks.

Having detailed and generalised Grudin's observations, section 3.3 examines the relationship between factors affecting the success of groupware. Particularly focusing on factors prevalent during the initial stages of system use, it is shown that groupware is susceptible to a vicious circle that confounds its prospects for success. Although the additional effort required by groupware may, to some extent, be offset by the benefit derived from it (Goodman & Abel, 1987), there is typically a cost/benefit disparity *across* users (Markus & Connolly, 1990): between those carrying out additional work and those who gain the benefit. Describing the vicious circle clarifies the relationship between the sources of user-effort, the benefits yielded by systems, the problems of establishing a "critical-mass" of users, and the encouragement required to entice individuals to adopt groupware. It is shown that the factor dominating this vicious circle is user-effort. Consequently, user-effort becomes the primary focus in the remainder of the chapter.

The effort inherent in collaborative work, and the additional effort required when telecommunication channels mediate interaction are described in sections 3.4 and 3.5. Following these brief discussions, sections 3.6 to 3.9 examine the user-effort explicitly derived from computer support for collaborative work.

3.2 Grudin's observations on groupware failure

Many papers have cited groupware's lack of popularity, but few have examined the causes. Of those that have, most have made specific investigations into the limitations of particular types of collaborative support, examples include: a critical evaluation of *Colab's* meeting-room environment (Tatar *et al.*, 1991); a review of video-conferencing as a technology to support groupwork (Egido, 1988); and a criticism of designers' tunnel-vision caused by technology-driven research into computer simulated physical proximity (Hollan & Stornetta, 1992). The

latter two papers concur that using technology to replicate human communication is inappropriate because it fails to exploit the computer's ability to contribute to collaboration; they argue that groupware must answer the question "what does it do that we couldn't do otherwise?" (Egido, 1988).

The most widely cited paper specifically on groupware's failure is Grudin's (1988; 1989), in which he identifies three major issues.

1. **The disparity between who does the work and who gets the benefit.** Users who execute additional work to support the system's operation often receive no direct benefit for doing so: there is a *cost/benefit* disparity. Their motivation to continue providing additional work is therefore low.
2. **The breakdown of intuitive decision-making.** There is a paucity of experience in designing collaborative interfaces. Designers' intuition, which may be adequate for single-user applications, is less reliable for the dynamic, complex, and subtle environment supported by multi-user systems. Managerial intuition can also mislead the design of groupware—managers, attracted to facilities that provide personal benefit, can overlook the implications for other system users.
3. **The underestimated difficulty of evaluating CSCW applications.** Evaluating groupware is necessarily complex, involving long trial periods and many participants. The inadequacies of laboratory testing are noted by Borenstein and Tyberg (1991):

"...CSCW cannot be judged in the absence of a real user community. Any system, therefore, that claims to make a contribution to CSCW, but has no significant base of regular users, is making an empty or unverifiable claim", page-230.

Although this statement over-emphasises its argument (substantial contributions have been made by demonstrational systems), prototyping groupware is complex, and improvement through iterative design (Gould & Lewis, 1985) is hindered by the discouragement resulting from frequent system changes (Cool *et al.*, 1992). Therefore, to effectively evaluate groupware, the application must be complete, and must provide a polished interface. Without these properties, isolating the causes of system failure (or

success) will be complex: should failure be attributed to a poor interface, or to some other issue?

The first issue highlights problems arising during system use, and forwards a reason why users are unwilling to adopt and maintain groupware. The second two issues concern the design and re-design of applications. Categorising Grudin's causes of failure into three levels (system use, design, and evaluation) provides a basis for further investigation into the problems encountered by groupware, and the ways to improve it, as summarised below:

System use — further examination of the causes of failure in system-use is required. Using Grudin's cost/benefit disparity as a platform for investigation, this chapter addresses the following questions:

- What are the components of cost and benefit as perceived by the user?
- What are they derived from?
- How are they related?
- What other factors adversely affect user-acceptance?

System design — the observation that designers' intuitions can be unreliable (Smith *et al.*, 1982; Norman, 1983) and mis-guided (Grudin, 1988) is correct, but knowing that intuitions are fallible is only the first step towards removing their weakness.

What assistance can be offered to guide designers' intuitions and to facilitate the avoidance of system rejection? This question is addressed by the groupware design principles detailed in chapter 4. The principles are founded on the observations of groupware's usage problems made in this chapter.

Evaluation — improvements in methodologies for groupware evaluation are required. Here too design principles can be of use; they provide a point of reference for identifying the positive and negative attributes of system behaviour.

3.3 The vicious circle confounding groupware's success

In this section, the factors encouraging groupware's use are related to those opposing it. It is shown that groupware is susceptible to a vicious circle that impedes the attainment of its intended benefits.

The additional effort required by a system will discourage people from using it. Goodman and Abel (1987) state that “People will use new communication system to the extent that they require no more effort than existing ones”, page-144. While Goodman and Abel acknowledge that work enhancements (benefits) resulting from system use may offset this discouragement, groupware tools are prone to a vicious circle that restricts the initial realisation of system borne benefits: see figure 3.1, extracted from (Cockburn & Thimbleby, 1992a), page-144. The vicious circle also inhibits the formation of a “critical mass” of users (Greif, 1988b; Ehrlich, 1987; Cool *et al.*, 1992) upon which all collaboration support systems are dependent.

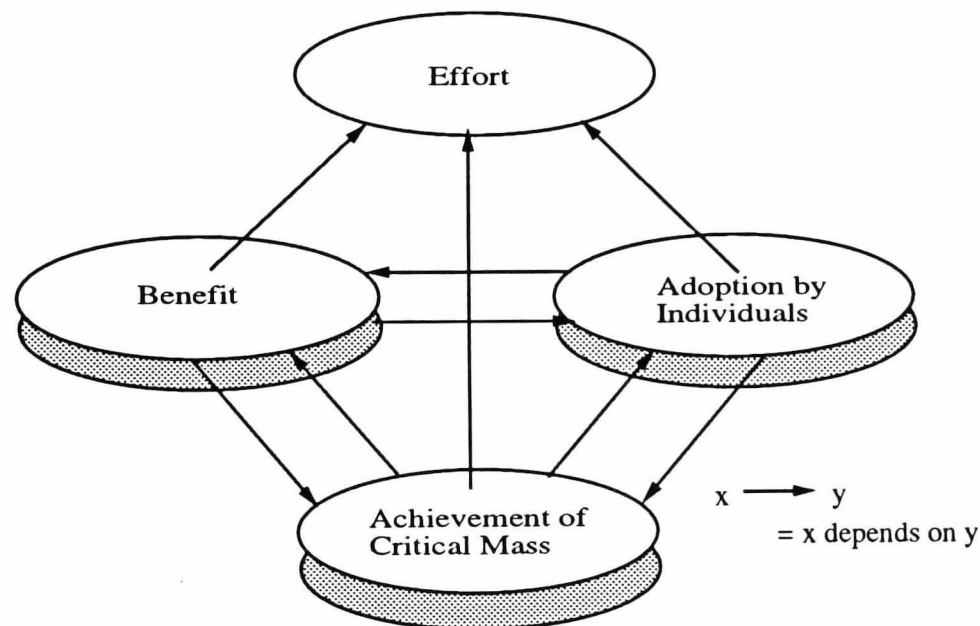


Figure 3.1: The “vicious circle” of dependencies in groupware adoption.

The system factors in the vicious circle’s chain of dependencies (described below) are its *benefits* (as perceived by users), its *achievement of critical mass*, and its *adoption by individuals*.¹ The key determinant in the realisation of each of these components is the level of *effort* involved in system use.

Benefit and benefit-lag. Each person’s willingness to use a new system is dependent on the benefits they perceive will be derived from it. During the initial stages of system use (*adoption*) this assessment is primarily determined by the *immediate* benefits offered: new users have no other experience on which to gauge the system’s value. Computer systems, however, suffer from a “benefit-lag” during which the effort put into mastering the interface

¹The word “adoption” will be used to describe people incorporating new systems into their working methods.

out-weighs the benefit received. Benefit-lag can lead to users adopting “satisficing strategies” (Thimbleby, 1990a): rather than devoting time and effort to learning new and (probably) more efficient ways of doing things, people continue to use methods that get the current task done *now*.² Observing the problems caused by benefit-lag, Mantei (1989) notes that “... a high learning threshold would cause meeting participants to reject the technology”, page-154.

Problems of benefit-lag apply to all computing systems, whether single or multi-user. The group nature of groupware’s benefits, however, make benefit-lag a particularly complex and critical issue for it to overcome. Not only must each individual accept and undergo the learning process necessary to master the new interface, but the group benefits encouraging them to do so may not be available until *critical mass* has been established.

Achievement of critical mass. Communication and collaboration support mechanisms are dependent on a “critical mass” of users. Attaining critical mass is dependent on adoption by a *sufficient* group of individuals. *Sufficiency* in this context has many interpretations contingent on the group, individual, and task requirements: for example, in one group task the main factor might be the number of collaborators using the system, while in another, the involvement of particular individuals could be the main determinant.

Adoption by individuals. Individuals will be encouraged to adopt a system if there is an established base of regular users. A community of users ensures that information about the system and how to use it will be readily available, and consequently helps to break the inertia-driven maintenance of current working practices.

Personal adoption of systems will be further encouraged if the rewards for doing so are clearly apparent, both in immediacy (see the description of *Benefit-lag* above), and *who* receives the benefits: *personal* use is most likely to be stimulated by *personal* benefits.

The vicious circle. Examining the relationship between these issues forming the vicious circle in an anti-clockwise direction: critical mass depends on adoption by individuals; adoption by individuals is encouraged by system borne benefits; but the benefits are, in turn, contingent on a critical mass of users!

²Similar observations on people’s reluctance to change their working methods are made in (Norman, 1987).

The vicious circle illustrates the problems encountered by many groupware systems. Although *potentially* capable of enhancing the efficiency of organisations, they have failed to do so because their many-faceted dependencies obstruct the attainment of critical mass, the realisation of benefits, and the encouragement of personal use.

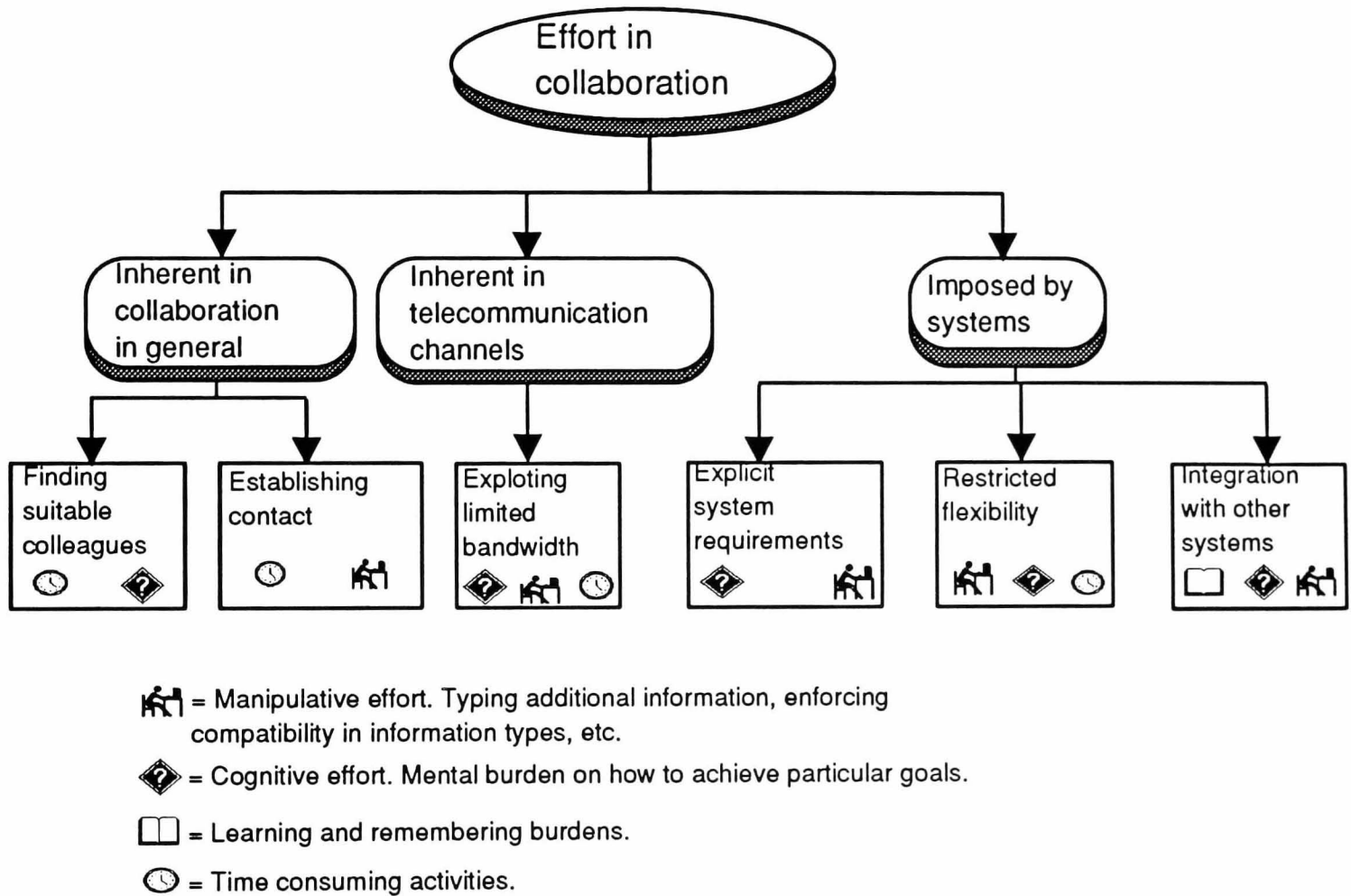


Figure 3.2: Summary of effort in collaboration.

3.3.1 User-effort: the key determinant

Figure 3.1 illustrates the dominating role user-effort plays in the initial acceptance of groupware. As mentioned above, system's prospects for adoption by individuals, and thus its likelihood of achieving critical mass, is dependent on the level of effort involved in using it. Many systems also *depend* on additional work to provide their benefits (see the description of explicit system requirements, section 3.7); should this work be omitted the benefits motivating each individual's adoption will be unavailable.

Minimising user-effort, therefore, greatly enhances a system's potential for success. For

this reason, the sources of user-effort, and groupware's *dependence* on it, are the primary considerations in the remainder of this chapter.

The sources of effort encountered in collaboration are depicted in figure 3.2. The effort of collaboration in general (shown in the figure's left-hand branch) is briefly discussed in section 3.4. The effort resulting from the use of telecommunication channels in mediating collaboration (the middle branch) is discussed in section 3.5. The user-effort resulting from groupware systems, shown in the right-hand branch of figure 3.2, is the focus of sections 3.6 to 3.9 which each describe a sub-branch of system imposed user-effort.

3.4 Effort inherent in collaboration in general

There are many social factors that can inhibit and discourage collaborative work. Many of these factors may be considered to increase "effort": personality clashes, for example, can certainly make collaboration burdensome. Social complications in collaboration are largely beyond the scope of this investigation.

The importance of *effort* in the formation and maintenance of collaborative relationships is widely recognised (Kraut *et al.*, 1988a; Ishii & Miyake, 1991; Cooper, 1991; Cockburn & Greenberg, 1993). The most likely partners for collaborative work are those with whom rich communication can be easily established: the *effort* is minimised. Close physical proximity provides the opportunity for rich, face-to-face, communication with minimal effort: perhaps a walk down the corridor. Several groupware applications use metaphors of close physical proximity to increase social interaction between colleagues. The underlying assumption is that increased social interaction leads to greater work productivity (Hirschheim, 1986): for a review of these systems, see section 2.5.3. TELEFREEK, described in chapter 6, uses base-entry-level technology to provide social-awareness facilities. It reduces the effort involved in finding suitable communicants and in subsequently contacting them.

3.5 Effort inherent in use of telecommunication channels

All groupware that supports non face-to-face collaboration is prone to limitations arising from the telecommunication channels used. The richness of interaction is inhibited by restrictions in channel bandwidth, and consequently, greater effort is required by collaborators to com-

municate information.

In face-to-face interaction subtle gestures perform a substantial communicative role (Smith *et al.*, 1991; Tang, 1991): for example, confirming that arguments have been adequately conveyed. The absence of such gestural queues when telecommunication mechanisms are employed commonly results in either excessive or inadequate explanation, thus increasing the effort required to sustain communication. The obvious solution to this problem is to increase the communication channel's bandwidth, allowing richer forms of interaction such as full motion video. However, research has indicated that high-bandwidth channels, although beneficial, are less so than might be expected (Gale, 1991; Bair, 1989; Minneman & Bly, 1991; Fish *et al.*, 1992), and the expense is prohibitive. An alternative approach is to use other means to communicate gestural information: *GroupSketch* (Greenberg & Bohnet, 1991), for example, explicitly provides gesture icons in its shared drawing environment (see section 2.4.2).

Switching between competing communication channels also requires user-effort. Communications initiated on one medium frequently continue on another (for example, in personal experience, many network Unix `talk` or `write` conversations are transferred to the telephone). Computer systems, and groupware in particular, have largely ignored channel switching: this issue is examined in chapter 6.

3.6 Effort imposed by systems

The sources of effort imposed on users by their work-support systems are shown in the right-hand branch of figure 3.2. The next four sections describe each sub-branch of system imposed user-effort in turn.

Explicit system requirements are described in detail in section 3.7, using enhanced asynchronous messaging systems as the primary example. The effort derived from the lack of integration between systems is examined in section 3.9, and the impact of restricted flexibility is described in section 3.8.

3.7 Explicit system requirements

Many systems explicitly require additional effort from users in order to support their functionality (Cockburn & Thimbleby, 1992b). Usually this effort takes the form of structured

information: such information will be termed *guidance*. Systems that require guidance for their successful operation will be termed *guidance-dependent*.

Guidance-dependence is best exemplified by the wide range of applications that support and enhance asynchronous messaging (see section 2.2). Based on electronic mail, these systems aim to improve message management and project coordination. Through increasingly sophisticated mechanisms users are provided with:

- filtering schemes—reducing information overload, prioritizing mail, and so on;
- passive conversational representation of message relationships;
- active coordination support—tracking the status of collaborative work commitments, enabling active assistance with, for example, project management.

To assist the identification of their guidance-dependence, the mechanisms used by each of these system types are briefly described below.

3.7.1 Filtering Schemes

Message filtering schemes, such as the *Information Lens* (Malone *et al.*, 1988) and ISCREEN (Pollock, 1988), ease the problems of information overload (Denning, 1982; Hiltz & Turoff, 1985). Typed message templates add structure to email, and prompt message senders to fill in fields relevant to the selected template-type. Incoming messages can then be filtered through rule-action pairs allowing pre-described actions to be executed on the receiver's behalf. For example, messages from Joe Bloggs with the subject `squash ladder` could be automatically deleted, and those from Tom Smith might be assigned to an Urgent directory.

The success of filtering schemes is dependent on message senders' altruistically carrying out additional actions: at least selecting appropriate templates, and filling in relevant fields. If accurate guidance (structured information) is absent the system's ability to filter messages will be severely limited or absent.

3.7.2 Passive conversation support

Conversational facilities augment email's role as a medium for collaborative and coordinated work. While message filtering is essentially curative (easing existing information management

difficulties), conversation-based groupware enables new methods of working, allowing the review of past collaborative efforts, improving the maintenance of group focus, and providing a platform for basing decisions and arguments (Romiszowski & Jost, 1990). A review of passive conversation groupware is given in section 2.2.2.

Like message filtering schemes, passive conversation groupware is dependent on users executing additional actions (supplying guidance) to specify the conversational action made by their messages. However, the absence of guidance is more serious than filtering systems, in which only the message receiver(s) are affected. The conversational representation provided by passive conversation systems is shared between conversation participants, consequently, if a message fails to carry correct guidance the entire group is affected by the system's inability to assess conversational context. For the system to maintain (or recover) accurate conversational representation the message's context must be manually established by whatever means supported. Such actions impose an additional burden on one of the message receiver's or (more likely) a third party system user.

3.7.3 Active coordination support

Passive conversation groupware allows users to review discussions so that, for example, the context of group decisions can be examined. In contrast, active coordination systems use a "knowledge" of work processes to support an active role in work coordination.³

Messages communicated through these systems are tracked, and their impact on the status of work commitments is automatically noted. The theory is that resultant commitment awareness enables the system to provide timely reminders, observe who is responsible for delays, and (controversially) attribute blame. The active role is typically enabled by a state transition model that formalises the processes of coordination: for example, see "conversations for action" (Winograd, 1987) and figure 2.6.

The guidance requirements of active coordination systems are similar to those of passive conversation systems. However, because the state transition model is used to play an *active* role in coordination, the consequences of absent guidance are more serious. Should the system's "knowledge" of the status of commitments become corrupt, the actions it autonomously takes will be incorrect: sending reminders to the wrong people or at the wrong

³See section 2.2.3 for a more detailed discussion.

time, incorrectly attributing blame for commitment defaults, and so on.

The problems encountered by active coordination support are best exemplified by one of the earliest groupware systems: *The Coordinator* project management system. The subject of many evaluative studies (Erickson, 1989; Carasik & Grantham, 1988; Henninger, 1991; Bullen & Bennet, 1990; Grehan *et al.*, 1991) user descriptions of *The Coordinator*'s support ranged from "facist" (Erickson, 1989) to "worse than a lobotomized file clerk" (Carasik & Grantham, 1988). Although *The Coordinator* could maintain the status of commitments if *all* communications contained correct and complete guidance, many factors make this situation an unrealistic ideal (described in the next section).

Active coordination groupware suffers from a fundamental dilemma: it aims to enhance efficiency in project management, but can only achieve it at the cost of inefficient working practices in which all communications are manipulated into system accessible formats.

3.7.4 Guidance dependence in asynchronous messaging systems

Each of the three schemes described above (message filtering, passive conversation support, and active coordination support) is dependent on explicit-user-actions (*guidance*) in order to operate successfully.

In this section the reasons for groupware's inability to access the guidance it requires are examined from two perspectives:

- the communications perspective — focuses on mechanisms from which communications originate, and their ability to convey guidance;
- the cost/benefit disparity perspective — examines the user's motives and rewards for supplying guidance.

Communications perspective on guidance availability

Figure 3.3 shows the subsets of communications relevant to a guidance-dependent system. The concentric circles in the figure are described below, from the innermost, outwards:

- The subset of communications containing complete and accurate guidance. These messages will be automatically and correctly captured by the system.

- Communications transferred through the system. Some will carry no guidance (perhaps due to work pressure, or any of the reasons discussed in section 3.7.4), and others will contain inaccurate guidance (perhaps due to mis-typing, or malicious and malevolent behaviour).

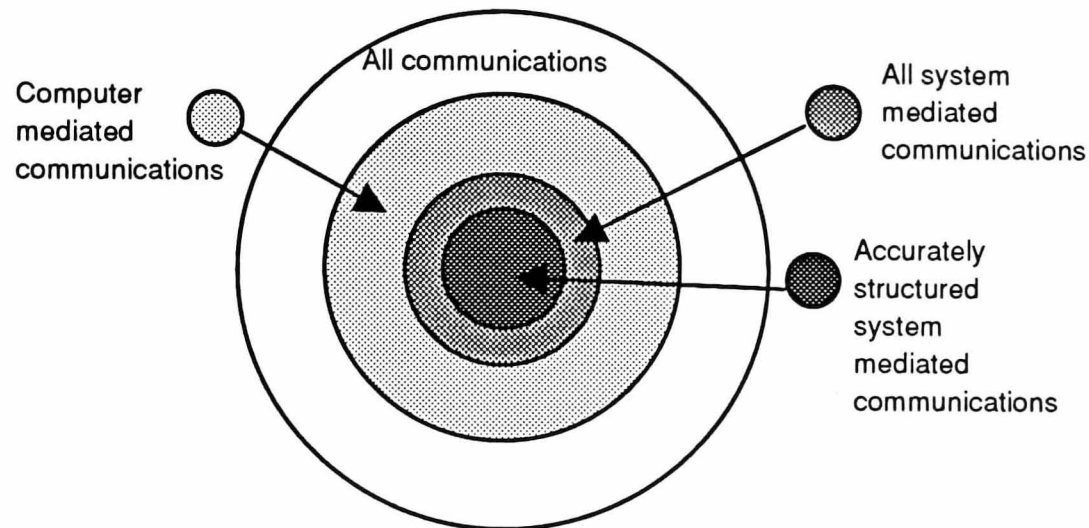


Figure 3.3: Subsets of communications and their formats.

- Computer mediated communications. Although some users may be willing and able to provide guidance some of the time, it is unreasonable to expect all users to carry out these actions all of the time. It may even be *impossible* for some colleagues to carry out the necessary actions, particularly when working at different locations on systems that use different guidance structuring schemes. The problems of incompatibilities between guidance structures are investigated in (Lee & Malone, 1990).
- All communications. Computer systems cannot (to date) capture messages communicated by non-computer media. Despite this, passive conversation and active coordination systems effectively require that they do. In order to maintain the integrity of their conversation or commitment knowledge *all* communications (relevant to the project, decision process, and so on) must be captured by the system, regardless of their origin. Consequently, these systems demand that someone, perhaps a third party user, must transfer *all* communications into a format that is accessible to the system.

Of these four message subsets, guidance-dependent groupware works efficiently in the centre (smallest) subset in which message senders' accept the burden of providing guidance for the benefit of others. At all other levels, the inability to autonomously capture guidance

requires that, for the benefits to be realised, some other user must undergo the effort of manipulating the message to “fake” its arrival with the appropriate guidance.

Cost/benefit perspective on guidance availability

The communications perspective shows that the user’s *ability* to provide guidance may be restricted by the communication facilities available. Under the cost/benefit perspective the difficulties encountered by guidance-dependent groupware are shown to be exacerbated. It demonstrates that personal motivation to provide guidance is likely to be low, and consequently, the innermost subset of communications (at which guidance dependent systems operate most efficiently, figure 3.3) will be small.

People are unlikely to be willing to undergo the effort of providing guidance when they receive no benefit for doing so. And yet, such altruistic behaviour is required for the success of guidance-dependent schemes. Guidance-dependent groupware will impose a cost/benefit disparity at one of five escalating levels (illustrated in figure 3.4):

- Level 1. The situation without groupware support. No system imposed requirements, and no system borne benefits.
- Level 2. Guidance-dependent groupware used exactly as required. Message senders’ accept the cognitive burden of selecting an appropriate template (a process that can be non-trivial when messages contain several conversational moves, or discuss inchoate ideas) and the manipulative burden of filling in the relevant fields. If no suitable message template type is available a new one must be defined, or an existing one adapted. Some systems also require that messages are split to enable identification of individual conversation components—an example being two messages, one of type “Commitment-Acceptance” and another of type “Meeting-Request” for the message “OK, I’ll do X. How about lunch?”

This effort is imposed on senders for the receivers’ benefit.

- Level 3. The sender accepts the burden of providing guidance, but the receivers fail to gain the benefit. This is possible if the recipient does not support the same system (and guidance structures), or if the commitment knowledge maintained by an active coordination system is corrupt.


















	Sender	Receiver(s)	Third Party	Comment
Level 1		 		Without groupware. No system requirements or benefits
Level 2		 		Groupware used exactly as required. Sender works for receivers' benefit.
Level 3		 		Sender executes actions, but receivers' fail to gain benefit.
Level 4		 		Additional actions omitted. No benefits until 3rd party intervenes.
Level 5		 		Guidance provided, but incompatibilities inhibit benefits. Guidance by 3rd party

Figure 3.4: Imposition of inconvenience at levels of cost/benefit disparity.

- Level 4. The sender decides that the cost of providing guidance is unwarranted and fails to do so. For the benefits to be realised, a third party must execute the actions on the sender's behalf. In message filtering systems the consequence of absent guidance is an inability to provide benefits. The consequences are serious for passive conversation and active coordination systems. If the additional work is not carried out, the shared conversational representation will be corrupt, and the knowledge of commitment status will become corrupt, causing problems such as redundant or mis-timed reminders. In such systems it is *essential* that a third party executes the necessary actions to maintain an accurate representation of the conversation (or coordination) within the underlying state-transition model.
- Level 5. The final escalation in the levels of inconvenience occurs when senders *do* carry out the relevant actions, but problems of incompatible systems (as at level 3) require a third party to carry out the actions (as at level 4).

The two perspectives on guidance availability described above (communications perspective, and cost/benefit perspective) demonstrate that, even with the best of intentions, guidance-dependent support will impose a substantial additional work burden on users.⁴ This additional cost in user-effort is likely to outweigh the user's perceived benefits, and consequently

⁴ *Mona*, the system described in chapter 5, provides enhanced email management and conversation facilities without guidance-dependence.

discourage system use. The relationship between costs (in terms of effort), benefit, the establishment of a “critical mass” of users, and personal motivation for system-use were discussed in section 3.3.

3.7.5 Requirements in synchronous groupware

Identifying the explicit user-requirements imposed by synchronous face-to-face groupware is more complex than noting guidance-dependence in asynchronous messaging systems. The fluidity with which interactions proceed in synchronous collaboration hinders precise attribution of effort to specific interface and system issues. This does not imply that systems supporting face-to-face group work have been more successful. The subtlety and rapidity of communication in face-to-face meetings tends to increase the incongruity of computer support: rather than facilitating meetings, their interface requirements make them cumbersome companions to free flowing natural human communication.

In evaluating the Colab project, Tatar *et al* (1991) recognised their failure to account for the differences between human and computer-supported synchronous communication. The *Cognoter* tool of Colab was built on the assumption that “communication consists of bits of verbal or textual material passed whole from person to person” (Tatar *et al.*, 1991), page-206. This requirement did not match the natural processes of information transfer. The excessive user-effort required to adapt to the system’s communication model caused user’s to hate the system, express astonishment that it should be built that way, and rapidly abandon it.

3.8 Lack of flexibility

Several groupware systems have been based on rigid theories of cooperative tasks and processes: examples include *The Coordinator* based on speech-act theory (see section 2.2.3) and gIBIS based on IBIS theory (discussed in section 2.2.2). These rigid theories intentionally constrain the users actions, roles, and decisions for a variety of purposes that range from assistance in decision making, to facilitating an active system-role in work coordination. By imposing constraints, these systems are inflexible and require specific styles of use which may conflict with those preferred by users. Groupware may also be inflexible in the way it presents information: perhaps enforcing consensus views of information and ignoring personal

or sub-group preferences.

Groupware design is both a technical *and* sociological process. Technical perspectives appeal to technically minded groupware developers, and to managers attracted by the potential increases in organisational efficiency. Social issues, however, govern the use or rejection of groupware.

Inflexible and constraining systems are likely to be unpopular: they necessarily enforce a form of “work to rule”, a phrase synonymous with inefficient, restricted and *inflexible* working practices. Requirements for specific styles of use, and the failure to account for sub-group or personal preferences and information interpretations will discourage system use (Greenberg, 1990b; Johansen, 1988; Greenberg, 1990a; Krasner *et al.*, 1991). Although users may find ways of working around system-imposed restrictions (perhaps using alternative mechanisms to record personal views—a paper note pad for instance), such work-around strategies illuminate system inadequacies, require additional effort, and will not lead to popular groupware.

3.9 Lack of integration between systems

Sources of additional effort derived from groupware go beyond the requirements and the flexibility constraints imposed by each *independent* system. Effort is also required to manage work environments in which various tools, facilities, and communication mechanisms are used in conjunction.

In computer supported *personal* work people are required to make *transitions* (changes in their styles and methods of working) between the facilities used in their everyday work. Effort in such transitions is derived from the cognitive burden of learning/remembering separate interfaces and the burden of manipulating data into compatible formats. Interfaces maintaining a consistent “look and feel” such as those employed by the Apple Macintosh (Apple Computer, 1988) or MicroSoft’s Windows III can ease these transitions: methods and techniques used in one interface can be transferred to others, and consistent utilities such as cut, copy, and paste ease data manipulation.

When computer support is used for *group* work, additional transitions are required: between single- and multi-user applications, *and* between alternative communication mechanisms.

Typically, in focusing on the work products of groups, groupware has diverted design

attention away from personal satisfaction. The support groupware provides for personal work is therefore inferior to that of single-user software. Consequently, people will be inclined to primarily use their familiar single-user applications, and transfer their work to the groupware environment only when essential. If groupware is used infrequently, the effort required to re-learn its interface may be sufficient to discourage participation in group-work altogether. Reducing user-effort by integrating personal- and group-work environments is discussed in (Cockburn, 1991c; Cockburn & Thimbleby, 1991).

The lacking integration between groupware (and other computer tools) is not only a source of user-effort, it is also a missed opportunity for groupware. Computers have the ability to access a variety of information about communications and collaborators: they can actively initiate collaborations, and can carry out autonomous processing to establish suitable colleagues or communicants. TELEFREEK, the system described in chapter 6, demonstrates the potential of heterogeneous computer platforms for collaborative work.

3.10 Summary

In any system, single- or multi-user, a requirement for high levels of effort will discourage users. Groupware encounters additional difficulties, illustrated in this chapter by the vicious circle of groupware adoption. This vicious circle relates groupware's benefits, critical mass, and personal encouragement, and it requires that each of these properties to be simultaneously available *before* groupware can become successful. User-effort plays a governing role in the fulfilment of each of these factors.

Ascribing groupware's limited success to user-effort and its dependencies, as this chapter has done, may seem simplistic. Sociologists may execrate the failure to explicitly include the social issues that permeate all collaborative work; psychologists may condemn the absence of empirical evidence. The counter argument is one of pragmatics: if groupware is perceived to be sufficiently valuable to warrant the effort required, then people will use it. The vicious circle inhibits the realisation of the benefits that could encourage users.

This situation appears to foretell an extremely gloomy future for groupware!

What is required is a kick-start, a break in the vicious circle allowing, for instance, benefits without the attainment of critical mass. The overbearing and discouraging role of user-effort throughout system adoption and its subsequent use must also be minimised.

In chapter 4, these observations are used to develop four principles for groupware design. These principles are particularly directed at easing problems during early stages of groupware use.

Chapter 4

Four principles for groupware design

4.1 Introduction

The preceding chapter started by generalising Grudin's (1988) observations on the causes of groupware failure into three levels: system-use, system-design, and system-evaluation. It went on to investigate groupware's problems during system-use, and has provided a platform from which to develop guidance against failure in system-design.

In this chapter, four principles for groupware design are described, together with strategies for achieving their aims. At a general level, the principles raise the questions that all groupware designers must address. At a system-specific level, they provide practical guidance for overcoming the primary obstacles to groupware's success: the imbalance between the users' perceived costs and benefits; the vicious circle in groupware adoption; and the requirements for user-effort. Designing with the principles promotes the development of acceptable groupware, and enhances groupware's potential for overcoming the problems that inhibit its attainment of a critical mass of users.

4.1.1 Chapter overview

In section 4.2.1 the observations made in chapter 3, on user-effort as *perceived by users*, are mapped onto system attributes that cause the effort. This mapping provides a *system-oriented* description of user-effort, which eases the derivation of design principles.

The principles and their aims are briefly introduced in section 4.2.2. Sections 4.3 to 4.6 describe each principle in detail. The general issues raised by each principle are described, and practical strategies for achieving its aims are proposed. Where possible, the principles and their strategies are exemplified by existing groupware applications (extracted from the research literature) which display similar approaches and desired properties. *Mona* and *TELEFREEK*, described in chapters 5 and 6, provide specific demonstrations of the principles' use.

Each principle's impact on designers is discussed with the principle's description (sections 4.3 to 4.6). A summary of their impact on design and evaluation is given in section 4.7.

Section 4.8 returns to groupware's vicious circle (section 3.3) and examines how the principles weaken and overcome it.

4.2 From groupware problems to groupware principles

Chapter 3 adopted the *user's* perception of user-effort derived from the tasks he or she is required to undertake in collaborative work. In the following section, to aid the development of design principles for groupware, the user's effort is attributed to a set of system properties.

4.2.1 System oriented view of user-effort

In chapter 3, figure 3.2 divided the user-effort derived from groupware into three categories: explicit system requirements; the lack of flexibility; and the lack of integration between systems. This division lends itself to a system-oriented description of user-effort: illustrated in figure 4.1, and described below.

Figure 4.1 depicts user-effort in terms of *transitions*: changes between the user's ideal way of working and those required or imposed by tools (perhaps between groupware systems, single-user computer applications, non-computerised communication mechanisms, and so on). Transitions from the user's ideal working techniques are required in the independent use of every application, but transitional effort is increased when separate systems (each requiring its own transitions) are used together: for example, a problem familiar to many computer supported co-writers is that of switching between the interfaces and the structures supported by different word processors.

In figure 4.1, each computer system is represented by a set of concentric circles, separating

the system into three effort-causing components:

User interface — when dedicated to a single user-interface, people can quickly become familiar and expert with it. When separate systems are used, substantial effort can be imposed by the interface transitions required: remembering or relearning techniques, and so on. In this chapter, the user interface is also viewed as containing the positive and enticing system aspects that can encourage its use.

Explicit requirements — the explicit requirements imposed by groupware on users were examined in section 3.7. Groupware typically requires that users provide specific information, and that they carry out explicit actions in order to maintain the system's ability to operate correctly (this requirement was termed guidance-dependence). People must learn and remember the differing requirements imposed by each of the systems they use. They may also have to reformat information to ensure that it is compatible between the systems. If people do not not comply with guidance-dependent requirements, the system will fail to provide its intended support, and a third-party user must carry out the required actions (see figure 3.4).

System constraints — the constraints imposed by systems are manifested by the lack of flexibility encountered by users (see section 3.8). The work-styles imposed by differing systems are unlikely to be identical, and need not match any user's ideal. Switching between different system-constraints will, again, require user-effort.

The final set of transitions, represented by the dotted box in figure 4.1, is concerned with issues of integration beyond each independent computer system:

Environment transitions — computer systems, represented by the concentric circles in figure 4.1, are only part of the work environment. In addition to the transitions required by computer systems (both to them, and between them), collaborative workers must also make transitions between the other tools and mechanisms used in their everyday work. The tools and facilities that make up the work environment include telephones, fax machines, telephone directories, file cabinets, and so on. The lack of integration between computer systems and the work environment in which they are embedded demands further user-effort.

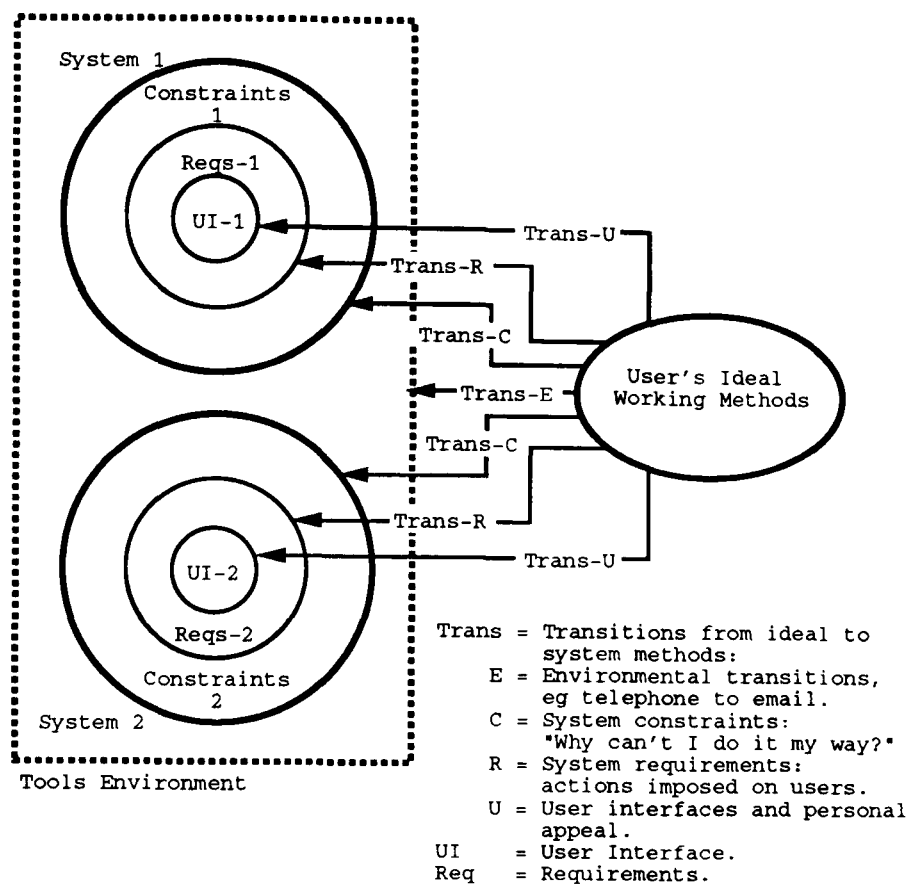


Figure 4.1: Transitions between the user's ideal working methods and those supported by work support tools.

User-effort is necessary to overcome the interfaces, requirements, and constraints of all computer systems, not just groupware. Groupware, however, has a greater reliance on users' accepting this effort as a necessary and worthwhile part of their everyday work: the vicious circle (figure 3.1) shows that personal acceptance of groupware carries implications beyond those of acceptance of single-user applications.

Viewing user-effort through transitions (as in figure 4.1) illustrates groupware's *competition* with single-user applications. Naturally groupware provides group support, but in doing so the personal work environment has been largely ignored. Consequently, when including personal work in the collaborative (groupware) environment, user's must accept and undergo the effort of inter-application transitions, for example: copying documents between single-user and groupware authoring systems.

4.2.2 Introducing the principles

Four groupware design principles were derived from the system-oriented view of user-effort (shown in figure 4.1):

Maximise the likelihood of personal system acceptance — aim to increase the perceived and *immediate* benefit of groupware.

Minimise the requirements imposed on users — explicit system requirements impose an additional work burden on users and reduce system compatibility.

Minimise the constraints imposed on users — constraints restrict the users' styles of working, and limit their flexibility in customising information input/output formats.

Maximise the potential for external integration — the role of groupware must be considered within entire work environments.

The beneficial properties of these principles should be immediately apparent: increasing benefits, reducing work, and increasing integration are “obvious” goals. Despite this, trends in groupware development demonstrate that designers' recognition of the essential nature of these properties is categorically absent. Echoes of agreement with the principles are available (in fragmented form) in selected CSCW literature, but groupware developers continue to follow developmental strategies that directly contradict these principles.

The emphasis of these principles is on design with current technology and with current understanding of group-work processes. The development of research systems (that ignore or conflict with the principles) will provide a greater understanding of collaborative work and its support, but pragmatic groupware for current use should follow the principles' recommendations.

In the following sections (4.3 to 4.6) the principles are described in detail. Together with each principle's description, a set of strategies for achieving its aims are detailed, and groupware applications that demonstrate desired properties are used wherever possible; a discussion of each principle's impact on designers is also provided . Due to the relationship between issues addressed by the principles (for example, user-acceptance and user-effort) some of the strategies are applicable to more than one principle. To avoid repetition, mention of these overlaps is minimal.

4.3 Maximise personal acceptance

Maximising personal acceptance is concerned with encouraging people to incorporate groupware applications into their work routines. Although *promoting* user acceptance may be sufficient, the provision of *benefit* is a more ambitious aim that emphasises the positive aspects of system use. The success of groupware is dependent on each user's willingness and ability to incorporate the system into personal working methods: all classes of users must be content with their role in the support provided. Each user must therefore receive a satisfactory balance between the amount of effort required to use the system, the benefit derived from it, the flexibility it supports, and the encouragement it provides.

There is a similarity in how people view systems for personal and group work (for example, a word processor and a collaborative writing system). A common question users ask about both types of tool is "what can it do for me?" During the initial stages of system use, this question will carry an additional component, "*now*". Naturally, users will be more willing to devote time and effort to learning a new system if the benefits they receive for doing so are immediately available and clearly apparent.

The personal acceptance principle argues for greater consideration of interface issues in groupware. If a task is better supported by a personal work tool than by its collaborative equivalent, users will continue to work primarily in the personal environment, overcoming transitions to the collaborative tool only when necessary. In addition to improving groupware user-interfaces, strategies for encouraging personal acceptance (particularly during early stages of system use) include "feature ticking", and the use of groupware "champions". The "reflexive perspective" argues for increased design attention on the individual, basing this argument on the group-like behaviour that people display when coordinating their personal work.

4.3.1 Catchpenny systems

Feature ticking (Thimbleby, 1990a) is a sales ploy used to add instant appeal to a wide range of modern products. Attractive features and additional facilities supplement the functionality of a variety of goods, turning attention away from the key task and onto fancy bells and whistles. While not condoning the design of poor (but feature rich) systems, a form of feature ticking can be used to supply instant user-appeal. Subtle forms of feature ticking can encourage and reward exploration of system facilities: for example, the sequence of commands necessary to

laser-print a document might be executed by an item in a sub-menu.¹

Although personal benefits can encourage system use, groupware's acceptance is also dependent on the costs incurred through its use (section 3.3). Feature ticking facilities, which are specifically designed to attract individuals, will be worthless in the collaborative context if they fail to forward group work. To this end, the "reflexive perspective" of CSCW reduces the distinction between personal and group support.

4.3.2 The "Reflexive Perspective" of CSCW

The reflexive perspective of CSCW (Thimbleby *et al.*, 1990; Cockburn, 1991c; Cockburn & Thimbleby, 1991) argues for greater integration between personal and collaborative work environments. This argument is motivated by the following observations:

- There are similarities between the coordination requirements of personal work and group work.
- Users can experiment and become familiar with communication facilities *reflexively*—they can direct their communications at themselves to gain confidence with the support provided.
- Many other benefits are made available by merging personal-work and group-work environments.

The following sections examine the similarities between the management requirements personal-work and group-work, and some of the advantages enabled by merging their support are reviewed. For a further discussion of these issues, see (Cockburn, 1991c; Cockburn & Thimbleby, 1991). The advantages of integrated work environments (including the merging of personal and group work) are further examined in section 4.6.

Distribution of personal work

CSCW concentrates on providing support for group work that is distributed through time or space or both. Reflexive CSCW (Thimbleby *et al.*, 1990; Cockburn & Thimbleby, 1991) observes that personal work may similarly be distributed through time and space.

¹Carroll (1987) uses an adventure game analogy in discussing potential ways for improving interface design and instructional techniques. The enticing appeal of user-interfaces is discussed in (Bloomer & Ingram, 1992).

Each individual may work on several machines (one at the office, one at home, a lap-top, and a secretary's machine) and several projects may be pursued at different times. With multiple tasks, and work places, the individual's coordination requirements are similar to those of asynchronously collaborating co-workers.²

Each person's group-like work coordination requirements are further exposed when their separate work coordination roles are considered. These roles include:

- *The management role* — before starting work, people must carry out work management tasks, such as deciding what to do, coordinating the necessary resources, establishing reminders to prompt further work on tasks (for instance, when a colleague sends some necessary information), and so on.
- *The worker role* — in which the actions necessary to advance or complete the work are carried out.
- *The meta-management role* — when a person has an assistant (human or computerised), the assistant must be instructed: for example, detailing what tasks are to be carried out on the persons behalf, and whether the actions should be carried out autonomously or should be notified.

Merging personal and group work environments

By merging support for personal and group work, the same or similar techniques are used across a wider range of tasks. Consequently, users' benefit in several ways:

- Greater familiarity, trust, and predicatability in the system's interface and functionality. These properties arise from a consistent interface across the personal and collaborative work environments. When faced with a new communication tool, users may be reluctant to collaborate due to risks of embarrassment should their lack of proficiency be displayed to others. Reflexive communication facilities allow new users to experiment with the facilities within personal work environments (directing their communications at themselves).
- Skills transfer from one environment to the other.

²For a discussion of related problems in distributed personal and group work, see (Dix & Beale, 1992).

- The effort of learning and remembering separate interfaces is reduced.
- Increased awareness of involvement in a chain of collaborative commitments. Work considered to be personal is often derived from or dependent on external sources, and colleagues may be reliant on a person's "personal" work.³
- Enhanced personal appeal—generally, the reflexive perspective emphasises the individual's role in collaboration. It therefore stresses the importance of providing *personal* appeal, and of catering for individual preferences or requirements. Methods for doing so have been discussed in section 4.3.1, and are further elaborated in section 4.6, and in (Cockburn, 1991c; Cockburn & Thimbleby, 1991).

4.3.3 Champions and encouragement

The "personal acceptance" principle is primarily concerned with ensuring designers attend to personal issues during groupware *development*. The principle can, however, be maintained during groupware's installation and use to encourage its adoption. "Champions" or "evangelists" can play a substantial role in promoting use of technology (Eveland & Bikson, 1988; Ehrlich, 1987; Francik *et al.*, 1991). These system exponents raise awareness of what groupware can achieve, and generally encouraging its use. In their study of decision making through email conferences, Fafchamps *et al* (1991) noted: "...the single most important factor for a successful computer conference is the activity level of the organiser of the conference", page-220.

4.3.4 Participatory design

Groupware designers should, whenever possible, actively involve end-users in the design process. "Participatory design" (Greenbaum, 1988; Clement & Gotlieb, 1987; Muller *et al.*, 1991; Kyng, 1991) assists in avoiding groupware based on either misguided intuitions about social processes in the work-place (on the part of designers), or inappropriate requirements specifications provided by a subset of the user community (typically, management who order the groupware product). Involving all end-users in groupware design is likely to foster a sense of

³An advantage of the otherwise unpopular *Coordinator* (see sections 3.7.3 and 2.2.3) was that it made explicit the personal involvement in collaborative commitments (Winograd, 1987).

group responsibility for the system's functionality, and encourages a better fit between the system, the existing cooperative processes, and the new social processes brought about by groupware (Kyng, 1991).

4.3.5 Implementing personal acceptance

To summarise, the primary aim of the personal acceptance principle is to promote direct, immediate, and personal benefits, that are independent of other users.

The strategies described for achieving personal acceptance have implications for groupware's functionality and interface.

Functionality — although the reflexive perspective observes similarities between personal and group work, it argues that the similarities should be exploited to reduce boundaries between personal and collaborative support (rather than basing collaborative support around personal tools).⁴

User-interface — the importance of encouraging, enticing, and appealing to users have been stressed by the strategies. Borenstein and Tyberg (1991) concur that a “highly polished and usable interface” is a fundamental requirement of groupware; furthermore, they state that the traditional design trade off between power and usability is inappropriate and damaging in group work support. To cater for diverse expertise levels among users, power *and* ease of use are mutual necessities.

Designing for personal acceptance is, therefore, not the same as designing single-user interfaces. The principle demands much from groupware designers: not only must they provide the relevant functionality for the group task, they must do so in a highly usable manner, and without compromising experts' or novices' requirements.

4.4 Minimise requirements

The minimise requirements principle addresses the main component of system imposed user-effort: explicit system requirements (see figure 3.2, and section 3.7).

⁴The coercion of single-user applications into groupware support is discussed in (Greenberg, 1990b), and general “reflexive” issues (the “personalization” of groupware) are raised in (Greenberg, 1990a); Patterson (1991) provides a comparison between the requirements of personal and group applications.

User-effort plays a pivotal role in system adoption (see section 3.3), but a system's *dependence* on user-effort has detrimental effects beyond issues of initial system use. These effects include the cost/benefit disparity between those carrying out actions and those receiving the benefit (section 3.2), and system incompatibilities due to dependence on particular information structures and formats.

Minimising requirements promotes the development of systems with reduced cost/benefit disparities and increased compatibility. Strategies for achieving this goal, detailed below, include avoiding dependence on additional work (section 4.4.1), utilising information inherently available through communication (section 4.4.2), and enabling shifts between those accepting and receiving the costs and benefits (section 4.4.3).

4.4.1 Avoid dependence on user actions

In section 3.7, groupware's dependence on explicit user actions was examined. The actions upon which systems depend were termed *guidance*, and systems dependent on guidance were termed *guidance-dependent*. It was noted that guidance-dependent systems impose a cost/benefit disparity across users, and that there is little personal motivation for carrying out the required actions.

Rather than *requiring* guidance, a more acceptable (and pragmatic) approach is to provide benefit when guidance is present, while not depending on it. It has however been argued that such relaxations of guidance-dependence are impractical due to the inter-relations and dependencies inherent in collaborative work:

“Can a CSCW application succeed if doing the extra work is left to individual discretion? Unfortunately, *probably* not.” (Grudin, 1988) page-86 (my emphasis).

While Grudin's observation is *probably* correct, the corollary that groupware should therefore require users to provide the additional work, will leave systems susceptible to groupware's vicious circle: section 3.1 detailed the severe problems encountered when systems depend on and require actions from users. Guidance-dependence is as likely to cause system rejection as leaving work to individual discretion.

The “necessity” of structured information is further noted in (Rodden & Sommerville, 1991):

“An underlying requirement within cooperative working support systems is the need for some structuring facility upon which to construct information handling systems,” page-161.

Certain forms of user-support will, indeed, be dependent on structured information. The major problem for system designers in supporting this structured information is *how* to retrieve it. The user is the most readily accessible and accurate resource, but research and experience has shown that systems can profitably look elsewhere: see section 4.4.2, and (Crow & Smith, 1993; Kozierok & Maes, 1993; Cockburn & Thimbleby, 1992b).

If explicit user guidance is the only potential information resource, then this strategy would caution against that form of support: it would necessarily impose a cost/benefit disparity (see section 3.2).

4.4.2 Use information that is available “for free”

Avoiding guidance-dependence raises conflicting desires in groupware design:

- groupware requires additional structured information to increase the efficiency of group work;
- however, avoiding dependence eliminates (or reduces) groupware’s access to the most obvious source of this structured information—the user.

Under this strategy, these conflicting ambitions are addressed by encouraging groupware designers to use alternative, “free”, information sources.

Although additional structured information is required to provide certain types of benefit, guidance (explicitly provided by users) is not the only information source. Information is often accessible to computers through the process of communication. Email messages, for example, contain headers that reveal the “who”, the “when”, and the “where” information about a message, and can also detail (among other things) the subject matter, and the message’s relationship with previous ones.⁵ Another source of “free” information in text-based communication is the text itself, which can be scanned by natural language parsers or simply

⁵ *Mona*, described in chapter 5 uses the “free” information in standard email headers and inferencing heuristics to provide enhanced facilities.

searched for keywords: MAFIA (Lutz *et al.*, 1990) and *Strudel* (Shepherd *et al.*, 1990) demonstrate such schemes; and “Latent Semantic Indexing” (Foltz, 1990; Foltz & Dumais, 1993) allows automatic generation of a messages “latent” semantic content which can be used to prioritise, filter, and retrieve messages based on the user’s assessment of previous messages with similar LSI values. These systems are more fully described in section 2.2.1.

Statistical information is a form of “free” information that is not restricted to text-based collaboration. Statistical information can be used to infer work-patterns, tasks, and knowledge in collaborative work: for example, the frequency of previous communications could be used to predict new communications, to notify that expected communications are overdue, and to prompt responses. Related issues, using interaction pace in CSCW, are discussed in (Gordon *et al.*, 1985; Dix, 1992c), and predictions based on the statistics of previous user behaviour are examined in Darragh and Witten (1992).⁶

4.4.3 Equal opportunity: enable shifts in cost and benefit

Designers and managers who strive for efficiency-enhancing groupware have, typically, assumed that people are willing to work for the benefit of others (Grudin, 1988; Bikson & Eveland, 1989; Nagasundaram, 1990). This assumption ignores social affects, including the users’ reluctance (or inability) to carry out actions that provide no *personal* benefit (see figure 3.4).

Equal opportunity, as applied by this strategy, reduces groupware’s imposition of additional work on a particular set of users. It increases the freedom of choice in deciding who provides “guidance”, and eases groupware’s cost/benefit problems.⁷

Rather than requiring one particular set of users to work for the benefit of others, under equal opportunity *any* set of users can execute the work. In message filtering systems, for example, (section 3.7), equal opportunity would allow various schemes for providing guidance:

⁶The use of “free” information sources is a common approach in supporting adaptable and intelligent interfaces.

⁷This strategy is adapted from Thimbleby’s (1986) paper which uses “equal opportunity” to reduce system distinctions between input and output. Thus, Thimbleby’s equal opportunity removes “assumptions about which side of the interface must supply a particular piece of information”, page 13. An equal opportunity calculator (for example) would allow the equation “5 + [] = 10” to be entered, and the system would fill in appropriate blanks.

- The sender provides guidance for the receivers' benefit—this is the guidance-dependent scheme usually adopted by message filtering groupware.
- The receiver works for the sender's benefit—the receiver's classification of incoming messages is made available to the original sender (perhaps by autonomous reply, or through shared work spaces).⁸
- Reflexive benefits—sender's or receiver's classify messages for their own benefit (these classifications might be accessible by others).

Equal opportunity can also ease problems arising from the organisational and social factors that determine whether it is appropriate to expect others to supply guidance. Guidance-dependent systems demand that *all* users satisfy system requirements regardless of their relative organisational status. Although it may be reasonable to expect subordinates to work on behalf of managers, the converse may not be true. Furthermore, although it may be reasonable to expect users' to supply guidance, it may not be perceived by users as "reasonable to demand". Related social issues in collaboration support are examined in (Erickson, 1989; Robbins, 1990; Holleran & Haller, 1990; Eveland & Bikson, 1988). Groupware's requirements should not inhibit the ability of social protocols (Gibbs, 1989) to govern cooperative work. Collaborating workers should therefore be allowed to resolve conflicts between expectations of actions at one level and execution of actions at another.⁹ Through equal opportunity both the ability to work on behalf of others *and* the flexibility (available through self motivated work) are supported. Consequently, the level to which systems impose on their users is reduced.

4.4.4 Strategies for minimal requirements in conjunction

The strategies for minimal requirements (detailed above) are not intended to replace guidance-dependent schemes; rather, they should be used to supplement them. They free users from the imposed cost/benefit disparity; they increase potential for system compatibility (by reducing the dependence on specific information formats); and they enhance flexibility by enabling users to work for personal benefit.

⁸ *Mona*, described in chapter 5 supports a shared work-space version of equal opportunity.

⁹ Social and system-imposed protocols will be examined in section 4.5.

When used in conjunction, the strategies provide multiple mechanisms for accessing the structured information that is necessary to support certain enhanced collaborative work facilities. For example, in message filtering systems, “avoiding dependence on user actions” can free message-senders from additional work, but it is only when this strategy is combined with “utilise free information” that message receivers stand to benefit regardless of the sender’s actions.

Figures 4.2a to c and 4.3d and e, represent the cost/benefit trends and aims of the strategies for minimised requirements. The purpose of these graphs is to clarify the strategies intentions; they are not the product of empirical evaluation.¹⁰ When considering guidance-dependent messaging systems, the x-axis represents the percentage of relevant messages that contain accurate guidance when initially sent (the sender supplies it). The y-axis shows the overall value of the system (as would be perceived by the society of users); positive value represents system-borne benefits, negative value represents additional costs.

Guidance-dependent systems: shown in figure 4.2a. Until a certain percentage of communications contain accurate guidance, it is likely that guidance-dependent systems will impose additional costs on users. The “worst case” cost, shown by B_w , depends on the style of support provided.

- In message filtering schemes B_w will be close to zero because the sender’s failure to provide guidance does not impose on other users.
- In passive conversation and active coordination systems (sections 3.7 and 3.7.3), B_w could be substantial because *all* messages must be converted to a format accessible to the system—otherwise the system’s knowledge of the current status of commitments (and its ability to support users) will become corrupt. During the early stages of system use, users have little motivation for supplying guidance (see section 3.3), and errors in providing guidance are likely. During these periods, the discouraging affect, represented by B_w , is likely to play a major role in system rejection.

¹⁰The principles provide generally applicable design guidance, and the strategies aim to highlight relevant issues for groupware developers. Providing data on empirical evaluations of existing groupware would not greatly assist in achieving this goal.

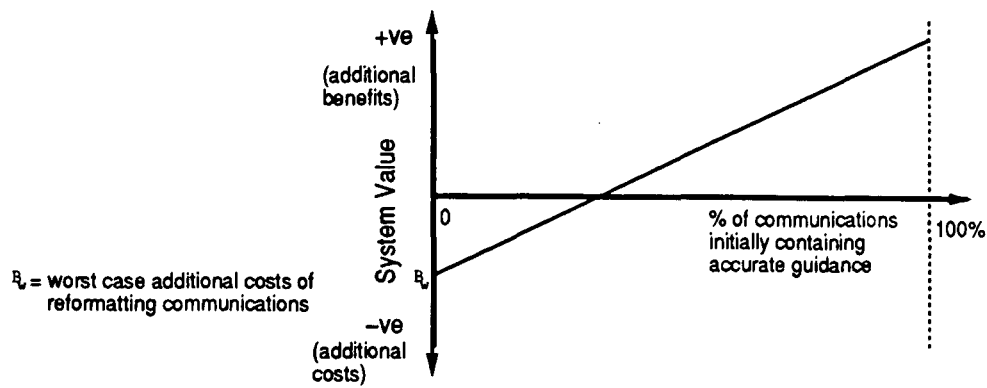


Figure a: Guidance-dependent systems.

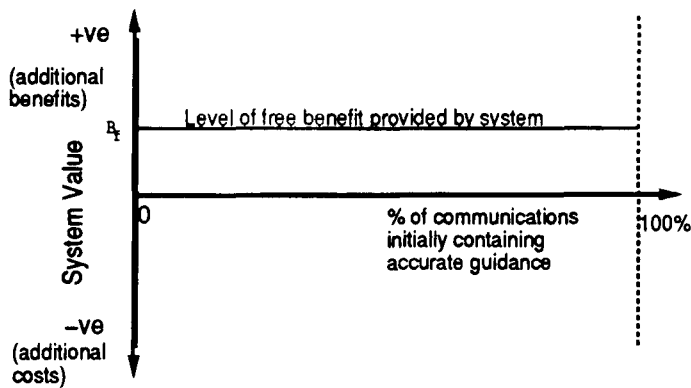


Figure b: Guidance-free systems.

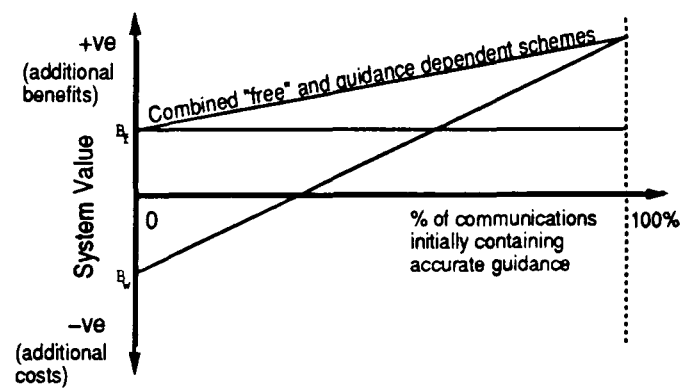


Figure c: Guidance-dependent, guidance-free, and combined systems.

These figures present a guide to the possibilities of the strategies, and render a simplified version of actual cost/benefit parameters; however, the general trends, which the figures clarify, are important for designers to recognise.

Figure 4.2: Comparative benefits with guidance-dependent and guidance-free strategies

Guidance-free systems: shown in figure 4.2b. Following the “use information that is available for free” strategy (section 4.4.2), systems could achieve a constant level of benefit (B_f), regardless of the actions taken by other users. This approach is demonstrated by *Mona* (see chapter 5).

Combined guidance-dependent and guidance-free approaches: shown in figure 4.2c. Combining the approaches shown in figures 4.2a and 4.2b, systems adopt “guidance free” mechanisms when guidance is absent, but use it when available. Consequently, benefits are available to users regardless of the presence of guidance. Such combined approaches will be particularly important during initial system use: encouraging users, and potentially prompting them to provide guidance enabling further benefits.

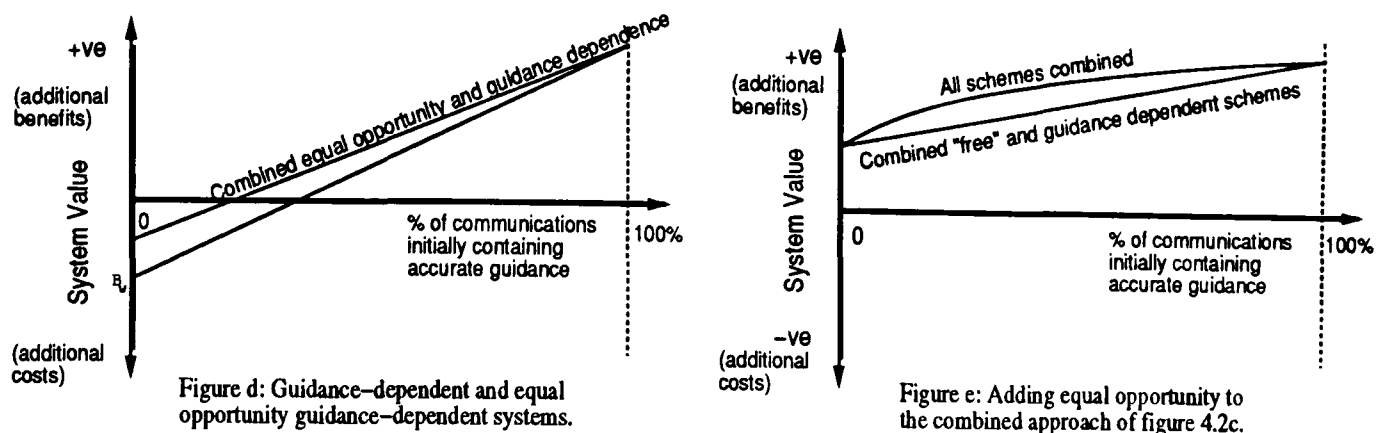
Incorporating equal opportunity. The the combined strategies detailed above (figure 4.2c) fails to account for several factors in collaboration, including the social standing of colleagues (and whether it is reasonable to expect them to provide guidance), and the fact that colleagues may be unable to provide guidance (perhaps due to incompatible systems). Incorporating equal opportunity guidance schemes in groupware reduces dependence on guidance being accessible from specific sources. The level to which systems impose on their users is therefore reduced, with a consequent reduction in the value of B_w when equal opportunity is combined with guidance-dependence strategies (figure 4.3d).

Combining equal opportunity with the combined schemes shown in figure 4.2c is likely to achieve benefits more rapidly, while asymptotically realising the same value when all communications contain explicit guidance (figure 4.3e).

4.4.5 Implementing minimise requirements

When groupware facilities require additional information it is natural that designers should look to the user to provide it. This principle, for minimised requirements, has two main suggestions. First, there *are* other sources, and the utmost use should be made of them. Second, if other sources are not available, designers must consider whether the system’s facilities will be sufficient to entice users *before* the user-society supplies guidance.

In developing a system that combines strategies (such as guidance-dependent, guidance-free, and equal opportunity) designers must, in effect, develop two (or more) integrated sys-



These figures present a guide to the possibilities of the strategies, and render a simplified version of actual cost/benefit parameters; however, the general trends, which the figures clarify, are important for designers to recognise.

Figure 4.3: Comparative benefits with the equal opportunity strategy

tems. Guidance-free schemes are likely to draw on heuristic methods which cannot ensure the validity of inferences made; they must therefore allow modification and correction. Equal opportunity schemes also require flexible mechanisms for capturing guidance, and its implementation is likely to be more complex than that of rigid guidance-dependent schemes.

Minimising requirements, therefore, demands additional work from designers. Although this may discourage designers from using it, the additional work is not a direct consequence of the principles; rather, it is a necessary design requirement in overcoming the usage problems encountered by groupware. The designers' additional work will be discussed further in section 4.7.

4.5 Minimise constraints

Minimising requirements is concerned with the implementation stage of groupware development. It focuses on *how* systems retrieve the information they require, and aims to avoid a dependence on user actions. Minimising constraints attends to problems arising at an earlier and more abstract level of system development. It examines the models and theories underlying groupware's support. The aim is to avoid inflexible and constraining styles of use.

There is a causal relationship between the imposition of user-constraints and resultant user-requirements. Explicit models of group work processes (for example the state-transition-networks underlying the *Coordinator*, see section 2.2.3) intentionally constrain the user's flex-

ibility, and require additional user-supplied information (guidance) for correct operation. Designers implementing such models must ensure that their system can secure the required information, and the user is the most obvious source—hence the imposition of requirements.

Constraining users to particular styles of use necessarily inhibits their flexibility. Although rigid working practices can, in principle, support highly efficient organisations, in reality few organisations operate according to such deterministic methods; furthermore, they cannot be made to do so (Nagasundaram, 1990). The minimise constraints principle concurs with the aims of Dykstra and Carasik (1991) who, during development of the *Amsterdam Conversation Environment*, considered “...what it was that we really wanted to support: processes or people?”, page 420. Their definition of support for *people* as “non-dependency-creating enablement” argues for groupware that leaves users free to develop protocols governing collaborative work as *they*, rather than their systems, see fit.

Strategies for achieving minimal constraints, detailed below, primarily aim to increase designers’ awareness of problems arising from inflexible and rigid systems. Specific and detailed strategies are likely to be inappropriate due to the diversity of the models implemented by groupware.

4.5.1 Be aware of the two level perspective of technology

Sproull and Kiesler’s (1991) two-level perspective of technology examines conflicts between the increased efficiency enabled by groupware, and groupware’s negative social implications (see section 3.8). The first level addresses the increased efficiency enabled by particular styles and uses of technology. The second level is concerned with social effects, raising issues such as user acceptance, personalised views of information, and individual preferences. The distinction between these levels can be expressed by the contrasting questions “what is *possible* with technology?” at the first level, and “how will it be used?” at the second.

Groupware designers, and all those involved in groupware development, must be aware of the social implications inherent in group work support. Technology that is capable of enhancing organisational efficiency will fail if the relevant social factors are ignored. Design alterations based on projections of a system’s social implications may temper the efficiency improvements attainable, but it is better to provide acceptable mechanisms providing some benefit than unacceptable ones which, despite great potential, fulfill none.

4.5.2 Beware of explicit models and theories of collaboration

Models and theories that are explicitly embedded into systems should be capable of supporting flexibility, completeness, and dynamic adaptability in a manner acceptable to all users. Developing such models/theories is likely to require substantial research effort (more than an equivalent system developed in an open, non-dependency-creating, manner). This difficulty has been illustrated by the failure of many systems adopting rigid models and theories, most notably the *Coordinator* (Carasik & Grantham, 1988), and *Cognoter* (Tatar *et al.*, 1991). In both these systems, the restrictions imposed by their underlying theories of coordination and communication were the major factors causing their failure (see sections 3.7.3 and 3.7.5).

CSCW research into collaborative activity promises to yield workable explicit models for groupware in the future. However, the lack of maturity and incomplete state of this research makes the use of explicit models in *current* groupware largely inappropriate.

4.5.3 Open, unconstrained enhancement

While models and theories of collaborative activity are under development, open and unconstrained systems allow users to develop protocols as they see fit. User-specific models might be used to supplement an open system, but they should not impose constraints on collaborative tasks or on their mediation. For example, user-models might record and install specific user-preferences, perhaps enabling tailored views, catering for personal hardware, and so on.

Several existing groupware applications exemplify open and unconstrained group enhancement. They support a variety of collaborative activities, described below:

- Several social browsing and virtual presence systems allow social protocols to prevail: for example *Cruiser* (Fish *et al.*, 1992) and *Portholes* (Dourish & Bly, 1992), see section 2.5.3.
- The *TeamWorkStation* (Ishii & Miyake, 1991) reduces the boundaries or “seams” in computer supported collaboration, and achieves this through a shared workspace which “is required to be ‘open,’ in the sense that no new piece of technology should block the use of already existing tools and methods”, page-39. See section 2.5.2.
- The *Object Lens* (Malone & Lai, 1988) and other messaging toolkits allow users to develop customised support through semi-structured messages without enforcing par-

ticular styles of use. However, their operation is largely dependent on others using the same semi-structured systems (Lee & Malone, 1990; Cockburn & Thimbleby, 1992b). See section 2.2.4.

- GROUPKIT (Roseman & Greenberg, 1992) is a toolkit for real-time groupware, it allows the addition of generic group work facilities such as floor control and shared annotation (in the form of transparent overlays) through its “open protocols”. See section 2.4.2.
- In support for collaborative writing, *Milo* (Jones, 1992a; Jones & Cockburn, 1993) avoids modeling specific writing styles/roles in order to free co-authors from constraints.

4.5.4 Implementing minimal constraints

Much of the development effort in a system based on an explicit model or theory will be devoted to establishing an appropriate model or theoretical platform. An alternative model-based approach is to use a range of models/theories, and allow selection of appropriate models, either by explicit user-choice or by having the system adapt to the properties displayed by the users. Both of these approaches require substantial design effort to provide a sufficiently complete model or range of models.

Theories of communication and models of user behaviour promise to support successful systems in the future (Keeler & Denning, 1991), but until fully developed they are unlikely to afford adequate platforms for cooperative work. Designers aiming to produce working and workable groupware (rather than research systems) should therefore beware of the constraints imposed by embedding explicit theories and models into groupware.

4.6 Maximise external system integration

The first three groupware principles are primarily concerned with design and use of groupware *in isolation*. In contrast, maximising external integration requires designers to consider their system’s role within, and relationship to, the entire work environment. In this extended collaborative context, group members use competing systems to execute similar tasks, and a variety of tools (computer and non-computer based) are drawn on to support and assist collaboration. The principle also encourages development of toolkits and platforms for integrated collaboration environments.

The user-effort resulting from inadequate integration between systems was discussed in section 3.9. Although it was noted that integration is assisted by consistent user-interfaces, such consistency is hard to achieve in groupware because of the different hardware platforms and user-interface software used by colleagues.

In section 4.2.1 user-effort was attributed to transitions (changes from favoured styles and methods of working): figure 4.1 summarises these transitions. The external integration principle attends to user-effort resulting from inter-application (or inter-environment) transitions. Its primary aims are twofold:

Curative — to reduce the number and magnitude of transitions between tools employed in collaborative work.

Augmentative — to improve and integrate access to resources that serve communication and collaboration requirements. Information such as who's available for interaction, and how to contact people (telephone numbers, email and surface addresses, video connection dial-up sequences, and so on) can be pooled with access to interaction media.

The strategies for achieving external integration, detailed below, describe groupware approaches that reduce transitions in CSCW (also termed “seams”, and “barriers”), and guide designers on how to develop integrable groupware.

4.6.1 Video fusion

Video fusion techniques, best exemplified by the *TeamWorkStation* (Ishii & Miyake, 1991), reduce the seams in synchronous computer supported work. Ishii and Miyake view seams as “discontinuities from old working practices”, page-39. Seams in collaborative work therefore include: the lack of compatibility between personally favoured tools (forcing at least one collaborator to adopt “foreign” working methods), and the disruption to communication efficacy caused by the loss of important communicative cues such as gesture.

Fusion allows separate video images (perhaps a computer screen, and a human face) to be simultaneously displayed “on top of” each other, like layers of transparent acetate. In this way two (or more) otherwise incompatible applications, or work environments, can be used together: see section 2.5.2, and figure 2.11 for further details.

Video-fusion techniques are extremely appealing, but they have limitations, the most serious being the storage of the fused work-product. The result of the video connection is purely visual, consequently it is only sustainable for the duration of the video connection. When the video collaboration is terminated, if the result is to be stored for future reference, its constituent parts must be merged: the problems of system incompatibility must still be overcome.

4.6.2 Heterogeneous environments

Video-fusion techniques are primarily curative: they ease problems of inadequate integration between tools and techniques. In contrast, heterogeneous platforms augment collaboration by drawing together access to, and information about, collaboration resources: the aim is to provide synergy in collaboration support through “integrated portfolio of media” (Bair, 1989).

TELEFREEK (Cockburn & Greenberg, 1993), described in chapter 6, implements an extensible CSCW environment based on, but not limited to, standard networked computers. By merging computer supported information sources with various communication mechanisms and collaboration applications, TELEFREEK users are provided with a personalised platform for communication and collaboration.

4.6.3 Minimise dependence on structure and format

The first three principles (personal acceptance, minimise requirements, and minimise constraints) note that groupware’s dependence on specific information formats is problematical. It imposes additional user-effort, reduces flexibility, and enforces constraints. This strategy notes that the potential for system integration is also hindered by such dependence.

Groupware must follow relevant standards, for example, the X.400 email standard (CCITT, 1987) details legal information formats. Additional information-structures might legally be added within standards (for instance the semi-structured email templates described in section 3.7.1), but designers must consider the impact of such structures on colleagues who do not have access to the same structuring mechanisms. Systems built on existing communication media should do so monotonically—new facilities and enhanced features should not affect those already in use.

4.6.4 Implementation platforms

To minimise the impact of system incompatibilities that are due to differing hardware and interface platforms, designers must either:

- replicate some of the implementation—enabling the application to run on a variety of hardware;
- choose a suitable hardware-independent development platform—for example, the X Window system (Scheifler & Gettys, 1986);
- use an interface development application that can generate code for several Graphical User Interface environments—for example, the *Open InterfaceTM* (Neuron Data, 1992) system.

Cross-platform interface generators, such as *Open InterfaceTM*, are a recent development. They are likely to substantially ease the implementation of integrable groupware. Groupware development toolkits, such as GROUPKIT (Roseman & Greenberg, 1992) and OVAL¹¹ (Malone *et al.*, 1992), also promise to increase the integrability of systems, and ease development. GROUPKIT is described in section 2.4.2, and OVAL is examined in section 2.2.4.

4.6.5 Implementing external integration

External integration is primarily concerned with minimising the impact of adverse factors that are beyond the direct control of groupware designers. These factors include:

- integration with existing systems which *do* impose specific information format requirements;
- integration across hardware platforms.

The principle also explores the use of computing facilities to better support collaboration:

- merged access to differing communication mechanisms, resources, and requirements.

The strategies for achieving external integration make varying demands on designers.

¹¹OVAL is a re-implementation of the *Object Lens* (hopefully now bug-free). Personal experiences with *Object Lens* led to frustration and abandonment due to its buggy implementation.

Video-fusion and heterogeneous environments—require entire systems to be developed.

Minimising dependence on structure and format—re-iterates the arguments of the principles for minimised requirements and minimised constraints, and consequently make similar demands for developmental effort (sections 4.4.5 and 4.5.4).

Maximising hardware independence—while currently a formidable task, cross platform graphical user interface applications are becoming available, and toolkits for groupware development are easing integration across implementation platforms.

4.7 The principles' effect on design and implementation

The discussions on implementation issues (sections 4.3.5, 4.4.5, 4.5.4, and 4.6.5) stress that designing groupware under the principles' recommendations is a major undertaking. The question asked by designers, however, should not be “which approach is the easiest?”, but rather “which *potentially successful* approach is the easiest?” The only reprieve for groupware developers comes from the recent advent of toolkits that ease the implementation of groupware (see section 4.6.4) and that assist programming graphical user interfaces: for example, *Open InterfaceTM* (Neuron Data, 1992) and *X-DesignerTM* (IST, 1991).

Research has shown that although technology is capable of enhancing group work, being *capable* is insufficient; groupware must also (and in many ways, primarily) be acceptable. “Surface” system issues, such as interface quirks, or failure to provide adequate appeal to new users, are likely to prompt system rejection. Designers may argue that such superficial problems will be overcome when its group work enhancements are realised, but rejection at an early stage will thwart groupware's prospects for achieving them.

Designers using these principles must maintain the highest levels of motivation and professionalism. This need for high standards is not a direct consequence of the principles, rather, they are qualities required for the development of successful and acceptable groupware.

4.7.1 A note on evaluation

In chapter 3 the causes of groupware failure were divided into three levels: system-use, system-design, and system-evaluation. Much has been said about use and design, and recommendations for improving both have been discussed. Evaluation, however, has received little

attention.

Full evaluation of groupware is necessarily time-consuming, and its results are dependent on many social and organisational factors; it is also largely subjective (Grudin, 1988; Markus & Connolly, 1990). As a consequence of these and other problems, full evaluation of groupware is prohibitively costly and unreliable. An alternative approach is to engage in “participatory design” (Greenbaum, 1988; Clement & Gotlieb, 1987; Muller *et al.*, 1991; Kyng, 1991) (section 4.3.4) in which users are involved in the design process, and are therefore aware of the system and its facilities as they are developed.

The groupware principles can assist in participatory design by alerting those involved (designers, managers, and users) to the likely impact of particular groupware facilities.

Although participatory design improves the quality of applications, evaluation remains an important part of the iterative cycle of groupware improvement. Here too, the design principles can assist by providing a handle for the assessment of system properties: the level to which the system satisfies personal acceptance; the affects of its user requirements; the level to which it is perceived to constrain working practices; and how successfully it merges with the work environment.

4.8 Relating the principles to the vicious circle

In section 3.3 the primary factors affecting groupware’s success were shown to be related by a vicious circle (figure 4.4). Chapter 3 examined the key determinate in the vicious circle, user-effort, and it was noted that a “kick-start” is required during the initial stages of groupware-use to overcome the vicious circle’s chain of dependencies. This section relates each principle and its strategies back to the components of the vicious circle, and examines their impact.

Maximise personal acceptance

The personal acceptance principle particularly focuses on the direct and immediate benefits supplied by groupware. The aim is to make systems appealing to individuals (regardless of the number of other users), and consequently make groupware less dependent on a critical mass of users. *Catchpenny facilities* entice new users, and further encouragement is supplied when *champions* (system proponents) are employed to raise enthusiasm and awareness of

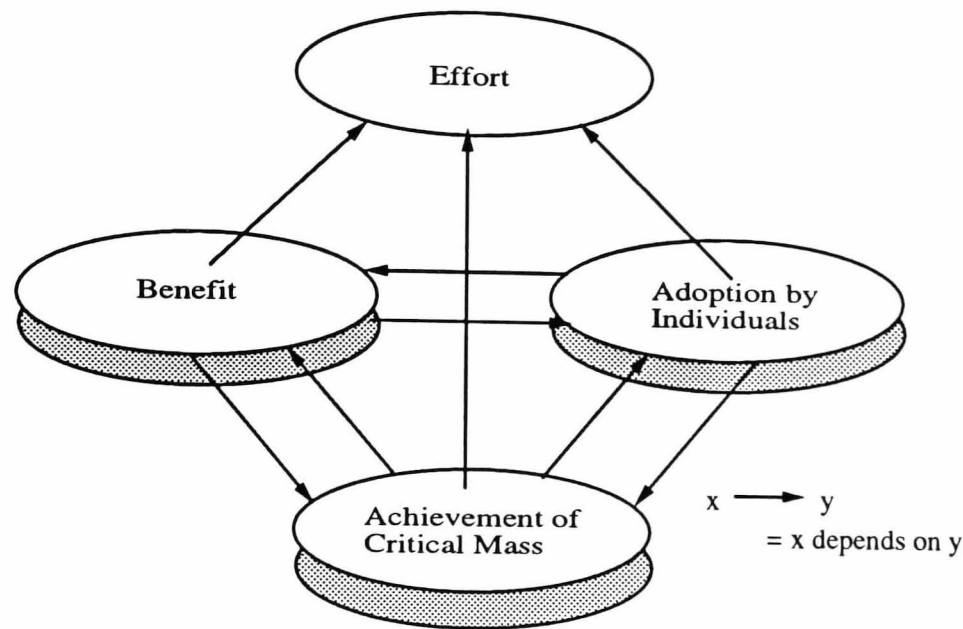


Figure 4.4: The “vicious circle” of dependencies in groupware adoption.

groupware’s potential.

The strategy for a *reflexive perspective of CSCW* (section 4.3.2) argues for merged personal-work and group-work support, and notes the reductions in user-effort that would arise.

Minimise requirements

The minimise requirements principle is directly concerned with user-effort: the primary factor that reinforces the vicious circle’s dependencies. The principle particularly focuses on avoiding the way that groupware imposes user-effort on users in order to provide its intended support.

In weakening the vicious circle’s relationship between effort and benefit, the strategy for *avoiding dependence on user-effort* may temper the ambitions of groupware (certain facilities may be impossible without such dependence). However, avoiding dependence is motivated by the pragmatics of collaborative work: some user’s will be unwilling to carry out additional work, and others may be unable to.

The *use information that is available ‘for free’* strategy provides a kick-start to user-benefits, independent of a critical mass of other users and their additional work. *Equal opportunity* reduces the level to which groupware impose on its users. It increases the freedom of choice in who carries out the additional work required for particular facilities.

Minimise constraints

Like minimise requirements, minimising constraints also reduces user-effort in groupware. Rather than the explicit forms of user-effort addressed by the minimise requirements principle, minimising constraints is primarily concerned with the user-effort that is derived from inflexible styles of use. Personal acceptance of groupware is also assisted by minimising constraints, because the flexibility it argues for makes systems more malleable to *personal* requirements.

Maximise external system integration

The user-effort derived from transitions between systems is reduced by the maximise external integration principle. Heterogeneous collaboration environments not only reduce user-effort in switching between work support tools, they also increase the benefits available to users by merging facilities in a way impossible without computer facilities. Video-fusion techniques increase the richness of communication, consequently reducing the effort in conveying information.

4.9 Summary

In this chapter, four groupware design principles have been proposed together with practical development strategies for achieving their aims. Existing groupware systems and relevant CSCW literature have been used to exemplify the principles' implementation and intentions.

The principles offer practical advice for groupware designers, and consequently address the causes of groupware failure in *system-design* (section 3.2). The direction of the principles' guidance is derived from an identification of the groupware properties that are responsible for inadequacies during *system-use*. The principles advise groupware developers on how to overcome these problems.

The following chapters describe two groupware systems, *Mona* and *TELEFREEK*, which explicitly demonstrate these principles.¹² These systems' vigorous exposition of the design principles results in the use of radically different support mechanisms when compared to existing groupware that offer similar group work facilities.

¹²A third system, *Milo* (Jones, 1992b; Jones, 1992a; Jones & Cockburn, 1993), has been developed under the principles' guidance, but is not described here.

Chapter 5

Mona: The principles and conversational email

5.1 Introduction

The development of *Mona*, a conversation based platform for email communication, was coupled with that of the groupware design principles. The discoveries, observations, and insights stimulated by *Mona*'s development provided fuel for the principles' increasing maturity: from initial prototypes exemplifying the reflexive perspective of CSCW, to the incorporation of “free” email enhancements motivated by the discernment of user-effort's debilitating affect on groupware.

In section 2.2 many enhanced email systems were examined, and in section 3.7 their common dependence on explicit user guidance was identified as a major cause of failure. In this chapter, *Mona* exemplifies an alternative, guidance-free, approach to the provision of email management enhancements.

The prototype systems, and preliminary studies that influenced *Mona*'s development are discussed in section 5.2. Section 5.3 provides a detailed description of *Mona*. *Mona*'s adherence to the four groupware design principles is examined in section 5.4.

Future work on *Mona* (and further work stimulated by it) is described in chapter 7, and a user's guide is available in appendix B.

5.2 Prototypes and investigations

Prior to examining *Mona*, this section describes its initial prototypes and preliminary investigations. These exercises substantially influenced the facilities supported by *Mona*, the mechanisms it employs, and the guidance provided by the groupware design principles.

5.2.1 HyperCommitments

The first prototype, *HyperCommitments*, was primarily motivated by the paper “Reflexive CSCW: Supporting long-term personal work” (Thimbleby *et al.*, 1990). In applying a reflexive perspective (see section 4.3.2) to the group-like work-coordination requirements of an individual, *HyperCommitments* (Cockburn, 1991a) experiments with the notion that all files represent a commitment to some course of action.

Reflexive CSCW views the single user as a group of one, working around numerous tasks (past, present, and future), in different places, and possibly on several machines—perhaps one at the office, one at home, a lap-top for commuting, and an assistant’s. The individual’s burden in coordinating work is further complicated by the potentially overwhelming size of personal computer data stores. Organising, maintaining, and retrieving information from these stores is a major overhead, and most frequent computer users will, at some time, stumble across a long forgotten file that represents an outstanding work item.

In improving management of personal information, research has concentrated on enriched relationships between items, and easing navigation around the information space (Furnas, 1985; Remde *et al.*, 1988; Sarkar & Brown, 1992). Hypertext (Conklin, 1988; Nielson, 1990a) enriches connections between information and can therefore enhance access; unfortunately, the connections are frequently too rich and consequently users become “lost in hyperspace”.

Temporal aspects of work can also be used to assist information management. In most operating systems, the use of time for file access is limited to time-ordered views of modification dates within individual directories (for example, `ls -t` in Unix). *Memoirs* (Lansdale *et al.*, 1989; Lansdale & Edmonds, 1992) and the *Memory Extender* (Jones, 1986) make explicit use of time and notions such as forgetting, links that fade over time, and so on. These systems use time as an aid to recall and information retrieval.

HyperCommitments was built to investigate the use of temporal information as an *active* assistant in managing information spaces. Rather than drawing on recollection of past events

to aid recall of files (in the way that *Memoirs* and *Memory Extender* do), *HyperCommitments* examines information recall based on future events. By requesting users to record the commitments associated with files, *HyperCommitments* can provide timely reminders, and allows users to access files through queries based on *future* events (for example, “all files that I must deal with next week”).¹

The implementation

HyperCommitments, implemented in HyperCard (Apple Computer, 1987; Coulouris & Thimbleby, 1992), supplements information management through time-ordered views of past activities and activity prompts in the form of commitment reminders. Acceptance of the system’s advice is at the user’s discretion: commitment dates are flexible and may be ignored.

Files, and their associated commitments, are accessed through two management levels:

Folder-management-level—a single HyperCard card that displays the names of all file-store folders that contain recorded commitments (folders accessible through *HyperCommitments* are termed *HyperCommitments-aware*). Mouse-clicking on a folder-name reveals the files it contains, and clicking on a file-name opens the file within its parent application.

File-management-level—every file within a *HyperCommitments-aware* folder is represented by a card in the file-management-level (an example is shown in figure 5.1). The cards show information about the file, including: its name, date of last access, parent application, and any additional notes that the user supplies. Commitments are attached to cards in the form of buttons then display their *activation date* (the day the system will start providing reminders). When clicked, these buttons produce a textual description of the commitment in a pop-up field.

HyperCommitments is automatically invoked at system start-up. All recorded commitments are examined, and “active” ones are notified. Each active commitment must be managed in one of four ways:

¹Recently, commercially available personal information management systems have used date-stamps of the type provided by *HyperCommitments* to enable similar reminder facilities. Most notable among these systems is *Agenda* (Kaplan *et al.*, 1990).

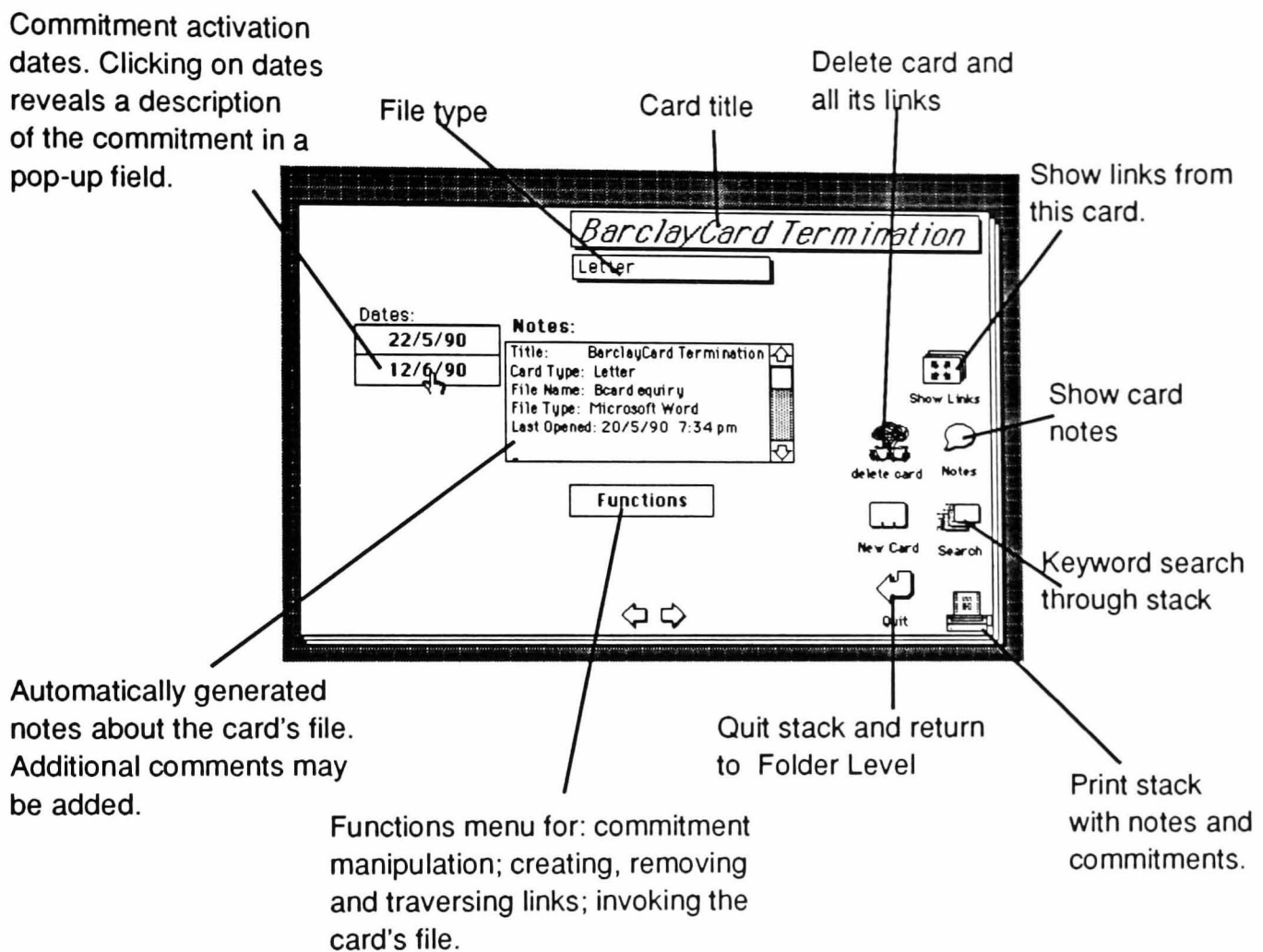


Figure 5.1: *HyperCommitments* file management level.

1. Open the underlying file within its parent application—doing so removes the commitment (the assumption being that the committed actions are being executed). Although automatic removal will sometimes be inappropriate, the design motivation is that automatic removal (though sometimes incorrect) will be preferable to requiring explicit commitment removal.
2. Change the commitment date.
3. Delete the date and commitment.
4. Ignore it—leaving the commitment to be re-notified at the next system invocation.

HyperCommitments also supports hypertext navigation of the information space through bi-directional links between cards in the file-management-level. A key-word search facility assists link creation.

Informal evaluation

Comments on *HyperCommitments*' interface and functionality were gathered from seven computer scientists, each having a single thirty-minute session of demonstration and use with *HyperCommitments*. Although evaluating *HyperCommitments*' temporal-basis for information management within a *single* session could only yield subjective results, there were several reasons for doing so. HyperCards's limited file manipulation operations, and its separation from the underlying Mac operating system, resulted in *HyperCommitments* enforcing constrained and restricted file-management schemes.² Consequently, it was unreasonable to expect people to use *HyperCommitments* as the main access to their file-stores.

HyperCommitments' "evaluation" was intended to be more of an exercise in participatory design, than a formal evaluation. It focused on interface issues and user-opinions of the temporal-basis for file-management. The responses were sufficiently positive to motivate further development in an environment that did not impose the restrictions of HyperCard.

5.2.2 Personal and communicated reminders in *mind*

The Unix-based prototype, *mind*, continued *HyperCommitments*' investigation into personal commitments, their relationship with files, and their use in assisting information management. The move from Mac to Unix environments was due to two primary factors:

- The local test-bed of computer users (the Computer Science Department at Stirling University) are almost unanimously Unix-familiar, and few regularly use Macs.
- Email in the Unix environment offered a suitable communication platform for experimenting with "reflexive" facilities (section 4.3.2), in which the same mechanisms are used for personal and communicated commitment.

The system

Implemented in C, *mind* is a Unix-based utility that allows textual reminders (optionally attached to files) to be posted to oneself or others.³ Reminders in *mind* have associated activation

²Although an appealing prototyping tool, HyperCard's lack of formality and bugs (Thimbleby *et al.*, 1992) hinder its use beyond interface mock-ups.

³See Appendix A for the *mind* manual pages.

dates that cause the reminder to be notified at (and after) the date. Notification is made in one of three ways (depending on the options used when posting the reminder):

1. Active assistance in navigating to the files that have associated commitments—provided by adding the characters `@!` to the file-name, and to all parent directories (propagating to `$HOME`). Thus, the path to file
`~/thesis/chap5/mona_figure`
would become
`~/@!thesis/@!chap5/@!mona_figure`
2. Passive assistance in navigating to commitment-files—provided by explicitly stating the path to the file. The text message that is associated with the reminder is also displayed (if one exists).
3. The reminder message is displayed when commitments are not explicitly associated with files.

Other `mind` facilities include canceling and changing reminders, reviewing upcoming commitments, and removing the navigational aid provided by `@!` renaming.

Informal evaluation

`Mnd`'s primary use was in posting personal reminders, without file connections: essentially it was used as an active *personal* diary. Although reminders were posted to others, users expressed surprise and dissatisfaction with its separation from email. Several noted that reminder functionality would be more appropriate as a supplement to email.

`Mnd`'s file attachment facilities were almost exclusively ignored; if a file attachment was required, the file's path was added to the text of their reminder.

Comments suggested that users found the temporal-basis for the management of (reflexive) communications useful. They liked `mind`'s ability to “hide” information until it became relevant, but disliked its separation from email when sending reminders to others.

The observation that people could benefit from temporal information management (to hide information until it becomes relevant) concurs with (Malone, 1983) and (Mackay, 1988). Malone and Mackay found that information is stored in temporary locations (such as the desk

surface, or email inbox) not because it is urgently needed, but rather because it will be needed in the future, and its visible presence serves as a reminder of pending actions.⁴ This indicates that information overload is not only caused by too much information too quickly; it is also due to too much at the wrong time. Temporal forms of information management, such as those provided by *mind* and extended in *Mona*, address this problem.

5.2.3 Pre-*Mona* studies

During *Mona*'s initial development, shell scripts that trapped all incoming and out-going email messages were installed in several users' accounts. Several weeks email was captured, and users were interviewed to assess the relationships they perceived between messages; they also completed a questionnaire about their email use and satisfaction. The results of this study, reported in Branskat (1991), suggested that the majority of contextual relationships between email messages could be extracted from information available in standard email message headers.⁵ In addition, it was found that message context information "explicitly" supplied by the user was unreliable—the "Subject:" field frequently bore no relationship to the actual subject matter.⁶

This study provided evidence for the potential of guidance-free email enhancements (see section 4.4.2). Although it is commonly accepted that "an underlying requirement within cooperative working systems is the need for some structuring facility" (Rodden & Sommerville, 1991), page 161; this study suggests that, for some enhancements, the structured information need not be explicitly provided by users.

5.3 *Mona*

Mona is an extended email application that incorporates several forms of enhanced information management. The enhancements include:

⁴A computerised desktop "Pile" metaphor is demonstrated in (Mander *et al.*, 1992).

⁵The email study was carried out by Sonja Branskat, a graduate student at the University of Calgary, Canada (Branskat, 1991); Andrew Cockburn supervised the project.

⁶This was due to automatic copying of "Subject:" fields when using email "Reply" facilities. "Reply", however, is used as a method for automatically filling in the "To:" field, and not necessarily for conversational replies.

- **Conversation-based relations between email messages**—similar to many groupware systems, including: *gIBIS* (Conklin & Begeman, 1988b; Conklin & Begeman, 1988a); *The Coordinator* (Winograd, 1987; Winograd & Flores, 1986; Flores *et al.*, 1988); *Strudel* (Shepherd *et al.*, 1990); WHAT (Hashim, 1991); and *Sibyl* (Lee, 1990). For a review of these systems, see sections 2.2.2 and 2.2.3.
- **Commitment reminders**—temporal information management facilities that extend the work of *HyperCommitments* and *mnd*.

The aims, motivation, and techniques of conversation-based email systems were described in section 2.2.2, and their dependence on user supplied guidance was reviewed in section 3.7.

The groupware design principles evolved and were clarified in conjunction with *Mona's* development. This process began, prior to *Mona's* implementation, when reviewing related literature on groupware for conversation-based email. The common dependence on user-guidance was identified as a potential cause of failure. Furthermore, systems that constrained users to particular styles of use (such as *The Coordinator*) not only failed, they were hated. These groupware problems are reviewed in chapter 3.

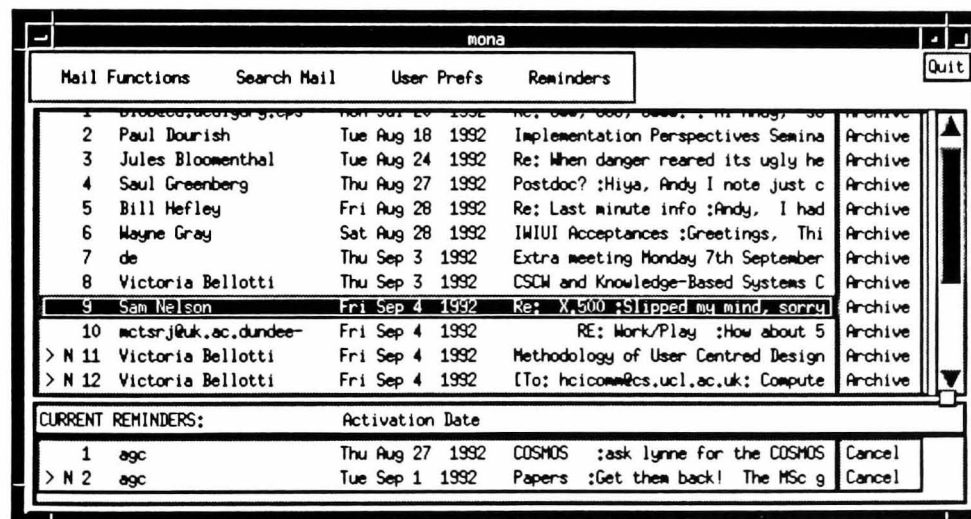


Figure 5.2: *Mona's* inbox.

5.3.1 *Mona's* aims

The research motivation for *Mona's* development came from a desire to investigate alternative methods for improving email management, and from the potential enhancements to collaborative work that email offers. The emphasis in this research is on methods and techniques. It

addresses issues that include the alternatives to guidance-dependence, and the appropriateness of the reflexive paradigm for groupware.

In groupware research it is exceptionally difficult to substantiate claims about the usefulness and work enhancing value of facilities without providing those facilities in a fully operational system. Prototyping is largely insufficient (Borenstein & Thyberg, 1991) because people can only be expected to use complete systems, with polished and usable interfaces, in their everyday work.

The experience and feed-back gained from *HyperCommitments* and *mind* were sufficiently positive to encourage the implementation of a complete groupware application: a substantial work commitment. The system (that became *Mona*) would experiment with guidance-free facilities and reflexive communication facilities within a full and complete email environment built on the X-Windows system (Scheifler & Gettys, 1986). To better demonstrate the guidance-free schemes, a design intention was to limit the system's functionality: it would totally avoid the semi-structured (and guidance-dependent) schemes implemented in related groupware.

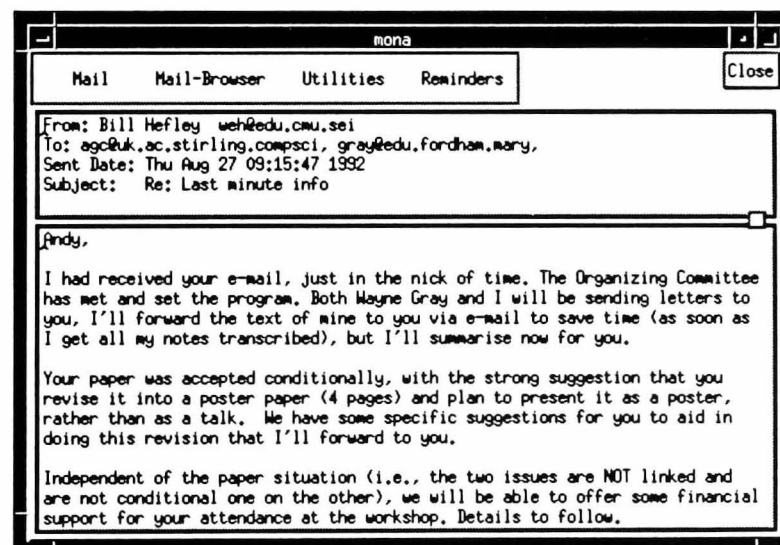


Figure 5.3: A standard mail item in *Mona*.

5.3.2 System overview

Incoming email messages, and those already contained in the user's mailbox are accessed through *Mona*'s main window, shown in figure 5.2. Also shown (at the bottom of this window) are the currently active reminder messages. When the mouse-cursor passes over a one-line

message summary, the line is highlighted (message number 9 in figure 5.2), and clicking the mouse button causes the selected message to be displayed in a separate window (figure 5.3). The windows that display reminder messages contain two additional fields: an activation date and a calendar (see figure 5.4). All email messages, in-coming and out-going, are automatically archived in a database that can be searched using *Mona's* “Search template” (figure 5.5).

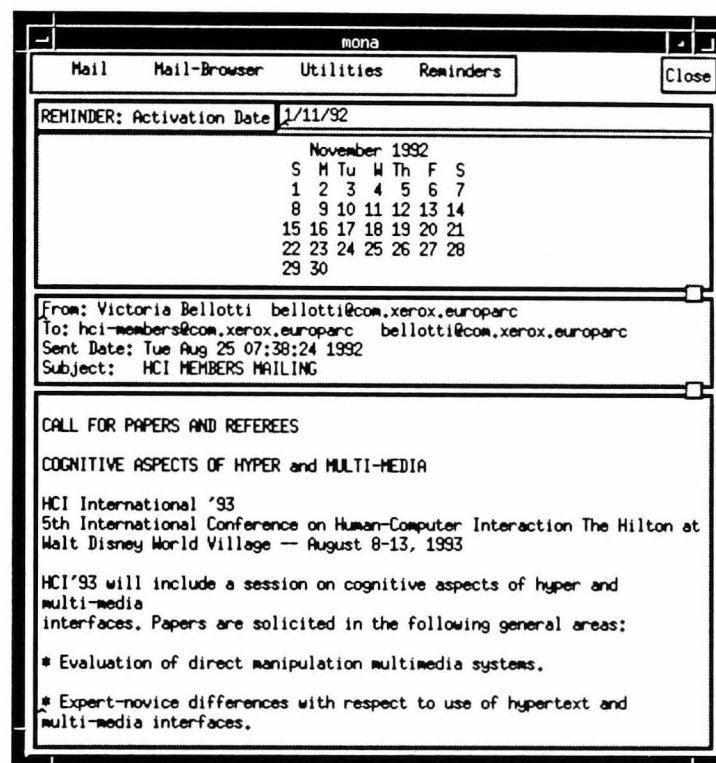


Figure 5.4: A reminder mail item.

All messages sent or received by *Mona* are automatically run through a set of conversational heuristics. These heuristics attempt to establish two pairs of links between messages. The first pair of links form a linear temporal-ordering of messages from a particular source. The second pair provides an interpretation of conversational context in email interaction. The conversational “web” arising from these links can be browsed, modified, and customised within *Mona's* conversational web window, shown in figure 5.6.

Mona includes all standard email facilities, such as “Reply”, “Send File”, “Delete”, and so on. This chapter is concerned with *Mona's* contribution to CSCW research, and consequently, its low-level facilities will not be described here. A complete user-guide to *Mona* is available in Appendix B (also available as a technical report (Cockburn, 1992)).

In the following sections *Mona's* techniques for providing guidance-free conversational context are detailed. The development of these facilities led directly to the “minimise require-

ments” groupware design principle. Regardless of its value and success, *Mona*’s approach in escaping guidance-dependence is vehement. Although it does provide guidance-free enhancements, it *intentionally* fails to support guidance-dependent schemes for those willing to work on behalf of others. *Mona*’s limitations of this nature are discussed in section 7.5.

In section 5.3.3 *Mona*’s techniques for achieving guidance-free facilities are described. The many other ways in which *Mona* follows the groupware design principles are detailed in section 5.4. *Mona*’s facilities particularly focus on supplying immediate and personal value to assist the breakdown of dependencies in groupware’s vicious circle.

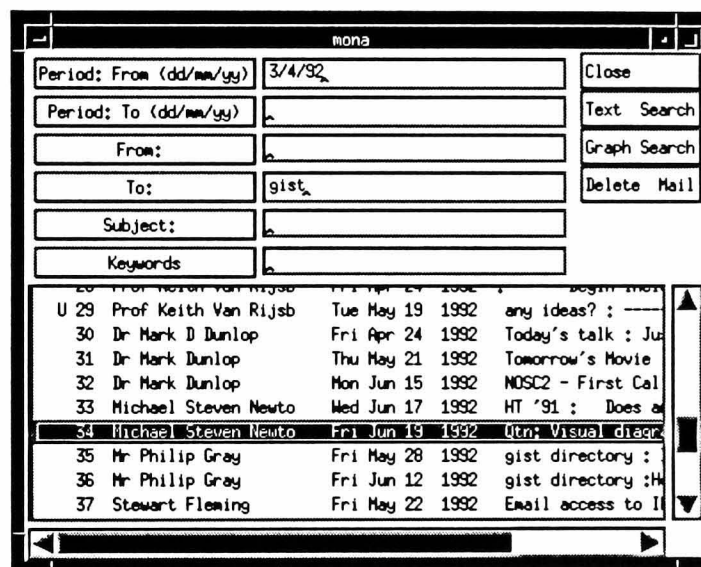


Figure 5.5: *Mona*’s search template.

5.3.3 Conversational context “for free”

To infer conversational context *Mona* uses RFC822 header information (Crocker, 1982) that is independent of user actions and guaranteed to be present in every message. Information about each new message is therefore limited to the names/addresses of the sender and recipient(s), the time and date at which the message was sent, and approximately when it arrived (available from the first **Received:** field). By combining this information with knowledge of previous incoming and outgoing mail items, contained in the mail archive, *Mona* infers the probable context relating messages, forming a “web” of conversational relationships (figure 5.6).

Mona attempts to establish four link types (two pairs) with each incoming or outgoing message:

- previous message by the same user (or source) — **previous by same**;

- next message by the same user (or source) — **next by same**;
- the inferred email cause(s) of a message — **cause**;
- the inferred email response(s) to a message — **response**.

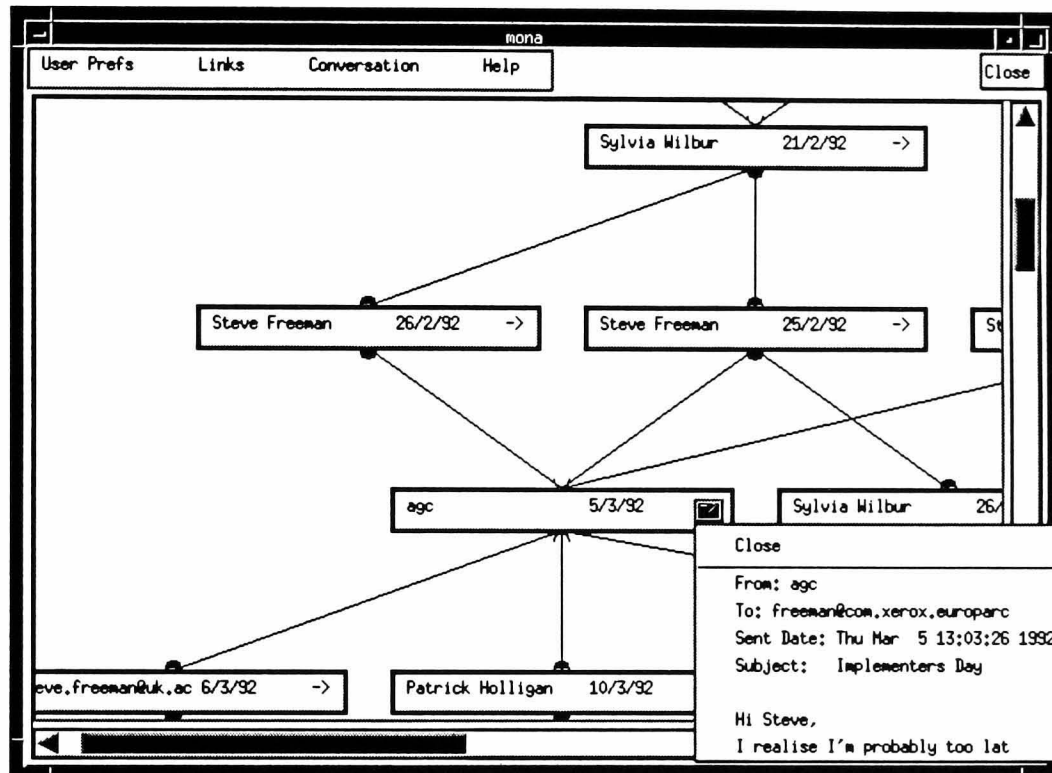


Figure 5.6: An automatically inferred conversational web.

Previous and Next links

The **previous** and **next** message links form a total ordering of communications originating from a single source (author or mailing list, for example). A previous message link is attached to new mail whenever the archive contains a message from the same source that was sent at an earlier time (using the sending rather than arrival time eases some of the problems of delayed messages). Using a, b, \dots as message variables and u, \dots as users, the rules are defined as follows:

$a_{\text{previous by same}} = b \text{ when}$ $\text{prev_part}(a, b) \wedge$ $\forall c : \text{prev_part}(a, c) \implies c_{\text{send time}} \leq b_{\text{send time}}$ <p>where</p> $\text{prev_part}(a, b) = (a_{\text{sender}} = b_{\text{sender}} \wedge$ $a_{\text{send time}} > b_{\text{send time}})$

Table 5.1: Previous by same rule.

Previous and next links are established in pairs, thus whenever a previous link is made a corresponding next link is established:

$a_{\text{next by same}} = b \text{ when } b_{\text{previous by same}} = a$

Table 5.2: Next by same rule.

Once created, previous and next by same links are only modified by message deletion. These links are particularly useful when modifying conversation structure; their use in this context is described below (section 5.3.4).

Cause and Response Links

Cause and **Response** links provide the conversational relationship between messages. They can be browsed with a graphical display of the conversation web (figure 5.6). While the naming of these links may over-stress the relationship between messages, it is intended that users will develop personal interpretations of link meaning without close attention to the specific terms used by *Mona*. When new messages are processed by *Mona* (both incoming and out-going messages) the cause(s) are determined as follows:

when receiving a message a from user u to a set of addresses U , the system will infer that for each receiver $u' \in U$, the cause of a is the most recently preceding message from u' that includes u in its list of recipients.

Thus, a cause is defined:

$a_{\text{cause}} = b \text{ when}$ $\text{conversation_part}(a, b) \wedge$ $\forall c : \text{conversation_part}(a, c) \implies c_{\text{receive time}} \leq b_{\text{receive time}}$ <p>where</p> $\text{conversation_part}(a, b) =$ $(a_{\text{sender}} \in b_{\text{receiver}} \wedge$ $b_{\text{sender}} \in a_{\text{receiver}} \wedge$ $b_{\text{receive time}} < a_{\text{receive time}})$

Table 5.3: Cause heuristic.

Cause and **response** links utilise message arrive-time (**receive time**) allowing quasi-causal effect to be based on events observable by the receiver (Lamport, 1978); whereas in establishing **previous** and **next** links the time of message sending provides an ordering based on events observable by the sender. As before, **cause** and **response** links are established in symmetric pairs. Each **response** is therefore established as a consequence of a **cause**:

$a_{\text{response}} = b \text{ when } b_{\text{cause}} = a$
--

Table 5.4: Response heuristic.

Under these rules, the number of causes that can be attached to a particular message is bounded by the number of individual recipients addressed in the message header. Should *Mona* fail to find any **cause**, a check is made to see if the message is addressed to a mailing list. Email addressing conventions make the **cause** heuristic inappropriate for inferring conversational context from mailing lists: the condition $a_{\text{sender}} \in b_{\text{receiver}}$ cannot be satisfied when the receiver is a mailing list (although messages can be sent **To:** a mailing list name, they are always **From:** a single person). Therefore, to provide some information relating to the context of mailing list messages, the previous n (where n is user-defined) messages addressed to the mailing list are attached as **causes**. To receive this separate mailing list inference of conversational context, *Mona* can be informed of each list the user belongs to.

5.3.4 Using and modifying conversational context

Through the use of heuristics, *Mona* provides what is:

At worst a zero cost, free, guide to conversational context. Even this worst-case is discretionary; the user can ignore it.

At best an accurate reflection of the user's interpretation of conversational context.

Realistically a system that provides predictable email management that users will learn to use effectively, despite its limitations in conversational classification.

Due to the differences between individuals' interpretation of context (Romiszowski & Jost, 1990), it may be argued that the best any system can provide is no more than a guide to conversation structure. Johansen (1988) emphasises the danger of imposing a single interpretation of context on users, "Structuring people's conversations is a risky business. It can be perceived as intrusive or worse." *Mona* therefore allows modification of the inferred conversation structure, and provides assistance in doing so. The **previous by same** and **next by same** links ease browsing and selecting communications with an individual, and a search template supports selective retrieval of messages satisfying a variety of combined properties (figure 5.5).

The conversation "web"

Each node in the graphical conversation web (figure 5.6) represents a single email item, identifiable through a user-customisable combination of sender-name, sent-date, and message-subject. A pop-up mail summary is available through a preview key (represented by the $->$ symbol), and a separate window displaying the complete message may be requested through menu options associated with each node. Additional guidance in navigating through the conversation web is provided by icons above and below each node showing whether further **cause** and **response** links remain unexplored: an open (unfilled) arc represents unexplored links, closed arcs show exhausted paths.

The combination of inferred conversational context and flexible modification of conversation structure enables the satisfaction of the design principles: additional work is not required, but if carried out it (reflexively) provides personal benefit. In highly collaborative work, however, the distinction between personal and group benefit can become blurred (Cockburn &

Thimbleby, 1991). *Mona* can be used to support shared views of conversation progression provided the collaborators have access to a common Unix directory—the path of the default mail archive directory may be changed to that of a shared directory using one of *Mona*'s preference settings. Supporting a shared view of conversations in this manner may be valuable in ensuring that co-workers have a mutual understanding of their relevant commitments and responsibilities.

5.4 *Mona* and the principles

Mona's heuristics for message relationship allow it to escape the guidance-dependence that has contributed to groupware's lack of success. Its ability to provide “something for nothing” motivated the “minimise requirements” groupware design principle. In the following sections *Mona*'s adherence to the remaining three principles is discussed.

5.4.1 Maximising personal acceptance

The vicious circle in groupware adoption (section 3.3) emphasises the importance of making groupware attractive to individuals. The most obvious way of achieving this is to provide a highly polished and attractive user interface (Bloomer & Ingram, 1992). *Mona*'s WIMP (Shneiderman, 1987) graphical interface is certainly more appealing and intuitive than the basic command line interface of Unix `mail`. Contrast figures 5.2 and 5.3 of *Mona* with figure 5.7 which shows the user interface to Unix `mail`.

Catchpenny facilities

Improving interfaces does not completely satisfy the “maximise personal acceptance” principle. Although newcomers to a particular style of support will gravitate towards systems providing the best interface, people who are already committed to a particular tool are likely to continue using it until notably disadvantaged (Goodman & Abel, 1987; Cockburn & Thimbleby, 1992a). Groupware must therefore offer additional appeal to encourage the breakdown of this inertia-driven use of current methods and systems.

To achieve this *Mona* provides “Catchpenny features” (section 4.3.1). These include one-mouse-click laser printed copies of letters, spell-checkers, and automatic signature messages. Although these facilities are useful, their design motivation is to provide instant and per-

```

xterm80c
Received: from 000040010500/ by uk.ac.stir.cs with NIFTP id AA04299;
  11 Nov 92 00:21:05 GMT (Wed)
Received: from alpha.Xerox.COM by sun3.nsfnet-relay.ac.uk with Internet SMTP
  id <sg.00328-0@sun3.nsfnet-relay.ac.uk>;
  Wed, 11 Nov 1992 00:18:52 +0000
Received: from mega.europarc.xerox.com ([13.1.252.106]) by alpha.xerox.com
  with SMTP id <12322>; Tue, 10 Nov 1992 16:17:48 PST
Received: from alpha.Xerox.COM by mega.europarc.xerox.com
  with SMTP (5.65c/IDA-1.2.9) id AA15931; Wed, 11 Nov 1992 00:10:32 GMT
Received: from bells.cs.ucl.ac.uk ([128.16.5.31]) by alpha.xerox.com with SMTP
  id <12376>; Tue, 10 Nov 1992 16:11:53 PST
Received: from sol.cs.ucl.ac.uk by bells.cs.ucl.ac.uk with local SMTP
  id <g.26175-0@bells.cs.ucl.ac.uk>; Wed, 11 Nov 1992 00:10:44 +0000
From: P.Dourish@uk.ac.ucl.cs
To: research@uk.ac.ucl.cs
Organisation: Dept of Computer Science, University College London
Phone: +44 223 341512
X-Mailer: PP/Ream v4.16c (The Choice for a New Generation)
X-Subliminal-Message: Send me all your thesis ideas
Mail> h
> 1 P.Dourish@uk.ac.ucl.cs Wed Nov 11 00:21 154/5537 Call for papers: Collab
or
2 P.J.Holligan@uk.ac.lut Thu Oct 22 11:23 38/1049 Re: Expenses: CSCW-SIG
3 pbl@uk.ac.stir.cs Tue Feb 23 19:23 158/6593 Excerpt from the Requirem
4 salisbury@edu.washington.cs Wed Jan 13 19:36 39/1428
5 saul@ca.ucalgary.cpcc Tue Mar 30 03:44 47/1945 Re: paper...
6 spm@uk.ac.stir.cs Wed Feb 24 09:19 33/1060 stuff
7 sylvia@uk.ac.qmw.dcs Tue Apr 7 16:09 47/2077 Re: Computer Support for
8 suchitra@ca.ucalgary.cpcc Sun Feb 9 04:09 49/2222 Hi there
Mail>

```

Figure 5.7: User interface to the Unix mail command.

sonal user appeal. *Mona's* conversational facilities are powerful and potentially valuable in cooperative work, but their immediate and direct benefit to users is perhaps unobvious. Its catchpenny facilities, however, are the bells and whistles to which new users can immediately relate.

Reflexive CSCW

Another strategy for maximised personal acceptance, described in section 4.3, is the “Reflexive Perspective of CSCW”. *HyperCommitments* and *mnd* experimented with work commitments, both personal and collaborative, and observations made from these systems were used to incorporate a (reflexive) reminder facility in *Mona*.

Mona's reminders are a special type of email message: an activation date is contained in the message header. When displayed (by *Mona*) the message's window shows the calendar-month of the activation date (figure 5.4). Any email message can be converted into a reminder by selecting a menu option, and subsequently stating the activation date (see Appendix B, the *Mona* user guide). In this way email can be easily “hidden” until the appropriate time, thus providing temporally-based information management. A search template similar to figure 5.5 allows reminders satisfying a set of properties to be displayed: for instance, all reminders about project X that become active in the next month.

The reminder facilities described so far are essentially reflexive: the receiver manipulates

activation dates for personal benefit. In communicating reminders, email senders can delay the notification of their message's arrival by assigning an activation date *before* they send it. On arrival, reminder messages are autonomously extracted from the inbox, and “sleep” until they become active: when received by mail systems other than *Mona* reminder messages are displayed as normal.

Mona's reminders serve several purposes, including the following:

- Reminders of events pending action (commitments)—notification is given when the activation date becomes current.
- Assistants in reducing information overload—reminders allow people to hide information until it becomes relevant.

Much of the burden of information overload is caused by the arrival of information at the wrong time. Those responsible for temporally-inappropriate messages may be aware that they are sending information at non-optimal times, but they may do so to “get it off their hands” before forgetting. Without information management assistants (such as *Mona*), the receivers of such messages must decide whether to leave it in their inbox (as a visual, but cluttering, reminder of upcoming events), or to remove it and risk forgetting at the appropriate time.⁷

- A general “hidden” information store—reminders allow notes, messages, files, etc, to be temporarily hidden. Useful notes such as command sequences or currently unclassifiable information (Malone, 1983) are held in reminders, allowing easy access without cluttering the work environment.

5.4.2 Minimising requirements

Mona's fundamental aim was to minimise dependence of user actions, and to experiment with guidance-free schemes for conversational context in email. The groupware design principle for minimised requirements largely evolved out of development experiences with *Mona*. Therefore,

⁷It may seem profitable to further extend temporal facilities in email: for example, to automatically delete reminders of past events. Such facilities are potentially counter-productive—for instance, despite missing a particular seminar, the fact that it occurred may be important.

this chapter's discussion of *Mona*'s motivation, ambitions, and techniques is essentially a description of "minimised requirements".

Mona intentionally demonstrates the "minimise requirements" principle to excess: in avoiding *all* guidance-dependence, it fails to support those people who wish to work for the benefit of others. A comparison between the costs and benefits available through guidance-dependent, guidance-"free" (like *Mona*), and combined dependent/free approaches is shown in figures 4.2a, b, and c. From these figures it is clear that *Mona* does not provide optimal support: optimal support, however, was not *Mona*'s primary objective. A discussion of future work and potential improvements to *Mona*'s functionality is given in chapter 7.

Equal opportunity

The "equal opportunity" strategy for minimised requirements increases the freedom of choice in who provides or executes actions guidance (see section 4.4.3). Rather than demanding senders to work for the receivers' benefit, equal opportunity would also allow senders to work for their own benefit, and the receiver to work for the sender's benefit. To some extent *Mona* achieves this within local-sites by allowing users to share common mail archive directories: the name of the default archive can be changed through one of *Mona*'s preference settings. Shared access to archives could allow any user (with the correct permissions) to alter, personalise, and customise the conversational interpretation that is initially established by *Mona*.

5.4.3 Minimising constraints

Discussing an email system's observance to a "minimal constraints" principle may seem inappropriate—surely email's constraints (as a communication medium) are unavoidable, and why should email-based systems impose additional constraints? This question seems more pertinent when considering that it is (arguably) email's lack of constraints that is responsible for its success. Email fulfilled a communication requirement *without* imposing requirements or constraints on styles of use, and remains the only mass groupware system in use.

Enhanced email groupware projects have, however, removed this freedom from requirements and constraints. They have been dependent on user-explicit guidance and particular (constraining) styles of use (see section 2.2 and 3.7).

In contrast to related groupware, *Mona* ensures that users are unconstrained in their

manipulation of conversational webs:

- Every user has a personal copy of the mail archive which, although initially in a state determined by the conversational heuristics, can be modified to reflect personal conversation views.
- *Mona's* support for shared views (see section 5.4.2), perhaps ensuring a common conversation interpretation, imposes no additional constraints above those of Unix directory permissions.
- In leaving the styles of collaborative work unconstrained, *Mona* allows social protocols to govern the appropriate schemes for modifying shared conversational representations.

5.4.4 Enabling external integration

Mona's adherence to the first three principles increases its level of integrability with similar systems:

1. Under the personal acceptance principle, reflexive CSCW allowed personal and group commitments to be integrated into a single reminders facility.
2. In minimising requirements, *Mona's* conversational heuristics enable enhanced facilities independent of the sender's or receiver's email systems.
3. *Mona* allows collaborators to select their own work governing protocols, thus reducing the likelihood of clashes between the users' preferred working styles and those enforced by systems.

Therefore, by following the first three principles, *Mona's* integrability has, almost coincidentally, been improved.

Monotonic development is an important issue for pursuing maximised integration: groupware that is built on existing mechanisms, with their own established patterns of use, should not hinder users who are committed to the existing facilities. *Mona* ensures that new users are not constrained to different working styles from those already employed. *Mona's* enhanced facilities are constantly available, but there is no requirement for their use. Additionally, the information *Mona* uses to store reminder dates and inferred conversational relationships

is held in message headers in a format conforming to RFC822 recommendations (Crocker, 1982), and also conforming to X.400 (CCITT, 1987). *Mona* therefore cannot adversely affect those using other email systems.

Mona is implemented in C for X-Windows running under Unix. This limitation could lead to complications for those using more than one hardware platform for accessing email—perhaps an X-Windows workstation at the office and a modem/laptop connection while travelling. For this reason, an additional program is available with *Mona*; it restores the mailbox to its basic Unix format. This program (`restore_mail`) can be installed in a `.profile` or log-on file to automatically restore the mail file if the current environment does not support X-Windows.

5.5 Summary

Systems that enhance email have concentrated on what is *possible* through electronic mail. Their potential is impressive: intelligent filtering of messages; management and coordination of projects; active assistance in managing commitments. Although such support is *possible*, its requirements and dependence on user actions renders it largely unrealistic. The realities of work pressure, lack of access to compatible tools, mistakes, even laziness, hinder the systems' ability to access the guidance upon which enhanced facilities depend.

Mona adopts an alternative, guidance-free, approach to collaboration support through email—it provides conversational context without dependence on explicit user guidance. Although conversational interpretations resulting from *Mona*'s heuristics are weaker than those determined by user-supplied guidance, they are obtained automatically, cost users nothing, and are available even when users fail to supply guidance or use differing email systems. *Mona*'s limitations (many of which are intentional) and potential directions for further work are discussed in Chapter 7.

Reducing user-effort is of obvious benefit to users, but *Mona*'s inferencing techniques also ease problems arising from systems supporting incompatible information structures. Minimising system specific requirements reduces dependence on critical mass—a system like *Mona* offers new users benefits regardless of the number of other *Mona* users. Furthermore, by reducing system requirements (for particular styles of use, the provision of specific information, and so on), users may incorporate systems into their personal working methods with minimal

impact, utilising enhanced features, dependent on additional information, as and when *they* see fit.

Design experience with *Mona* contributed much to the advancement of the groupware design principles. There is therefore a natural and unsurprising correlation between its properties and the principles' aims and strategies. TELEFREEK, described in the following chapter, affords an entirely different type of groupware support to that of *Mona*, and follows (rather than advances) the design principles.

Chapter 6

TELEFREEK: Integrating collaboration support

6.1 Introduction

Mona's main research contribution is its demonstration of a guidance-free scheme for providing conversational context: it addresses and reduces the problems encountered by guidance-dependent groupware. In contrast, TELEFREEK, the system described in this chapter provides an entirely new type of groupware support: it is an integrated environment for collaboration support. Related systems provide subsets of TELEFREEK's functionality, but typically depend on prohibitively high entry-level technology, and fail to fully exploit the potential of computing facilities.

TELEFREEK's development was stimulated by the fourth groupware design principle (for maximised external system integration). In following the first three principles, TELEFREEK adopts a variety of schemes which include: personal tailorability; the use of information sources and computing facilities that are freely available to networked computers; and the avoidance of constraining work practices.

Insights arising from the development of TELEFREEK and its prototypes, and observations of related groupware are used to record human factors that affect collaboration. These factors particularly focus on the problems of establishing contact, and form a design foundation for future work on heterogeneous collaboration platforms.

6.1.1 Chapter overview

The issues addressed by TELEFREEK are introduced in section 6.2 with a critical summary of related groupware.

TELEFREEK and its prototype are described in section 6.3. TELEFREEK is a heterogeneous collaboration platform that, unlike related groupware, uses the capabilities of currently ubiquitous networks. It supports valuable facilities within local-area-networks, and demonstrates the potential of heterogeneous collaboration environments.

Section 6.4 records human factors affecting two general problems in collaboration:

1. finding suitable colleagues;
2. establishing contact with them.

The level to which TELEFREEK satisfies these factors is discussed, and is used to forward directions for further work.

Further work, including an technique-outline for extending TELEFREEK to the global Internet, is described in section 7.6. A user-guide for TELEFREEK is provided in appendix C.

6.2 Limitations of “social” and “seamless” groupware

Section 2.5.3 reviewed groupware that supports a sense of community through facilities for “social browsing” and “community awareness”. Groupware providing “seamless workspaces” for integrated work environments are reviewed in section 2.5.2.

The limitations and problems encountered by these systems contributed to TELEFREEK’s development. These issues are reviewed in the following sections.

6.2.1 Imitation of proximity or augmentation of collaboration?

Seamless communication environments and social presence groupware, predominantly support synchronous interaction through high-bandwidth channels. The use of high-bandwidth communication mechanisms is motivated by the following observations:

- Research has noted the important communicative role performed by subtle cues such as facial expression, gesture, and tone of voice (Tang, 1991; Tang & Leifer, 1988; Kiesler

et al., 1988; Smith *et al.*, 1991; Tatar *et al.*, 1991). This information is best transferred through rich and unconstrained communication media.

- Face-to-face interaction is (*almost* always) the most efficient communication technique. Exactly replicating it through high-bandwidth channels would be beneficial.
- People are comfortable with, and have a wealth of experience in, face-to-face communication.

These issues are not contentious. What is, however, is the presupposition that *imitating* face-to-face interaction is therefore the most appropriate interaction paradigm for distributed collaborative work (Hollan & Stornetta, 1992).

This assumption fails to recognise the essential differences between face-to-face interaction and its telecommunicated imitation. While it is tempting to believe that face-to-face interaction can be adequately simulated given sufficiently rich communication media, research evidence suggests that this is beyond current (widespread) capabilities: “it is often (though not always) the conclusion of studies that the audio/video medium is much closer to the audio-only medium than it is to the face-to-face condition” (Hollan & Stornetta, 1992), page–120.

Imitations are usually second-best to the real thing. According to Novick and Walpole(1990), groupware’s attempts to replicate interaction processes is, “analogous to using current technology to build a mechanical horse rather than a car”, page–230. A better, less constraining, approach would be to investigate the new facilities that modern technology enables (Egido, 1988), and to use these to enhance collaboration.

6.2.2 Other problems

Groupware for social awareness and seamless interaction environments encounter other problems and limitations, several of which are derived from their dedicated use of high-technology mechanisms.

Restrictions on the user community. The base technology employed by these systems restricts both the range of potential collaborators, and the range of tasks supportable. Almost exclusively video connections are pivotal to the facilities provided, thus, colleagues and

potential colleagues without video equipment are inaccessible. While not condemning high-technology facilities (such as video and active badges) as collaboration enhancing technologies, the base technology of integrable systems should cater to the *lowest* common denominator, thus avoiding (as far as possible) technology-based restrictions on who to collaborate with.

Technology overkill. High-bandwidth communication facilities can be wasteful, expensive, and redundant. Low technology facilities may often provide equivalent functionality, allowing the valuable bandwidth to be reserved for communication functions in which it is truly required. An example of inefficient use of video channels was observed in trials of the CRUISER (section 2.5.3). Users often established continuously open video channels to empty offices, allowing the colleague's arrival to be seen in peripheral vision. Surprisingly, on arrival, the colleagues typically walked between offices to engage in face-to-face collaboration—the dedicated high-bandwidth video channel effectively transmitted one bit of information! The CRUISER designers used the term “ambush” to describe this use of video channels (Fish *et al.*, 1992).¹

The interaction style. Typically, these systems explicitly support a single, synchronous, interaction style. Synchronous systems should minimise restrictions on their use by incorporating asynchronous capabilities—for example, telephone answering machines enable asynchronous interaction through an essentially synchronous media.

Lack of integration. Many of these systems provide isolated support environments; colleagues engaged in collaboration will have difficulty (or be prohibited from) incorporating other systems into their communication. Furthermore, few promote the need for channel switching (swapping between interaction media, discussed further in section 6.4.2).

Scalability. Many of the systems are bound to small, tightly connected, sites. Those wishing to access collaborators beyond these sites must do so through different systems.

Addressing assistance. Because these systems are developed for tightly connected sites, they assume collaborators know how to contact each other—addressing assistance is not ex-

¹TELEFREEK, described in section 6.3, includes a low-technology “ambush” facility.

plicitly provided. Information on communication access beyond these sites (perhaps through the global Internet) would best be served by addressing mechanisms that are integrated (or incorporated) into communication systems.

6.2.3 Synergy in collaboration support

The groupware systems summarised in section 2.5 typically focus on *one* of the following styles of user-support.

1. **Social browsing for casual interaction** — promoting the formation and maintenance of working relationships (see section 2.5.3).
2. **Whereabouts and availability for directed encounters** — assisting users with specific needs for collaboration (see section 2.5.3).
3. **Seamless interaction spaces for improved workspace integration** — enabling *some* communication techniques and tools to be used in conjunction (see section 2.5.2).

Functionality converges between groupware in these categories, but the systems have almost unanimously supported isolated communication mechanisms, and are typically dependent on video.² Although each of these domains *can* be supported independently, to realise their full potential, it is necessary to exploit the mutual-enhancement available by combining these support functions. Furthermore, to maximise the range of collaborators accessible through groupware, its infrastructure should be founded on the lowest common denominator of technology.

TELEFREEK, described in the following section, offers an unconstrained and extensible platform for communication requirements and resources. It merges access to facilities for social awareness, directed encounters, communication resources, and handy utilities. Each of TELEFREEK's facilities is independently useful, but its value is primarily derived from the synergy it achieves by drawing together communication support.

²EuroPARC's RAVE (Gaver *et al.*, 1992) (section 2.5.3) integrates many facilities, but is explicitly bound to synchronous and video support; the MOCCA project (Benford *et al.*, 1992) discusses the properties that should be displayed by integrated environments, but falls short of any implementation.

6.3 Integrated collaboration support in TELEFREEK

TELEFREEK's design is founded on four primary observations arising from the limitations of related groupware (described above):

1. The issues of social presence and integration of communication methods are closely related and mutually dependent—they are too tightly intertwined to be most effectively supported by separate systems.
2. Previous systems providing social presence information have done so in too passive a manner. Their failure to explicitly cater for users' active social presence requirements (such as notifying when individuals arrive) is reflected by unexpected and inefficient use of communication media (Fish *et al.*, 1992), see section 6.2.2.
3. Most social presence and integration systems have assumed the availability of high-bandwidth communication mechanisms and specialised equipment, and have based their functionality around them. These systems appear to surmise that video links are necessary to integrate working methods, or to provide social presence. Little attention has been paid to the integration of existing communication facilities or to making the most profitable use of low-bandwidth technology to support social presence (which is all most people have).
4. Many resources for communication, collaboration, social presence, and so on, are already available on networks. These offer valuable facilities, but their lack of integration renders them inaccessible to the majority of computer users. Access to such facilities should be merged, and doing so promotes synergy in communication and collaboration support: it is natural that, for example, information *about* communications (email addresses, surface mail addresses, and telephone numbers) should be combined with *access to* communication mechanisms and facilities.

TELEFREEK addresses the issues raised by Bair (1989):

“How can we extend electronic media to meet users' needs?... we know that the use of a variety of integrated media, selecting each for the appropriate purpose, is the ideal situation. The availability of an *integrated portfolio of media...* is my long-term vision”, page-211

While independent holistic communication support packages will remain a far-flung ideal for years to come (Kraut *et al.*, 1988b), heterogeneous environments can draw together access to the facilities (such as communication mechanisms and information sources) upon which communicants rely, consequently providing “integrated” platforms for collaboration assistance.

Avoiding the high-technology entry level ubiquitously displayed by related groupware, TELEFREEK uses resources *freely* available on Internet network computers. It provides a heterogeneous portfolio of information about, and access to communication resources, together with social presence guidance. It also allows users to customise and tailor their access to personally favoured utilities.

TELEFREEK’s design guidance

The four groupware design principles (chapter 4) prompted recognition of the lack of support for integrated communication infrastructure. The principles further guided the implementation of TELEFREEK’s functionality. In achieving TELEFREEK’s primary objective (pooling access to existing communication resources) the groupware design principles are rigorously upheld:

- One user’s rejection of TELEFREEK does not affect the benefits available to others (see section 4.3.2), and customisation facilities promote personal appeal (section 4.3.1).
- User-benefit is provided through resources *freely* available to networked computers (see section 4.4.2).
- There are no explicit requirements or constraints in its use (sections 4.4 and 4.5), and it is highly flexible and adaptable.
- TELEFREEK’s entire aim is to integrate access to communication and collaboration support—this ambition is in direct concordance with the fourth groupware design principle (see section 4.6).

6.3.1 System overview

Unlike the majority of groupware systems, TELEFREEK does not directly provide a medium for collaboration. Instead it provides access to a variety of media, and offers information that

guides the selection of appropriate channel. The goal is to ease the individual's selection of whom to collaborate with and how to maintain the communication, and to ease transitions between interaction mechanisms. It also provides an easy to use and extensible interaction platform.

The TELEFREEK prototype runs under Unix and the X Window system (Scheifler & Gettys, 1986). An initial prototype demonstrating the look and feel of TELEFREEK and mimicing the proposed functionality was developed in HyperCard (Apple Computer, 1987; Thimbleby *et al.*, 1992) (figure 6.1). During demonstrations of the HyperCard prototype, extensive comments on potential design directions were received prior to starting implementation of the Unix based system.^{3 4}

Filters		Named User?	Active Users	Interrupt Stat
U-Name	Friends	Activity/ Mins		
alan		Away	Telephone ?	E-Mail
agc	Unix/C Expts	Away		
amc		Typing	Fax	Shared Term ⏏
gco	CSCW Class	Asleep 20		
gwhr		Typing		
hwt	Stirling	Typing	Talk ⏏	H-Memo ⏏
jpg		Asleep 7		
lmc	Climbers	Away		
sam	Sam Nelson	Typing		
spm	Steven Marsh	Asleep 1		
srj	Steve Jones	Away		

Figure 6.1: The Hypercard prototype of TELEFREEK.

The user's activity status (*typing*, *asleep*, *away*) could be inferred from the time since their last keystroke. Selecting a particular user with the mouse raises icons over particular communication media, providing advice on appropriate channels for interaction.

³The prototype and its informal evaluation were carried out with Saul Greenberg and the KSI Seminar group at the University of Calgary, Canada.

⁴TELEFREEK's current implementation works, benefits users, provides an attractive interface, and is extensible, but its range of facilities is restricted. Extensions to TELEFREEK are discussed with future work in chapter 7.

To maximise portability and flexibility almost all processing of TELEFREEK's utilities is delegated to sub-units that are independent of the core system (in the Unix environment these units are shell scripts). Thus, TELEFREEK is a graphical user interface facility with minimal "hard-wired" functionality.

Figure 6.2 shows the main window of TELEFREEK.⁵ The left hand side of the window deals primarily with social presence information, and the right offers heterogeneous access to a variety of communication and subsidiary (optionally personalised) information facilities. TELEFREEK's four main interface components are described below:

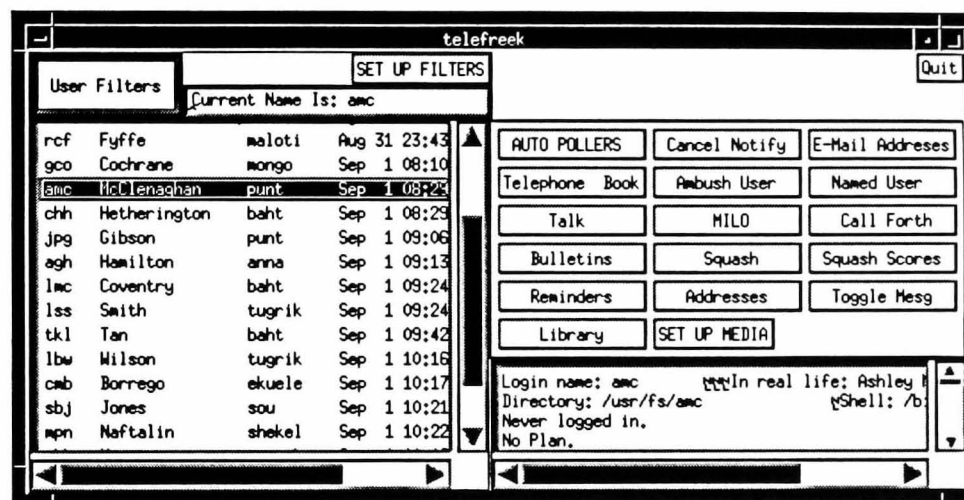


Figure 6.2: The main TELEFREEK window.

The community list

The scrollable list on the left of TELEFREEK's window displays an optionally filtered view of the user community (filters will be described below). Each line, which is updated every few minutes, shows a user name, their real name, the machine they are on, and the time they logged on at. People in the list can be "selected" by clicking on the display line that shows their name, and this causes information about them (by default their `finger` information) to be displayed in the "Information display" (see below). Selecting a user also makes them become the current default for communication or further query (for instance, the default `Talk` person, or `Telephone Book` search name). The name of the current user is displayed just right of the `User Filters` button, in the display `Current Name Is:xxx`.

⁵Details of how to use TELEFREEK are provided in the appendix C.

User filters

The button on the top-left of figure 6.2 displays a pull-down menu that shows a set of community sub-groups: see figure 6.3. Selecting a particular group causes the community list to be modified so that it displays only those group members who are currently logged-onto the network.

TELEFREEK's filtering mechanisms draw on observations of the use of social presence groupware. The "user ambushes" observed during CRUISER use (see sections 2.5.3 and 6.2.2) suggest that passively stating who is about (social browsing) is insufficient for many communication needs. People not only wanted to know who is about, they also wanted to be notified when colleagues arrived. Similar and related issues were observed in the use of PORTHOLES (Dourish & Bly, 1992). Its users criticised its video representation of social presence for not providing enough active information. It required too much screen space and provided too little dynamic information to warrant constant screen space, but calling up the window to reveal social presence required too much effort: "[the window] takes up too much space on my screen to be up continually so it's overhead to use it" and "not much happens; the turn around for new information is so slow that I'm not too much motivated to use it; I'm never guaranteed of seeing much", (Dourish & Bly, 1992), page-545.

Social presence information must, therefore, satisfy the users' dynamic requirements for awareness knowledge about particular people (groups, individuals, or both), and must do so as easily (in terms of user-effort) and efficiently (in terms of communication transmission costs) as possible. Consequently, TELEFREEK supports a variety of facilities for social awareness: users can browse the entire network community, or they can maintain awareness of particular social or task groups; directed encounter assistance, through which the system announces the arrival of particular people or group-members, can also be requested.

Social browsing and directed filtering

TELEFREEK supports user-defined filtered views of the current community of network users. Although the entire community can be displayed, filtered views reduce the information overload caused by large communities. Particular community filters are requested through the filters menu (figure 6.3), and the menu can be customised to reflect each user's personal interests and needs (both social and task-specific). For example, one filter might monitor all



An example filters menu

Additional menu option resulting from changes to the filters form (see figure 6.4)

Figure 6.3: Modifying the filters menu.

people with a knowledge of “C”, and another might maintain awareness of squash players.

Filters are customised by clicking the SET UP FILTERS button (top centre figure 6.2). This raises a pop-up form (figure 6.4) that binds menu-items to variants of the Unix `who` command. Adding the highlighted line in figure 6.4 causes the menu change shown in figure 6.3.

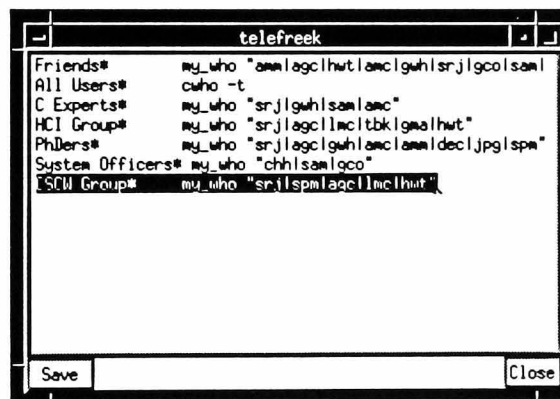


Figure 6.4: Editing the filters form modifies the filters menu.

Directed encounters and active notification

Active announcement of the arrival of particular group members or individuals is also supported in TELEFREEK. The announcement is made by an audible beep, changing the TELEFREEK icon (see figure 6.5), and by displaying the names of those who have arrived in the information display (bottom right of figure 6.2). This facility is directly analogous to the “user ambushes” observed in the use of CRUISER (described above). The set of users included in an “ambush” are specified through a pop-up form that appears when the Ambush User(s) button is clicked (see figure 6.2).



Normal icon



Passive event notification icon

Figure 6.5: TELEFREEK icons.

The information display

At the bottom right of the TELEFREEK window is a read-only scrollable information display. This is used to display information about selected users, and as a user-feedback area for many of TELEFREEK's utilities.

Utility buttons

The array of buttons on the right of figure 6.2 provide a customisable and expandable set of functions. It is through these buttons that most of TELEFREEK's communication media and information sources are accessed. There is no restriction on the functions carried out by buttons; users can add, remove, and modify the functionality of buttons (see section 6.3.2). Typically, the actions of TELEFREEK buttons fall into general categories:

- Communication mechanisms—for instance the **Talk** button launches a Unix **talk** or **write** conversation with the currently selected user (as shown in the display top-centre of figure 6.2). The **write** conversation is mediated through a pop-up window which is automatically removed when the conversation terminates. Similar functions could be added to access email, and, given the equipment, video-call facilities.
- Groupware applications—for instance the button **MILO** launches the co-authoring system MILO (Jones, 1992a; Jones, 1992b).
- Establishing links to external sites—the buttons **Call Forth** and **Library** execute command sequences that dial up external computer accounts and display the communication in a pop-up window.
- Access to information sources that are relevant to communication—the buttons **Telephone Book**, and **Email addresses**, look up the current user in the relevant files, and show

the results in the information display. `Named User` allows user-queries through a variety of naming conventions (login name, surname, first name).

- **General communication and social functions**—the buttons `Reminders`, `Bulletins`, and `Squash` execute commands, and display their results either by popping-up windows, or by displaying their output in the general TELEFREEK information display. It is important that access to these general social awareness utilities (for instance encouraging social interaction through the squash challenge ladder) is integrated within TELEFREEK; users will be more likely to keep up to date with such social activities if the overheads of doing so are reduced.
- **Interrupt status**—the button `Toggle Mesg` controls the ability of others to make contact through the `Talk` or `write` facilities. A message reporting the current access status is provided in the information display.

Other system specific buttons provided by TELEFREEK are:

- **AUTO POLLERS**—this button accesses a pop-up form showing the commands that require TELEFREEK to periodically check for status changes: for example, the arrival of mail or a bulletin announcement. Announcement of these events is made by changing the TELEFREEK icons (see figure 6.5), by giving an audible beep, and by announcing the relevant event in the information display;
- **Cancel Notify**—cancels event notification, and reverts the TELEFREEK icon to its normal state;
- **Ambush User**—notifies the user when an individual or any member of particular sub-groups arrives (see section *Active filtering* above);
- **SET UP MEDIA**—used to customise the **Utility buttons**, see section 6.3.2.

6.3.2 Extendability, flexibility, and customisation

Communication requirements vary between individuals, and each person's requirements are dynamic. Similarly, the facilities available on a network are not static: new computer based facilities become available and colleagues find more efficient ways of doing things. Systems

should support and enhance the incorporation and extension of facilities; they might also actively encourage users to share popular facilities (see section 7.6.1).

An alternative, but equally important view of system flexibility is the balance between system power and its ease of use. Borenstein and Thyberg investigate this issue in groupware (Borenstein & Thyberg, 1991), and conclude that while the ubiquitous trade off between power and ease may be acceptable in single user interfaces, the differences between groupware users' requirements allows no such trade off. TELEFREEK's aims to provide ease-of-use through its core functionality (community-list, filters, ambush facilities), while power is supported through its ability to be customised and extended.

The customisation of TELEFREEK's community filters was described in section 6.3.1: the changes to the filters menu shown in figure 6.3 are caused by adding the highlighted line shown in figure 6.4. TELEFREEK's utility buttons can also be extended and adapted to reflect personal interests and requirements. These modifications are made by altering the shell script calls held in the pop-up media-form that is accessed through the SET UP MEDIA button, as exemplified in table 6.1.

Consider the actual occurrence of a University's library becoming accessible through a computer network. The command to access the library was posted on the network bulletin board (an event notified by, and accessed through, TELEFREEK). To incorporate this facility into TELEFREEK the user clicks the SET UP MEDIA button, and a pop-up window containing the current media-form is displayed. The user adds a new line to the file. The components of the line are the label of the new button, `Library`, a separator symbol, `*`, and the Unix command to request a new window and launch the library call `xterm -display $DISPLAY -sb -e rlogin library -l janet&`, (see figure 6.6). Having saved the changes, the set of utility buttons are re-drawn, and selecting the new button `Library` will invoke a new window, and call-up the library within it.

Table 6.1: An example extension to the functionality of TELEFREEK.

System administrators could add new facilities to TELEFREEK, thus saving large numbers of users from doing so independently. Such alterations to TELEFREEK's core functionality are similar to those taken by each individual user, but may have negative implications for system use. Users are likely to be disturbed by externally imposed alterations to *their* systems (Cool

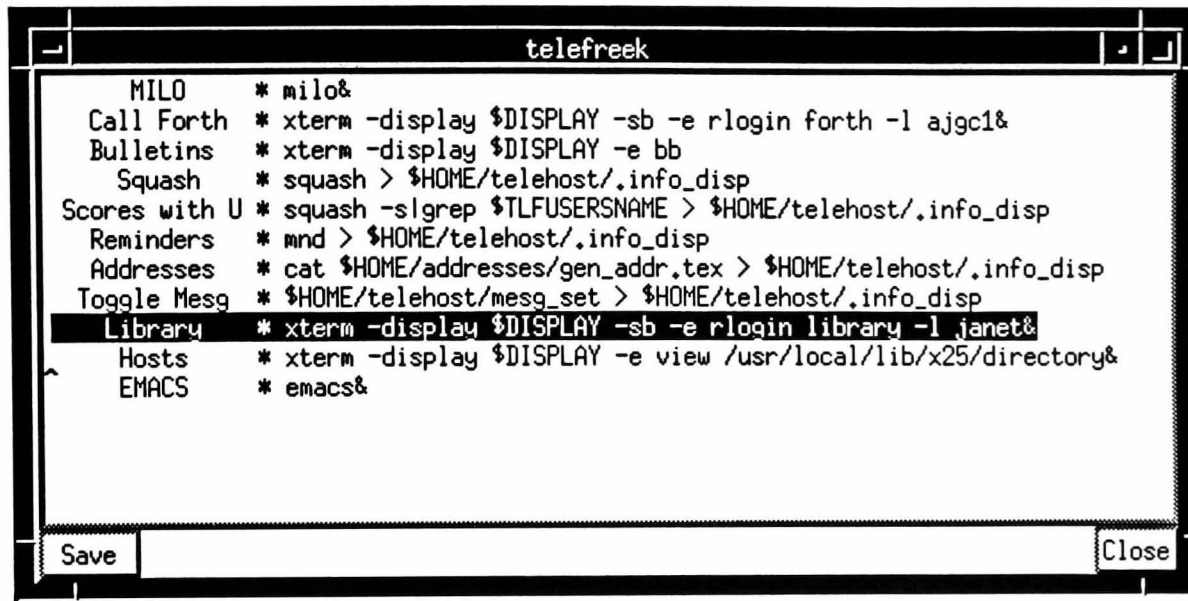


Figure 6.6: Adding a “call library” function to TELEFREEK’s media form.

et al., 1992), consequently, a high level of consistency should be maintained.

6.3.3 TELEFREEK as a prototype

Although successful as an independent system, TELEFREEK’s primary research ambitions are to demonstrate a heterogeneous collaboration environment, and to exemplify the groupware design principles. In contrast to related groupware, it merges access to facilities (such as social browsing, directed-encounter assistance, and integration of communication media) *without* depending on restrictively high-bandwidth technology. TELEFREEK also allows users to customise and extend its features.

TELEFREEK is an exciting and thought provoking system, but it is, in essence, a prototype. This prototype status is derived from the facilities that it *almost* achieves, rather than from those that it fails to support. Its most substantial limitation is the restriction of its social presence information to local-area-networks. A technique outline for extending TELEFREEK to the global Internet, and other directions for further work, are provided in section 7.6.

6.4 Determining *who?* and *how?* in communication

During the development of TELEFREEK and its prototype, user-requirements for heterogeneous collaboration environments were evolved and clarified. Information sources for this process included comments gathered during interactive demonstrations of the HyperCard TELEFREEK

mock-up, and the rapidly expanding CSCW literature that describes related groupware and its use (see section 2.5).

Imagine a common situation in which Jane, a computer supported worker, requires a quick answer to a problem—she reaches for the telephone and dials up a few colleagues who might be able to provide an answer. The colleagues are away from their telephones so messages are left on their answer-phones requesting that they call back. Jane’s initial question remains unanswered so she turns to e-mail, posting a message to a group of people who are likely to be able to help. Soon, a message is returned which, though not providing an exact answer, raises some interesting related issues. Several e-mail messages are rapidly transferred in a synchronous style (but via a frustratingly narrow bandwidth) until one of the users sends the message “What’s your phone number there?” With true synchronous communication established the issue is quickly resolved.

Jane now suffers the consequences of her earlier attempts to establish communication. Returned telephone calls and further offers of assistance from receivers of the initial email plea distract her and add to her burden of information overload.

Table 6.2: A typical “synchronicity scenario”.

Several of the problems addressed by heterogeneous collaboration environments (particularly those concerning making contact) are exemplified in the “synchronicity⁶ scenario” (table 6.2).⁷ The first problem is getting in touch with the right people. The specific needs of communication initiators—perhaps an immediate answer—are thwarted by inadequate knowledge about suitable recipients. They need to know *who* is around, whether they are available for interruption, their pertinence for the current issue, whether the recipient’s social status is appropriate for the communication task, and so on. The second problem is *how* they contact the person and subsequently mediate the communication, issues include: what style of communication is most appropriate (synchronous or asynchronous); what facilities does the recipient possess; and which groupware tool best fits current requirements.

⁶“Synchronicity” was first used by Carl Jung to describe “... a coincidence in time of two or more causally unrelated events”, translated in (Jung, 1952).

⁷The term “synchronicity” is used here to encapsulate many of the problems of making contact. The term is appropriate for two reasons: first, groupware’s functionality is frequently founded on artificial barriers between synchronous and asynchronous modes of working (Rhyne & Wolf, 1992); second, failure in making contact may lead people to feel there is a conspiracy (social, or perhaps meta-physical as Jung suggests) which confounds their success.

- 1. Factors in selecting the right people for communication.**
 - a) Social presence plays a dominant role in the formation and maintenance of working relationships.
 - b) Communication needs can dictate the pertinent community members.
 - c) People need to contact particular individuals.
 - d) Social status affects rights to “whereabouts” knowledge, and rights to initiate communication.
 - e) The recipient may not want to be interrupted by the caller.

- 2. Factors in selecting the right communication channel.**
 - a) Tasks strongly influence the minimally acceptable communication channel.
 - b) Communication mechanisms are restricted to those mutually accessible.
 - c) The intended period of interaction affects the choice of communication channel.
 - d) Inertia can inhibit people switching to more appropriate communication channels.

Table 6.3: Human factors affecting selection of people and mechanisms in communication.

The human factors affecting selection of people and mechanisms for communication are elaborated in the following sections, and the extent to which TELEFREEK satisfies them is discussed. These factors form a design foundation for heterogeneous collaboration environments, and direct extensions to TELEFREEK. They are summarised in table 6.3, adapted from (Cockburn & Greenberg, 1993).

6.4.1 Selecting the right people for communication

Part one of table 6.3 summarises the human factors affecting the decision on *who* to collaborate with. In providing a communication infrastructure, groupware systems (such as TELEFREEK) should consider these issues to best satisfy the correspondents’ desires.

Social presence plays a dominant role in the formation and maintenance of working relationships. Social presence is concerned with the whereabouts and availability of potential communicants (Kraut *et al.*, 1988a). Inadequate access to this type of information causes frustration and information overload, as illustrated by the “synchronicity scenario” (table 6.2). In many situations real-time communication is the most desired, but the least available. Troublesome and annoying difficulties, such as telephone-tag, can make users resort to less rapid, but more reliable mechanisms: email, post-it notes, or even fax. Although responses are not immediate, receipt of the message is virtually guaranteed—but now the

recipient must find the originator.

Before the advent of networked computers, access to social presence information was limited to such means as walk-about, clock-in cards (stating where people might be, not necessarily where they are), or by actually establishing contact. Networked computers, however, readily provide a variety of information about the community of users. Other modern technologies, in conjunction with computerised techniques, allow closer and more continual updating on the whereabouts and activities of individuals: for example, see the review of active badge technology in section 2.5.3.

Social interaction has been shown to play an important role in the formation and maintenance of working relationships (Kraut *et al.*, 1988b; Kraut & Streeter, 1990; Root, 1988; Fish *et al.*, 1992). Allowing constant and easy access to “who’s about” enhances awareness of the proximity of co-workers, potentially prompting collaboration which may involve a stroll down the corridor, a phone call, or the use of groupware. Groupware should therefore promote users’ awareness of who is reachable through the network, who is currently accessible, and should facilitate easy access to them.

TELEFREEK promotes social browsing through its community list (see section 6.3.1). The entire network community can be displayed, or the community can be filtered to reveal socially-oriented sub-groups (friends, squash players, and so on).

Communication needs can dictate the pertinent community members. Often the key motivation for communication is the necessity for an immediate solution to a problem. In such situations the social aspects of who to interact with are subordinate to the time pressures in receiving an answer. Although blanket addressing may resolve problems quickly, there is a substantial cost in the form of wasted time and interruption to the majority of those reviewing and answering the request. Even when particular individuals are known to be capable of resolving the issue, they may not be available, so alternatives must be pursued.

Social presence information increases the user’s ability to direct their urgent queries by showing those currently available. Computers can further assist by directing users to the people most likely to be able to satisfy specific needs. Computer supported schemes for finding pertinent people include the following:

- Databases of information about network users, what they do, and what groups they

belong to—X.500 directory user agents provide some of this information across the Internet (Prinz & Pennelli, 1991; Iannella, 1992).

- Community filters providing awareness of relevant experts (Kraut & Streeter, 1990)—as supported by TELEFREEK (see section 6.3.1).
- Expert systems that automatically answer previously encountered queries and find appropriate experts for new ones: for example, COKES (Kaye & Karam, 1987) and the *Answer Garden* (Ackerman & Malone, 1990).

As a side effect of improved accuracy in communication addressing, information overload is reduced; fewer people receive messages and consequently there will be fewer delayed (and probably redundant) replies.

People need to contact particular individuals. The selection of communicants will often be pre-determined by particular tasks and needs. When the needs are highly specific, it is likely that work can only continue with particular individuals: for example, when collaboratively writing a paper the only relevant communicants are the other co-authors. Groupware should therefore facilitate finding and contacting particular people.

TELEFREEK provides two mechanisms for directed encounters. The **Named User** button (section 6.3.1) allows queries about particular people through a variety of naming conventions (login-name, surname, first name). The **Ambush User(s)** facility (section 6.3.1) allows users to request notification when particular individuals (or groups) log onto the network.

Social status affects rights to “whereabouts” knowledge, and rights to initiate communication. The relative social status of communicants can raise complications stemming from protocols (formal or informal) for communication through organisational hierarchies. While it may be acceptable for managers to phone subordinates, the reverse need not be true. When it could be an issue, groupware should make people aware of the social status of their potential contacts.

While social status information is not explicitly supported by TELEFREEK, its customisation facilities provide hooks through which file-access functions can be added. It also supports environment variables that contain information about the currently selected user. Combin-

ing these capabilities, facilities that access the relative social status of colleagues could be incorporated into TELEFREEK.

The recipient may not want to be interrupted by the caller. Attempts to establish communication commonly cause unwanted interruptions. This is typically unsatisfactory to all parties—the caller is sorry for the interruption, and the receiver must recover the previous train of thought or action. Personal assistants provide a manual solution to this problem (for those who can afford to employ them), but this is rarely the norm for computer mediated communication.

Computers can offer a more accessible alternative. First, groupware should allow receivers to accept or reject the call—reflecting the common use of answer phones as filters for incoming phone-calls. Second, users can record their interrupt status. Callers can review this status, and use their judgement to decide whether interruption is warranted. Alternatively, systems can filter incoming messages using a variety of information sources (social status of caller and recipient, urgency ratings, interrupt status, and so on). Through this method, priorities can be automatically assigned to messages, and thresholds used to announce or reject communications: ISCREEN (Pollock, 1988), described in section 2.2.1, combines some of these facilities to provide an “intelligent office assistant”.

TELEFREEK, as described above, accesses the Unix command `mesg` that permits or denies messages to a terminal (see section 6.3.1). More sophisticated interrupt control mechanisms could be added through TELEFREEK’s customisation facilities.

6.4.2 Selecting the right communication channel

Human factors affecting the choice of communication channels and mechanisms include the following:

Tasks strongly influence the minimally acceptable communication channel. The collaborative task will, to a large extent, dictate the appropriate communication mechanism for information exchange. For example, if the task is transferring a textual document to another person, email would be the preferable to telephone dictation, and for collaborative editing, a real-time shared editor and voice link would be superior to email. While the less preferred mechanisms are capable of task mediation, they cause bottlenecks in the work.

It would be simple and tempting to request the highest quality network channel, for these are most likely to satisfy the users requirements in rate of information exchange (Kraut *et al.*, 1988a). Succumbing to the temptation of high-bandwidth technology can, however, be wasteful of resources. What the user *needs* is the minimally sufficient channel for task and user satisfaction. Increasing the bandwidth of channels, while appealing, does not necessarily assist collaboration and is consequently not cost-effective (Gale, 1991; Fish *et al.*, 1992; Tatar *et al.*, 1991).

Groupware should allow users to access a variety of communication media, and collaborative work support tools, facilitating their choice of appropriate support mechanisms. Although TELEFREEK does not actively assist the selection of communication channels, it pools access to a variety of media. By presenting the alternative mechanisms within a single communication platform, it is intended that the users' selection of the most fitting interaction device will be facilitated.

Communication mechanisms are restricted to those mutually accessible. Obviously, communication facilities are restricted to those available at the initiator's site. More problematical for the initiator, however, is knowing the facilities accessible to the receiver. For example, making a video-dependent call is pointless if the person called does not have a camera wired to the network!

Groupware supporting social presence information can go some way towards easing these problems. It can note what equipment is available, and suggest appropriate mechanisms. It can also provide alternate ways to contact someone: having noted that a colleague is away from the office, the relevant home or mobile telephone numbers could be offered.

Related facilities were demonstrated in the HyperCard TELEFREEK mock-up (figure 6.1). Having selected an individual for interaction, the mock-up "accessed" his/her activity level (this could be inferred from idle time information supplied by, for instance, the Unix `rwho` command). The system then "suggests" suitable communication channels: for instance, having observed that the user is away from the office, the system would provide email addresses and home telephone numbers. TELEFREEK does not support these facilities (primarily because the `rwho` daemon `rwhod` is disabled in its current environment): this limitation in TELEFREEK, and others, are discussed more fully in section 7.6.

The intended period of interaction affects the choice of communication channel. The period of interaction is the cycle time related to various aspects of interaction. Before computers, the four most common means for interaction (aside from face to face) were: the telephone, for synchronous and immediate contact; the telegraph, for rapid asynchronous communication over wide distances; memos, for local asynchronous contact with a quick turnaround; and surface mail, for communication tolerating a turnaround time of several days. Technologies, such as facsimile machines and networked computers, have made several new channels of communication available, increasing both the choice of media and the range of supportable interaction periods. Collaborative workers are now at liberty to select a medium that satisfies their need for feedback.

The period of interaction suggests several crucial factors affecting selection of media:

- *Task period*—the time over which each unit of task activity must be completed: for example, turnaround in one co-author's revisions to a paper.
- *Environment period*—the delay imposed by the environment: for example, time-zones, or the fact that it is usually socially unacceptable to telephone someone at 3AM.
- *Propagation period*—the delay imposed by the communication technology and how often people check the channel for activity: for example, this period is low for telephone interaction, high for surface mail.
- *Perceptive period*—the maximum delay that users feel is tolerable for successful interaction. In real time communication the perceptive period is likely to be a common factor between all participants: for example, a three second delay on a satellite audio link is likely to disrupt conversation and therefore be equally unacceptable to all conversants. However, when using asynchronous technology (such as email) there could be a conflict in the perceived period: for example, while one correspondent may be content to take two weeks responding, the delay could frustrate others.

The duration of these periods are highly variable and subject to exceptions. It is therefore unreasonable to expect any single groupware application to establish a balance between periods *imposed* (by the environment and support technology) against those *desired* by correspondents. An alternative approach is for the computer to make users aware of the available

range of mediums (as achieved by TELEFREEK's merged access) and their different period characteristics. Because the desired period of interaction can change over time, the computer can also encourage media-switching when channels, more fitting to the current interaction pace, are available.⁸

Inertia can inhibit people switching to more appropriate communication channels.

The ubiquitous and fluid nature of transitions between communication media deserves noting as the final factor affecting the choice of mechanisms for interaction. Messages received on one device (such as a telephone) commonly stimulate further messages using a different and more appropriate mode of interaction (such as email). Conversely, inertia in working methods (in this case, communication techniques) frequently results in inappropriate communication mechanisms being maintained for no express purpose. This phenomenon is exemplified in the "synchronicity scenario" (table 6.2) by the rapid exchange of email before the comment "what's your telephone number there?"

Implications for computer systems from this observation are twofold. First, systems should allow users to easily swap between communication mechanisms: for example, on receiving email, the system may offer a variety of appropriate means to reply (perhaps using a "knowledge" of the sender's availability). Second, systems could take an active role in easing the communicants' transitions between communication mechanisms: for example, observing conversational breakdown, and suggesting more appropriate interaction devices.

With few exceptions, intelligent adaptive interfaces in *single-user* applications have had nominal success (Woods, 1993). Applying such techniques to groupware, with its additional complexity, is therefore speculative (and beyond the scope of demonstrational systems such as TELEFREEK). However, dynamic flexibility in groupware is examined through techniques of computational reflection in (Dourish, 1992a).

6.5 Summary

The fourth groupware design principle (section 4.6) noted the lack of integration between the currently available computer facilities. Heterogeneous collaboration environments that merge access to such facilities were forwarded as a strategy for achieving greater integration

⁸Modifying the user interface to suit interaction pace is discussed in (Dix, 1992b).

in section 4.6.2. In following these recommendations, this chapter describes TELEFREEK: a groupware application that demonstrates the potential of such heterogeneous collaboration environments.

TELEFREEK's functionality focuses on three mutually-enhancing support styles: social browsing for casual interaction; "whereabouts" and availability for directed encounters; and "seamless interaction spaces" for improved workspace integration. It was noted that although previous groupware applications have supported each of these support styles *independently*, they have failed to exploit the synergy available by using the natural relationship between them. Furthermore, these groupware systems depend on prohibitive (and in some cases, inappropriate) high-bandwidth technology.

Although directly motivated by the fourth groupware design principle, TELEFREEK also follows the recommendations of the other three principles:

- enhancing personal benefit—through support for customisation and tailorability, and by avoiding a dependence on other users;
- minimising requirements—by utilising information freely available on networked computers;
- minimising constraints—by using the lowest (widely accessible) common denominator of technology.

Observations and insights arising from TELEFREEK's development, and experiences related in research literature were combined in recording a set of human factors in collaboration. These factors addressed the general collaboration issues of *who* to communicate with, and *how* to mediate interactions. The factors form a design foundation for collaboration platforms, and provide directions for further work and extensions to TELEFREEK's functionality.

Chapter 7

Conclusions and further work

7.1 Introduction

The thesis introduced four principles for groupware design. Founded on an investigation of the problems encountered by groupware, the principles provide design guidance that promotes the development of successful collaboration support systems. The principles focus on practical groupware for current use: they accept our limited understanding of collaborative processes, and the limitations of widely accessible networks. Although concentrating on pragmatic, usable, and successful groupware, the principles' recommendations are also largely applicable to research prototypes and exploratory implementations.

Two systems that provide separate styles of collaboration support were designed under the principles' guidance: *Mona* and TELEFREEK. Through its support for conversational relationships between email messages, *Mona* promotes email as a medium for collaborative and coordinated work. Its development was tightly coupled with that of the principles. TELEFREEK provides a platform that accesses facilities serving peoples' requirements and desires in collaboration.

In contrast to existing groupware applications that offer similar functionality, *Mona* and TELEFREEK use markedly different techniques to achieve their user-support. Their use of atypical schemes is a consequence of their pursuance of the groupware design principles. Although these systems intentionally restrict their functionality in order to better illustrate the principles, they are successful, work enhancing, tools which blend into the personal work environment. This ability to improve collaboration support, regardless of prototype status,

can largely be ascribed to the principle for personal acceptance, under which groupware *must* be of personal value. Independent of the design principles, *Mona* and TELEFREEK each make notable research contributions (Cockburn & Thimbleby, 1991; Cockburn & Thimbleby, 1992a; Cockburn & Thimbleby, 1993; Cockburn & Greenberg, 1993).

The following sections critically summarise the contribution of each chapter, and note some directions for further related research. The final section provides the general conclusions, and assess the overall contribution of the thesis.

7.2 An overview of CSCW and groupware

To place the research of the thesis in context, chapter 2 reviewed groupware and CSCW research. Particularly oriented towards groupware design and implementation, the range of groupware was presented in four categories: messaging systems, collaborative authoring tools, computer enhanced and supported meeting environments, and seamless interaction and social presence systems. The applications developed under each category were summarised in tables. Concerns were raised that explicitly categorising groupware can precipitate technology-driven research, and that this directs design attention away from the true ambition of groupware: the empowerment of users.

7.3 Problems in user-acceptance of groupware

Chapter 3 explored the problems specifically encountered by groupware. Until recently, groupware was ubiquitously cited as being a failure. In the early 1990s the situation is changing, but slowly, and in restricted domains. Groupware's small-scale successes have been limited to local sites or tightly-coupled remote sites. Although the difficulty of developing groupware is well known, the understanding and descriptions of groupware's complications are vague: a cost/benefit disparity; failure of intuition; evaluation problems.

From the foundation of Grudin's (1988) observations of failure, groupware's difficulties were generalised into issues of system-use, system-design, and system-evaluation. The chapter focused on system-use, examining the users' costs in terms of the effort required. It was shown that groupware encounters particular obstacles during its introduction and early stages of use. A vicious circle exists that, without close attention, confounds the prospects for groupware

success. This vicious circle enforces a mutual dependence on user-effort, user-benefits, personal system use, and social system use (critical mass). It was noted that a “kick-start” in user benefits, and a breakage in the dependencies of the vicious circle is required.

7.4 Four principles for groupware design

The observations made in chapter 3 were used in chapter 4, for two purposes. First, the failure of design intuitions and the lack of design guidance were addressed by four groupware design principles that guide system development. Second, under each principle, a set of strategies pinpoint the groupware attributes that contribute to the difficulties imposed by the vicious circle. The strategies also forward techniques for avoiding these problems.

The four principles were described as follows:

1. **Maximise the likelihood of personal system acceptance.** This principle noted the importance of catering for the individual in collaboration support. It is motivated by the observation that a society’s rejection of groupware is driven by an accumulation of individual rejections, and that, conversely, if groupware is made more acceptable to individuals (without hindering its value in group support), it will be more acceptable to the user society.
2. **Minimise the requirements imposed on users.** Concerns the requirements that users’ must satisfy for correct system operation. It was shown that dependence on user-actions in groupware imposes a cost/benefit disparity across users. Such dependence also frequently demands that third parties become involved to maintain the system’s ability to operate. The “minimise requirements” principle observed that *depending* on user-action in groupware is inappropriate. Alternative techniques for accessing the information required to provide particular groupware facilities were forwarded by the principle’s strategies.
3. **Minimise the constraints imposed on users.** The third principle was primarily concerned with the models and theories upon which groupware is built. It also addressed groupware’s flexibility. It was noted that models of communicative activities and theories of collaborative tasks are, as yet, inadequate and incapable of providing effective foundations for groupware. The principle argued that practical groupware (rather than

research systems) should adopt open and unconstraining support structures, enabling and promoting social protocols in the government of group work.

4. **Maximise the potential for external integration.** The fourth principle addresses two primary issues. First, how groupware fits into the wider work environment in which competing and incompatible systems are maintained by colleagues. Second, how computer facilities can improve satisfaction of collaboration and communication needs and desires.

Ideally the development of computer supported collaboration systems should involve all those who know most about it: computer scientists, sociologists, ethnographers, anthropologists, psychologists, users, and so on. Unfortunately, this is usually impossible, and even when feasible, it remains essential that the computer scientists responsible for implementation are aware of the relevant issues.

These *design* principles therefore concentrate on the needs of computer scientists and groupware implementors. Within this domain, the principles are both *necessary* and *comprehensive*. Furthermore, they are *beneficial*.

They are **necessary**¹ because they identify the properties repeatedly causing groupware's failure, and raise alternatives that overcome and avoid these failings.

They are **comprehensive** because they cover the full range of each system's properties: from "superficial" interface issues (addressed by "catchpenny facilities" under the personal appeal principle) to the issues of how groupware integrates with existing working practices and systems (under the maximise external integration principle).

They are **beneficial** because they foster the exploitation of existing computing and network facilities for the benefit of current users, and emphasise the importance of supplying discernible benefits to *all* system users.

¹Principles, or other devices, that impart *equivalent* design guidance would, naturally, have an equivalent effect. Although the particular phrasing in which these issues are couched is important (for clarity and comprehension), it is the issues raised by the principles, and their guidance, that is necessary. A lengthy discussion on the equivalence of principles is beyond the scope of this thesis.

7.4.1 Further work with the principles

To date, three systems, *Mona* (chapter 5), TELEFREEK (chapter 6), and MILO (Jones, 1992b; Jones, 1992a; Jones & Cockburn, 1993), have been developed under the principles' recommendations. Strategies for achieving each principle's intentions are provided in chapter 4. It is expected that as more groupware applications are developed under the principles' guidance, additional strategies will be added to those recorded in chapter 4.

As observed in section 4.7.1, the principles say little about the evaluation of groupware. Consequently, there is scope for further work on an additional principle that explicitly addresses issues of groupware evaluation. The guidance provided by such a principle would, most likely, require specialist knowledge of the social processes in which groupware is embedded, and would therefore affect a different set of researchers to those affected by the four principles described in this thesis.

Although issues of evaluation are largely outside the principles' guidance (they address design, not design *and* development), they can play an important role in evaluation. They highlight system properties for the subject of empirical evaluations (such as user-appeal, and user-requirements). As an alternative to formal system evaluation, the principles can also promote awareness and understanding of groupware problems when engaged in participatory design (see section 4.7.1).

7.5 Automatic conversational context in *Mona*

Chapter 5 described *Mona*, a groupware system that fervently follows and demonstrates the design principles. Its rigorous exemplification of the principles should be expected, considering the fact that it and the principles were developed in conjunction.

Mona promotes email as a medium for collaborative and coordinated work. Through automatic inferencing mechanisms *Mona* generates an interpretation of email conversational context. By basing its conversational support on information available in all email messages, users' receive these additional facilities "for free": *Mona* requires no additional user-effort. Conversations can be browsed in several ways, including a graphical "web" that shows the

²Specific strategies resulting from MILO's adoption of the principles have not appeared in this thesis, but can be found in (Jones & Cockburn, 1993).

multiple threads of conversational activity. Other enhanced facilities provided by *Mona* include a “reminders” facility that demonstrates the personal acceptance principle.

7.5.1 *Mona*: related further work

Mona's primary intention as a research prototype was to contrast its guidance-free schemes (that require no additional user-effort) with the usual guidance-dependent schemes. In providing this contrast, *Mona*'s consideration of the principles is over-zealous. Not only does it avoid depending on the work of others, it fails to heed additional information when provided. This limitation in *Mona*'s user-support is intentional. Had *Mona* provided both guidance-dependent and guidance-free schemes (as the principles' state should be done), distinguishing the impact of *Mona*'s guidance-free schemes would be more complex.

The consequence of *Mona*'s intentional limitations, and of its success in providing “free” support, is the stimulation of several directions for further work:

Guidance-dependent facilities. In totally avoiding dependence on user-effort, *Mona* fails to provide additional user-support when colleagues *are* willing to work on behalf of others. Groupware, such as *Mona*, should allow cooperation (and altruism) in the provision of guidance, as demonstrated by normal guidance-dependent groupware. Although entirely guidance-dependent schemes leave systems vulnerable to groupware's vicious circle (chapter 3), combined approaches enable optimum user-support. Such combined schemes would use explicit guidance when available, but would resort to guidance-free techniques when absent (see figures 4.2a, b, and c). Incorporating combined schemes into *Mona* would raise interesting research issues, including the following: when to use explicit guidance; how to balance the use of free and explicit guidance; and the reliability of explicit guidance (accounting for its incorrect and malicious use).

Further “free” support. *Mona* uses fixed deterministic heuristics to infer conversational context. The advantage of deterministic rules is that, ideally, they provide unchanging, and consequently, predictable performance.

User support can also be provided “for free” by inferring what the user will do or prefers based on statistical measures (group or personal) of past activity. Over prolonged use, statistical systems might offer more accurate inferences than those of a deterministic system such

as *Mona*. However, changes in system behaviour encountered while user profiles are being established may frustrate and confuse users (Woods, 1993; Shneiderman, 1993), causing system rejection *before* it achieves group benefits.

Statistical inferences could be used to extend *Mona*'s "free" facilities, for instance:

- *Prioritization of incoming mail*—information on the use of *Mona*'s search template could be used to infer that frequent search patterns are of particular interest to the user. For example, the fact a user frequently searches for the keywords CSCW and groupware could cause messages containing those keywords to be assigned a high priority.
- *Uses of interaction pace*—statistical information on the interaction pace with particular individuals or groups might prove valuable, for example: prompting overdue replies; supplying further information for inferring conversational context; and modifying the interface to reflect current interaction requirements (Dix, 1992b; Dix, 1992c).

For a further discussion of systems using statistical information based on user behaviour, see (Monk, 1989; Darragh & Witten, 1992; Carroll, 1993).

Rule customisation. *Mona*'s rules are currently fixed in C code. This is an unnecessary constraint on those who wish to tailor the conversation-inferencing rules. *Mona*'s default rules and heuristics would be better represented in a predicate-logic language, such as PROLOG, and users should be allowed to alter and extend them. Towards this end *Mona* can produce the properties of any email item as PROLOG predicates.

7.6 TELEFREEK: Integrating collaboration support

Chapter 6 focused on TELEFREEK: a platform for accessing a heterogeneous collection of computing facilities for communication and collaboration. TELEFREEK pools access to an extensible and customisable set of facilities, that include the following: general awareness facilities for social browsing; semi-selective awareness of particular user groups; directed encounter assistance for finding particular individuals; information sources; communication media; and other computer-supported tools.

Directly motivated by the fourth groupware design principle, TELEFREEK's design also followed the other three principles. TELEFREEK's use of widely available low-bandwidth tech-

nology was contrasted to previous groupware for social presence and integrated environments which, almost unanimously, constrains users to synchronous interaction over video channels.

The chapter also recorded a set of human factors that affect collaboration, particularly focusing on the problems of initiating collaboration. They were stimulated by insights arising from the development of TELEFREEK, and by observations on the use of related groupware.

7.6.1 Further work with TELEFREEK

Like *Mona*, TELEFREEK is characterised as much by what it *almost* achieves as by what it *actually* does. As an integrated platform for communication, TELEFREEK is necessarily incomplete: every new communication mechanism, or information source might be incorporated. To ease this problem of completeness, and to reflect the differences between individuals' communication desires, TELEFREEK is highly user-customisable. Regardless of its capability for user-defined extensions, TELEFREEK, and integrated platforms in general, offer potential for further work. Many of the directions for further work are pinpointed or derived from the collaboration human factors described in chapter 6. Opportunities for further work include the following:

Assistance in channel switching. During TELEFREEK's development, facilities for active advice on appropriate communication channels were considered and demonstrated in the HyperCard mock-up, figure 6.1). Active communication advice would allow TELEFREEK to recommend (and perhaps disable) particular communication mechanisms when a user-name is selected, or when incoming messages are received. For example, on receiving email from someone who is no longer logged on, the use of a Unix `talk` facility is guaranteed to fail, and could be disabled.

Research on facilities enabling active channel switching advice would be valuable. Systems could make a variety of inferences based on their access to social presence information of colleagues, their activity level (inferred from idle-time information accessible through Unix commands such as `rwho`), and so on.³

³Active advice was excluded from TELEFREEK's functionality for two reasons: first, to vigorously demonstrate the "minimise constraints" principle, TELEFREEK facilitates the user's decision by providing relevant information, rather than constraining the user's decision on appropriate methods. Second, at TELEFREEK's current installation site, the `rwho` server is disabled.

Assistance in channel switching might also be provided by having the system adapt its interface (or the channels used) according to the pace of interaction. For a discussion of related ideas, see (Dourish, 1992a; Dix, 1992c; Dix, 1992b).

Extended integration. TELEFREEK currently enhances integration by pooling access to communication related utilities. It also provides limited integration between resources: for example, the community information command (a variant of `rwho`) is integrated with the user-enquiry command (`finger`) through the community list, and a person selected from the community list becomes the default for further communication or inquiry. However, TELEFREEK's integration between utilities could be improved, for example: when receiving email, a variety of information about the sender could be autonomously added to relevant files such as `addresses`, `phone-book`, and the Unix `.mailias` file.

Social presence across the Internet. TELEFREEK's social presence information is limited to local-area-networks. Although valuable for directed encounters (for instance the "ambush" facility), the majority of social browsing support is of limited use within local sites: those who want social contact can browse *real* corridors; why provide a simulation?

Joe in Stirling (Scotland) wishes to keep in contact with colleagues in Calgary (Canada). He sends a message to Mary at Calgary asking if she will let him run some commands in her account.

Mary, having agreed, informs her version of TELEFREEK that she trusts Joe to run a specific set of commands (perhaps to list the local community, to ambush individuals, and so on), and mails the relevant commands to Joe.

Joe installs the commands in his TELEFREEK system so that, for instance, on requesting who is about at Calgary, a command is sent by email to Mary. On arrival at Mary's account, a `nohup` (uninterruptible, see (Bourne, 1983)) email scanner extracts the message (identified by the header-field `X-telefreek: request`). If the message is from a trusted colleague (which Joe is), and the command is within Joe's permitted set of operations, it is executed, and the results are returned by email, with an identifying header-field `X-telefreek: response`.

When the message arrives in Joe's mailbox, his TELEFREEK email scanner extracts the message and displays the requested information.

Table 7.1: Outline of a scheme for extending TELEFREEK to the Internet community.

Planned further work on TELEFREEK will extend its social presence facilities (browsing and directed encounters) to the Internet. Implementation of these extensions has not yet begun, but an outline of the proposed mechanism is presented in table 7.1.

The ability to run commands in other people's accounts raises many issues of security and trust, and provides further potential for research.⁴ The *imail* prototype (Hogg, 1985) describes related work in which active-object messages are executed in recipients accounts, and based on the information (user-supplied) collected they either re-distribute themselves or return to the original sender.

Sharing buttons and facilities. TELEFREEK should support users in sharing popular facilities and utilities. Through email, customised button utilities could be shared between users, automating much of the button installation process. For a discussion of similar facilities, see (MacLean *et al.*, 1990).

7.7 General Conclusions

The thesis has detailed four principles for groupware design. Two groupware applications, *Mona* and TELEFREEK, serve as demonstrations of these principles.

The principles attend to current and pragmatic issues in groupware design. They concentrate on the needs of computer scientists and groupware implementors. Within this domain, the principles are both *necessary* and *comprehensive*. Furthermore, they are *beneficial*.

They are *necessary* because they identify the properties causing groupware's failure, and offer alternatives that overcome and avoid these problems. They are *comprehensive* because they cover the full range of system properties. They are *beneficial*, particularly during the initial stages of groupware use, because they emphasise the need for supplying discernible benefits to *all* system users.

⁴For a discussion of trust between interacting parties, see (Marsh, 1992).

References

- Ackerman, MS, & Malone, TW. 1990. Answer Garden: A tool for growing organizational memory. *Pages 31–39 of: Lochovsky, FH, & Allen, RB (eds), Proceedings of the Conference on Office Information Systems*. April 25-27 1990. Cambridge Mass.
- ACM Press. 1988. *Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26–28 1988. Portland, Oregon. ACM Press.
- ACM Press. 1990. *Proceedings of the Third Conference on Computer Supported Cooperative Work* October 7–10 1990. Los Angeles. ACM Press.
- ACM Press. 1992. *Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work* October 31 to November 4 1992. Toronto, Canada. ACM Press.
- Action-Technologies, Selected Software Products. 1987. *What is the Coordinator Software*. Action-Technologies, 2200 Powell St, Suite 1100, Emeryville, CA 94608.
- Ahuja, SR, Ensor, JR, & Lucco, S.E. 1988. A Comparison of Application Sharing Mechanisms in Real- Time Desktop Conferencing Systems. *Pages 238–248 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26–28 1988. Portland, Oregon. ACM Press.
- Apple Computer, Inc. 1987. *Apple Macintosh HyperCard user's guide*. Apple Computer, Inc.
- Apple Computer, Inc. 1988. *Human Interface Guidelines: The Apple Desktop Interface*. Addison-Wesley.
- Auramaki, E, Lehtinen, E, & Lytinen, L. 1988. A Speech-Act-Based Office Modelling Approach. *ACM Transactions on Office Information Systems*, 6(2).
- Austin, JL. 1962. *How to do things with words*. Oxford: Clarendon Press.
- Baecker, RM, & Buxton, WAS (eds). 1987. *Readings in Human-Computer Interaction: A Multidisciplinary Approach*. Morgan-Kaufmann.
- Bair, JH. 1989. Supporting Cooperative Work Teams with Computers: Addressing Meeting Mania. *Pages 208–217 of: Proceedings of the 34th IEEE Computer Society International Conference - CompCon Spring. San Francisco, CA. February 27 - March 3. 1989*.
- Bannon, LJ, & Schmidt, K. 1989. CSCW: Four Characters in Search of a Context. *Pages 358–372 of: Proceedings of the First European Conference on CSCW. (E-CSCW '89)*. London. September.

- Bannon, LJ, Robinson, M, & Schmidt, K (eds). 1991. *Proceedings of the 1991 European Community Conference on Computer Supported Cooperative Work* September 24–27 1991. Amsterdam. Kluwer Academic.
- Bauersfeld, P, Bennett, J, & Lynch, G (eds). 1992. *Proceedings of CHI'92 Conference on Human Factors in Computing Systems* Monterey, May 3–7 1992. Addison-Wesley.
- Belew, RK, & Rentzepis, J. 1990. HyperMail: Treating Electronic Mail as Literature. *Pages 48–54 of: Lochovsky, FH, & Allen, RB (eds), Proceedings of the Conference on Office Information Systems*. April 25–27 1990. Cambridge Mass.
- Benford, S, Prinz, W, Mariani, J, Navarro, L, Bignoli, E, Brown, CG, & Naslund, T. 1992. *MOCCA - A CSCW environment*. Part of the European CO-TECH programme. Steven Benford: Department of Computer Science, Nottingham University. UK.
- Bikson, TK, & Eveland, JD. 1989. Technology Transfer as a Framework for Understanding Social Impacts of Computerisation. *Pages 28–37 of: Smith, MJ, & Salvendy, G (eds), Work with Computers: Organizational, Management, Stress and Health Aspects*. Elsevier Science Publishers B.V., Amsterdam.
- Bloomer, S, & Ingram, F. 1992. User interface design and multimedia. *Pages 54–61 of: Rees, MJ, & Iannella, R (eds), Interface Technology: Advancing Human-Computer Communication. Proceedings of OZCHI 1992. CHISIG Annual Conference*. Bond University Australia, Queensland. November 26–27.
- Bly, SA, Harrison, SR, & Irwin, S. 1993. Media spaces: video, audio, and computing. *Communications of the ACM*, **36**(1), 28–47.
- Borenstein, NS, & Thyberg, CA. 1988. Cooperative Work in the Andrew Message System. *Pages 306–322 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26–28 1988. Portland, Oregon. ACM Press.
- Borenstein, NS, & Thyberg, CA. 1991. Power, ease of use and cooperative work in a practical multimedia message system. *International Journal of Man-Machine Studies*, **32**(2), 229–259.
- Bourne, SR. 1983. *The UNIX system*. Addison-Wesley.
- Bowers, J. 1992. Editorial. *Computer Supported Cooperative Work: An international journal*, **1**(1–2), 1–5.
- Branskat, S. 1991 (November). *How electronic mail is used: Implications for the design of e-mail systems*. M.Sc. Project Report: CSCW Class, Department of Computer Science, University of Calgary. Alberta.
- Brobst, SA, Malone, TW, Grant, KR, & Cohen, MD. 1986. Toward Intelligent Message Routing Systems. *Pages 351–359 of: Uhlig, R (ed), Computer Message Systems '85. Proceedings of the 2nd International Symposium on Computer Message Systems*. Elsevier Science Publishers B.V. (North-Holland).

- Bullen, CV, & Bennet, JL. 1990. Learning from User Experience with Groupware. *Pages 291-302 of: Proceedings of the Third Conference on Computer Supported Cooperative Work* October 7-10 1990. Los Angeles.
- Bush, V. 1945. As We May Think. *The Atlantic Monthly*, 176(1), 101-108.
- Carasik, RP, & Grantham, CE. 1988. A Case Study of CSCW in a Dispersed Organisation. *Pages 61-65 of: Proceedings of CHI'88 Conference on Human Factors in Computing Systems*.
- Carroll, J. 1990. *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*. MIT Press.
- Carroll, JM. 1987. The adventure of getting to know a computer. *In: Baecker, RM, & Buxton, WAS (eds), Readings in Human-Computer Interaction: A Multidisciplinary Approach*. Morgan-Kaufmann.
- Carroll, JM. 1993. Creating a design science of human-computer interaction. *Interacting with computers*, 3-12.
- Carroll, JM, Smith-Kerker, PL, Ford, JR, & Mazur-Rimetz, SA. 1987/1988. The minimal manual. *Human Computer Interaction*, 3, 123-153.
- Cashman, PM, & Holt, WH. 1980. A Communication Oriented Approach to Structuring the Software Maintenance Environment. *ACM SIGSOFT Software Engineering Notes*, 5(1), 4-17.
- Catlin, T, Bush, P, & Yankelovich, N. 1989. InterNote: extending a hypermedia framework to support annotative col laboration. *Pages 365-378 of: Hypertext '89 Proceedings*. Pittsburgh, Pennsylvania November 1989. ACM Press.
- CCITT. 1987. *Draft recommendation on message handling systems, X.400, Version 5*.
- Chapanis, A. 1975. Interactive computer communication. *Scientific American*, 232(3), 36-92.
- Chilton, P. 1989. *Introducing X.400*. NCC Publications.
- Clement, A, & Gotlieb, CC. 1987. Evolution of an Organisational Interface: The New Business Department at a Large Insurance Firm. *ACM Transactions on Office Information Systems*, 5(4).
- Cockburn, A. 1991a. *File coordination in HyperCommitments*. Draft paper, available from the author. Department of Computing Science, University of Stirling, Scotland.
- Cockburn, AJG. 1991b. *The mnd manual pages*. Department of Computing Science, University of Stirling, Scotland. FK9 4LA. Appendix A of PhD thesis (1993) entitled "Groupware design: principles, prototypes, and systems.
- Cockburn, AJG. 1991c. Reflexive CSCW and Managing Commitments. *Pages 9/1-9/5 of: Colloquium on CSCW: Some Fundamental Issues*. March 15. 1991, no. 065. IEE.

- Cockburn, AJG. 1992. *A user's guide to Mona: an enhanced email system*. Tech. rept. 97. Department of Computing Science and Mathematics, University of Stirling, Stirling, Scotland, FK9 4LA.
- Cockburn, AJG. 1993. *A user's guide to TELEFREEK: a communication and social awareness environment*. Tech. rept. 105. Department of Computing Science and Mathematics, University of Stirling, Stirling, Scotland, FK9 4LA.
- Cockburn, AJG, & Greenberg, S. 1993. *Making Contact: getting the group communicating*. Tech. rept. 93/498/03. Department of Computer Science, University of Calgary, Alberta T2N 1N4 Canada. Submitted to COOCS '93.
- Cockburn, AJG, & Jones, SRA. 1992. *Four principles for groupware design: encouraging adoption and easing system use*. Paper presented at DTI Seminar on Implementation Perspectives on CSCW Design, To appear: Springer-Verlag.
- Cockburn, AJG, & Thimbleby, HW. 1991. A Reflexive Perspective of CSCW. *ACM SIGCHI Bulletin*, 23(3), 63–68.
- Cockburn, AJG, & Thimbleby, HW. 1992a. Automatic conversational context: Avoiding dependency on user effort in groupware. *Pages 142–149 of: Rees, MJ, & Iannella, R (eds), Interface Technology: Advancing Human-Computer Communication. Proceedings of OZCHI 1992. CHISIG Annual Conference. Bond University Australia, Queensland. November 26–27.*
- Cockburn, AJG, & Thimbleby, HW. 1992b. *Reducing user effort in collaboration support*. Tech. rept. 93. University of Stirling, Stirling, Scotland, FK9 4LA.
- Cockburn, AJG, & Thimbleby, HW. 1993. Reducing user effort in collaboration support. *Pages 215–218 of: Gray, WD, Hefley, WE, & Murray, D (eds), Proceedings of the 1993 International Workshop on Intelligent User Interfaces, Orlando, Florida. January 4–7. New York: ACM Press.*
- Collin, S, Bawa, J, Bull, E, Eager, A, Fletcher, P, Goodwins, R, & Phillips, T. 1992. Email: bringing people together. *PC Magazine*, November, 252–276.
- Conklin, J. 1988. HyperText: An Introduction and Survey. *In: Greif, I (ed), Computer Supported Cooperative Work: A Book of Readings*. Morgan Kaufmann.
- Conklin, J, & Begeman, ML. 1988a. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. *ACM Transactions on Office Information Systems*, 6(4), 303–331.
- Conklin, J, & Begeman, ML. 1988b (March). gIBIS: a hypertext tool for team design deliberation. *Pages 247–251 of: Hypertext '87 University of North Carolina, Chapel Hill, North Carolina, November 13–15 1987.*
- Cook, P, Ellis, C, Graf, M, Rein, G, & Smith, T. 1987. Project Nick: Meetings augmentation and analysis. *ACM Transactions on Office Information Systems*, 5(2), 132–146.
- Cool, C, Fish, RS, Kraut, RE, & Lowery, CM. 1992. Interactive design of video communication systems. *Pages 25–32 of: Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work October 31 to November 4 1992. Toronto, Canada.*

- Cooper, G. 1991. Context and its Representation. *Interacting with Computers: the Interdisciplinary Journal of Human-Computer Interaction*, 3(3), 243–252.
- Coulouris, G, & Thimbleby, H. 1992. *HyperProgramming*. Addison-Wesley.
- Crocker, DH. 1982 (August). *Standard for the format of ARPA Internet Text Messages; RFC-822*. ARPANET Working Group Requests for Comments: RFC-822.
- Crow, D, & Smith, B. 1993. The role of built-in knowledge in adaptive interface systems. *Pages 97–104 of: Gray, WD, Hefley, WE, & Murray, D (eds), Proceedings of the 1993 International Workshop on Intelligent User Interfaces, Orlando, Florida. January 4–7*. New York: ACM Press.
- Crowston, K, & Malone, TW. 1988. Intelligent software agents. *Byte Magazine*, December, 267–271.
- Crowston, K, Malone, TW, & Lin, F. 1987–1988. Cognitive Science and Organizational Design: A Case Study of Computer Conferencing. *Interacting with Computers: the Interdisciplinary Journal of Human-Computer Interaction*, 3, 59–85.
- Darragh, J, & Witten, I. 1992. *The Reactive Keyboard*. Cambridge University Press.
- De Cindio, F, De Michelis, G, Simone, C, Vassallo, R, & Zanaboni, AM. 1986. Chaos as coordination technology. *Pages 325–342 of: Proceedings of the first conference on Computer-Supported Cooperative Work*, Austin, Texas, December 1986.
- Denning, PJ. 1982. Electronic Junk. *Communications of the ACM*, 25(3), 163–165.
- Descartes, R. 1927. *A discourse on method: translated by Veitch, J; edited by Rhys, E*. Everyman's Library, no. 570. Dent, JM and Sons Ltd.
- Dix, A. 1992a. *Cooperation without Communication: the problems of highly distributed working*. Human-Computer Interaction Group, University of York, England. YO1 5DD.
- Dix, A, Finlay, J, Abowd, G, & Beale, R. 1993. *Human-Computer Interaction*. Prentice Hall.
- Dix, AJ. 1992b. Beyond the Interface. *Pages 171–190 of: Engineering for Human-Computer Interaction: Proceedings of IFIP TC2/WG2.7 Working Conference, Ellovuori, Finland, 10–14 August*.
- Dix, AJ. 1992c. Pace and interaction. *Pages 171–190 of: Proceedings of HCI '92: People and computers VII*.
- Dix, AJ, & Beale, R. 1992. *Information requirements of distributed workers*. University of York, Heslington, York. YO1 5DD. Pending publication.
- Dix, AJ, & Miles, VC. 1992. *Version control for asynchronous group work*. Tech. rept. YCS 181. University of York, Heslington, York. YO1 5DD. Poster presentation at HCI'92: People and Computers VII.
- Dollimore, J, & Wilbur, S. 1991. Experiences in building a configurable CSCW system. *Pages 173–181 of: Bowers, JM, & Benford, SD (eds), Studies in Computer Supported Cooperative Work: Theory, Practice and Design*. North-Holland.

- Dourish, P. 1992a. *Computational Reflection and CSCW Design*. Rank Xerox EuroPARC, Cambridge, UK.
- Dourish, P. 1992b. *Implementation Perspectives on CSCW Design. Call for papers. DTI CSCW SIG Workshop, 13 October 1992*.
- Dourish, P, & Bly, S. 1992. Portholes: Supporting awareness in a distributed work group. *Pages 541-547 of: Proceedings of CHI'92 Conference on Human Factors in Computing Systems* Monterey, May 3-7 1992. Addison-Wesley.
- Draper, SW, & Oatley, K. 1990. Action centered manuals or minimalist instruction? Alternative theories for Carroll's minimal manuals. *Pages 222-243 of: Holt, P, & Williams, N (eds), Computers and Writing: State of the Art*. Blackwells.
- Dubs, S, & Hayne, SC. 1992. Distributed facilitation: A concept whose time has come? *Pages 314-321 of: Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work* October 31 to November 4 1992. Toronto, Canada.
- Dykstra, EA, & Carasik, RP. 1991. Structure and Support in Cooperative Environments: the Amsterdam Conversation Environment. *International Journal of Man-Machine Studies*, **34**, 419-434.
- EC-CSCW. 1989. *Proceedings of the 1989 European Community Conference on Computer Supported Cooperative Work* September 13-15 1989. Gatwick, London, UK. Proceedings from The Computer Sciences Company, Computer Sciences House, Brunel, Slough, SL1 1XL, UK.
- EC-CSCW. 1993. *Proceedings of the 1993 European Community Conference on Computer Supported Cooperative Work* September 13-17 1993. Milan, Italy. To appear.
- Egido, C. 1988. Videoconferencing as a technology to support groupwork: A review of its failure. *Pages 13-24 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26-28 1988. Portland, Oregon.
- Ehrlich, SF. 1987. Strategies for Encouraging Successful Adoption of Office Communication Systems. *ACM Transactions on Office Information Systems*, **5**(4), 340-357.
- Ellis, CA, Gibbs, SJ, & Rein, GL. 1991. Groupware: Some Issues and Experiences. *Communications of the ACM*, **34**(1), 38-58.
- Elrod, S, Bruce, R, Gold, R, Goldberg, D, Halasz, F, Janssen, W, Lee, D, McCall, K, Pederson, E, Pier, K, Tang, J, & Welch, B. 1992. Liveboard: A large interactive display supporting group meetings, presentations and remote collaboration. *Pages 599-607 of: Proceedings of CHI'92 Conference on Human Factors in Computing Systems* Monterey, May 3-7 1992. Addison-Wesley.
- Elwart-Keys, M, Halonen, D, Horton, M, Kass, R, & Scott, P. 1990. User Interface Requirements for Face to Face Groupware. *Pages 295-301 of: Proceedings of CHI'90 Conference on Human Factors in Computing Systems* Seattle, April 1990. ACM Press.
- Engelbart, D, & Lehtman, H. 1988. Working together. *Byte Magazine*, December, 245-252.

- Engelbart, DC. 1963. A Conceptual Framework for the Augmentation of Man's Intellect. *Pages 1-29 of: Howerton, P (ed), Vistas in Information handling.* Spartan Books, Washington.
- Erickson, TD. 1989. Interfaces for Cooperative Work: An Eclectic Look at CSCW 88. *SIGCHI bulletin*, 21(1), 56-64.
- Eveland, JD, & Bikson, TK. 1988. Work Group Structures and Computer Support: A Field Experiment. *Pages 324-343 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26-28 1988. Portland, Oregon. ACM Press.
- Fafchamps, D, Reynolds, D, & Kuchinsky, A. 1991. The dynamics of small group decision-making using electronic mail. *Pages 211-224 of: Bowers, JM, & Benford, SD (eds), Studies in Computer Supported Cooperative Work: Theory, Practice and Design.* North-Holland.
- Fish, RS, Kraut, RE, Root, RW, & Rice, RE. 1992. Evaluating Video as a Technology for Informal Communication. *Pages 37-48 of: Proceedings of CHI'92 Conference on Human Factors in Computing Systems* Monterey, May 3-7 1992. Addison-Wesley.
- Fish, RS, Kraut, RE, Root, RW, & Rice, RE. 1993. Video as a technology for informal communication. *Communications of the ACM*, 36(1), 48-61.
- Flores, F, Graves, M, Hartfield, B, & Winograd, T. 1988. Computer Systems and the Design of Organisational Interaction. *ACM Transactions on Office Information Systems*, 6(2), 153-172.
- Foltz, PW. 1990. Using Latent Semantic Indexing for Information Filtering. *Pages 40-47 of: Lochovsky, FH, & Allen, RB (eds), Proceedings of the Conference on Office Information Systems.* April 25-27 1990. Cambridge Mass.
- Foltz, PW, & Dumais, ST. 1993. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12), 51-60.
- Francik, E, Rudman, SE, Cooper, D, & Levine, S. 1991. Putting Innovation to Work: Adoption Strategies for Multimedia Communication Systems. *Communications of the ACM*, 34(12), 53-63.
- Furnas, GW. 1985. Experience with an Adaptive Indexing Scheme. *Pages 131-136 of: Human Factors in Computing Systems II. Proceedings of the CHI'85 conference.* Amsterdam; North-Holland/ACM.
- Gale, S. 1991. Adding Audio and Video to an Office Environment. *Pages 49-62 of: Bowers, JM, & Benford, SD (eds), Studies in Computer Supported Cooperative Work: Theory, Practice and Design.* North-Holland.
- Gaver, W, Moran, T, MacLean, A, Lovstrand, L, Dourish, P, Carter, K, & Buxton, W. 1992. Realizing a video environment: EuroPARC's RAVE system. *Pages 27-35 of: Proceedings of CHI'92 Conference on Human Factors in Computing Systems* Monterey, May 3-7 1992. Addison-Wesley.

- Gibbs, SJ. 1989. LIZA: An Extensible Groupware Toolkit. *Pages 29–35 of: Proceedings of CHI'89 Conference on Human Factors in Computing Systems* Austin Texas, April 1989.
- Goodman, GO, & Abel, MJ. 1987. Communication and Collaboration: Facilitating Cooperative Work Through Communication. *Office: Technology and People*, 3(2), 129–146.
- Gordon, M, Belew, R, Kochen, M, & Lindsay, R. 1985. Managing Computerised Conferences. *Pages 179–184 of: Proceedings of the National Online Meeting*.
- Gould, JD, & Lewis, C. 1985. Designing for Usability: Key Principles and What Designers Think. *Communications of the ACM*, 28(3), 300–309.
- Greenbaum, J. 1988. In Search of Cooperation. An Historical Analysis of Work Organisation and Management Studies. *Pages 102–114 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26–28 1988. Portland, Oregon.
- Greenberg, S. 1990a. *Personalizable Groupware: Accommodating Individual Roles and Group Differences*. Tech. rept. 90/404/28. Alberta Research Council.
- Greenberg, S. 1990b. Sharing Views and Interactions with Single-User Applications. *Pages 227–237 of: Lochovsky, FH, & Allen, RB (eds), Proceedings of the Conference on Office Information Systems*. April 25-27 1990. Cambridge Mass.
- Greenberg, S. 1991a. An annotated bibliography of computer supported cooperative work. *Pages 359–413 of: Greenberg, S (ed), Computer-Supported Cooperative Work and Groupware*. London: Academic Press.
- Greenberg, S (ed). 1991b. *Computer supported cooperative work and groupware*. Computer and People Series, Academic Press, London.
- Greenberg, S, & Bohnet, R. 1991. Group Sketch: A Multi-User Sketchpad for Graphically-Distributed Small Groups. *In: Proceedings of Graphics Interface '91*. June 5-7. Calgary, Canada.
- Greenberg, S, Bohnet, R, Roseman, M, & Webster, D. 1992 (November). *GroupSketch*. SIG-GRAPH Video Review. (supplement of the ACM Transactions on Graphical Systems), 87.
- Grehan, R, Eglowstein, H, Thompson, T, & Yager, T. 1991. Getting Groups on Schedule. *Byte Magazine*. September 1991, 250–264.
- Greif, I (ed). 1988a. *Computer Supported Cooperative Work: A Book of Readings*. Morgan Kaufmann Publishers, San Mateo, California.
- Greif, I. 1988b. *PANEL Session CSCW: Breakthroughs for User Acceptance*. *Pages 113–144 of: Proceedings of CHI'88 Conference on Human Factors in Computing Systems*.
- Greif, I, & Sarin, S. 1987. Data Sharing in Group Work. *ACM Transactions on Office Information Systems*, 5(2), 187–211.
- Grudin, J. 1988. Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces. *Pages 85–93 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26–28 1988. Portland, Oregon.

- Grudin, J. 1989. Why Groupware Applications Fail: Problems in Design and Evolution. *Office Technology and People*, 4(3), 245–264.
- Harper, RHR, Lamming, MG, & Newman, WM. 1992. Locating systems at work: implications of the development of active badge applications. *Interacting with Computers: the Interdisciplinary Journal of Human-Computer Interaction*, 4(3), 343–363.
- Hashim, SH. 1991. WHAT: An argumentative groupware approach for organizing and documenting research activities. *Journal of Organizational Computing*, 1(3), 275–302.
- Henninger, S. 1991. Computer Systems Supporting Cooperative Work: A CSCW '90 Trip Report. *ACM SIGCHI Bulletin. Special Issue on CSCW*, 23(3), 25–28.
- Hill, RD, Brinck, T, Patterson, JF, Rohall, SL, & Wilner, WT. 1993. The Rendezvous Language and Architecture. *Communications of the ACM*, 36(1), 62–67.
- Hiltz, HR, & Turoff, M. 1981. The Evolution of User Behaviour in a Computerized Conferencing System. *Communications of the ACM*, 24(11), 739–751.
- Hiltz, SR. 1984. *Online Communities - A Case Study of the Office of the Future*. Ablex Publishing Corporation.
- Hiltz, SR, & Turoff, M. 1985. Structuring Computer-Mediated Communication Systems to Avoid Information Overload. *Communications of the ACM*, 27(7), 680–689.
- Hirschheim, RA. 1986. Understanding the Office: A Social-Analytical Perspective. *ACM Transactions on Office Information Systems*, 4(4), 331–344.
- Hogg, J. 1985. Intelligent message systems. *Pages 113–133 of: Tsichritzis, D (ed), Office Automation*. Springer Verlag.
- Hollan, J, & Stornetta, S. 1992. Beyond being there. *Pages 119–125 of: Proceedings of CHI'92 Conference on Human Factors in Computing Systems Monterey, May 3–7 1992*. Addison-Wesley.
- Holleran, PA, & Haller, RW. 1990. Characteristics of a Well-Designed Electronic Communications System. *Pages 841–847 of: Human-Computer Interaction. INTERACT '90*. Elsevier Science Publishers B.V. (North-Holland).
- Iannella, R. 1992. Directory systems interfaces. *Pages 110–117 of: Rees, MJ, & Iannella, R (eds), Interface Technology: Advancing Human-Computer Communication. Proceedings of OZCHI 1992. CHISIG Annual Conference*. Bond University Australia, Queensland. November 26–27.
- Ishii, H, & Kobayashi, M. 1992. ClearBoard: A seamless medium for shared drawing and conversation with eye contact. *Pages 525–532 of: Proceedings of CHI'92 Conference on Human Factors in Computing Systems Monterey, May 3–7 1992*. Addison-Wesley.
- Ishii, H, & Miyake, N. 1991. Toward an Open Shared Workspace: Computer and Video Fusion Approach of TeamWorkStation. *Communications of the ACM*, 34(12), 37–50.

- Ishii, H, Kobayashi, M, & Grudin, J. 1992. Integration of inter-personal space and shared workspace: ClearBoard design and experiments. *Pages 33-42 of: Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work* October 31 to November 4 1992. Toronto, Canada.
- IST, Ltd. 1991 (January). *X-Designer User Manual*. XD/02 Issue 2 edn. Imperial Software Technology Ltd., 95 London Street, Reading, Berkshire. RG1 4QA. UK.
- Johansen, R. 1988. *Groupware: Computer Support for Business Teams*. Free Press, New York.
- Johnson-Lentz, P, & Johnson-Lentz, T. 1982. Groupware: the process and impacts of design choices. *In: Kerr, EB, & Hiltz, SR (eds), Computers and Writing: State of the Art*. Academic Press, New York.
- Jones, S. 1990 (September). *A discussion of issues and systems relevant to computer supported cooperative work*. Tech. rept. 64. Department of Computing Science and Mathematics, University of Stirling, Stirling, Scotland. T.R. 64.
- Jones, S. 1992a. MILO: a computer based tool for (co)authoring structured documents. *In: Sharples, M (ed), Computer Supported Collaborative Writing*. Springer-Verlag.
- Jones, S. 1993. *Easing the writing task: designing computer based systems to help authors. To be submitted 1993*. Ph.D. thesis, Department of Computing Science and Mathematics, University of Stirling, Scotland. FK9 4LA.
- Jones, SR. 1992b. *A user's guide to MILO: a computer based system to support (co) authors of structured documents*. Tech. rept. 92. University of Stirling, Stirling, Scotland, FK9 4LA.
- Jones, SRA, & Cockburn, AJG. 1993. *Reducing requirements in computer supported writing tasks*. Paper presented at: 6th UK conference on computers and writing. University of Wales. April 13-15.
- Jones, WP. 1986. The Memory Extender Personal Filing System. *Pages 298-305 of: Human Factors in Computing Systems III. Proceedings of the CHI'86 conference*. Amsterdam; North Holland/ACM.
- Jung, CG. 1952. Synchronicity: an acausal connecting principle. *In: The collected works of C.G. Jung. Part 8*. Princeton University Press.
- Kaplan, SJ, Kapor, MD, Belove, EJ, Landsman, RA, & Drake, TR. 1990. Agenda: A Personal Information Manager. *Communications of the ACM*, **33**(7), 105-116.
- Kaye, AR, & Karam, GM. 1987. Cooperating Knowledge-Based Assistants for the Office. *ACM Transactions on Office Information Systems*, **5**(4), 297-326.
- Keeler, MA, & Denning, SM. 1991. The Challenge of Interface Design for Communication Theory: from interaction metaphor to contexts of discovery. *Interacting with Computers: the Interdisciplinary Journal of Human-Computer Interaction*, **3**(3), 283-301.

- Kiesler, S, Zubrow, D, Moses, AM, & Geller, V. 1985. Affect in Computer-Mediated Communication: An Experiment in Synchronous Terminal to Terminal Discussion. *Human Computer Interaction*, 1, 77-104.
- Kiesler, S, Siegel, J, & McGuire, TW. 1988. The Variable Impact of Computer Technologies on the Organization of Work Activities. *Pages 657-682 of: Greif, I (ed), Computer Supported Cooperative Work: A Book of Readings*. Morgan Kaufmann.
- Kille, SE. 1991. *Implementing X.400 and X.500: the PP and QUIPU systems*. Artech House; Boston.
- Kincaid, CM, Dupont, PB, & Kaye, AR. 1985. Electric Calendars in the Office: An Assessment of User Needs and Current Technology. *ACM Transactions on Office Information Systems*, 3(1), 89-102.
- Koo, CC. 1988. A commitment-based communication model for distributed office environments. *Pages 291-298 of: Proceedings of the 1988 Conference on Office Information systems, March 23-25. Palo Alto, CA*. New York: ACM Press.
- Kozierok, R, & Maes, P. 1993. A Learning Interface Agent for Scheduling Meetings. *Pages 81-88 of: Gray, WD, Hefley, WE, & Murray, D (eds), Proceedings of the 1993 International Workshop on Intelligent User Interfaces, Orlando, Florida. January 4-7*. New York: ACM Press.
- Kraemer, KL, & King, JL. 1988. Computer-based systems for cooperative work and group decision making. *ACM Computing Surveys*, 20(2), 115-146.
- Krasner, H, McInroy, J, & Walz, DB. 1991. Groupware Research and Technology Issues with Application to Software Process Management. *IEEE Transactions on Systems, Man and Cybernetics*, 21(4), 704-712.
- Kraut, R, Egido, C, & Galegher, J. 1988a. Patterns of Contact and Communication in Scientific Research and Collaboration. *Pages 1-12 of: Proceedings of the Second Conference on Computer Supported Cooperative Work September 26-28 1988*. Portland, Oregon.
- Kraut, R, Galegher, J, & Egido, C. 1988b. Relationships and Tasks in Scientific Research Collaborations. *Pages 741-769 of: Greif, I (ed), Computer Supported Cooperative Work: A Book of Readings*. Morgan Kaufmann.
- Kraut, RE, & Dumais, S. 1990. Computerization and the quality of working life: The role of control. *Pages 56-67 of: Lochovsky, FH, & Allen, RB (eds), Proceedings of the Conference on Office Information Systems. April 25-27 1990*. Cambridge Mass.
- Kraut, RE, & Streeter, LA. 1990. Satisfying the Need to Know: Interpersonal Information Access. *Pages 909-915 of: Human-Computer Interaction. INTERACT '90*. Elsevier Science Publishers B.V. (North-Holland).
- Kunz, W, & Rittel, H. 1970. *Issues as Elements of information systems*. Tech. rept. Working Paper Number 131. Institute of Urban and Regional Development, University of California, Berkeley. California.

- Kyng, M. 1991. Designing for cooperation: Cooperating in design. *Communications of the ACM*, 34(12), 65–73.
- Lamming, MG, & Newman, WM. 1991. *Activity-based Information Retrieval: Technology in Support of Human Memory*. Tech. rept. 91-03. Rank Xerox EuroPARC, Cambridge. UK.
- Lamport, L. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7), 558–565.
- Lansdale, M, & Edmonds, E. 1992. Using Memory for Events in the Design of Personal Filing Systems. *International Journal of Man-Machine Studies*, 36, 97–126.
- Lansdale, MW, Young, DR, & Bass, CA. 1989. MEMOIRS: A Personal Multimedia Information System. *Pages 845–852 of: Salvendy, G, & Smith, MJ (eds), Designing and using human-computer interfaces and knowledge based systems*. Elsevier Amsterdam.
- Lauwers, JC, Joseph, TA, Lantz, KA, & Romanow, AL. 1988. Replicated architectures for shared window systems: A critique. *Pages 249–260 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26–28 1988. Portland, Oregon. ACM Press.
- Leadbetter, M, & Seeley, D. 1992. Extending electronic mail for coordination and groupware. *Pages 150–157 of: Rees, MJ, & Iannella, R (eds), Interface Technology: Advancing Human-Computer Communication. Proceedings of OZCHI 1992. CHISIG Annual Conference*. Bond University Australia, Queensland. November 26–27.
- Lee, J. 1990. SIBYL: A Tool for Managing Group Decision Rationale. *Pages 79–92 of: Proceedings of the Third Conference on Computer Supported Cooperative Work* October 7–10 1990. Los Angeles.
- Lee, J, & Malone, TM. 1990. Partially Shared Views: A scheme for Communicating among Groups that Use Different Type Hierarchies. *ACM Transactions on Office Information Systems*, 8(1), 1–26.
- Leland, MDP, Fish, RS, & Kraut, RE. 1988. Collaborative Document Production Using Quilt. *Pages 206–215 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26–28 1988. Portland, Oregon. ACM Press.
- Lotus. 1989. *Lotus Notes Users Guide*. Lotus Development Corp., Cambridge, MA.
- Lovstrand, L. 1991. Being Selectively Aware with the Khronika System. *Pages 265–277 of: Proceedings of the Second European Conference on Computer Supported Cooperative Work*. Sept 25-27, 1991. Amsterdam.
- Lowe, DG. 1985. Co-operative structuring of information: the representation of reasoning and debate. *International Journal of Man-Machine Studies*, 23(9), 97–111.
- Lutz, E, Kleist-Retzow, H, & Hoernig, K. 1990. MAFIA – An active mail-filter-agent for an intelligent document processing support. *ACM SIGOIS Bulletin*, 11(4), 16–32.
- Mackay, WE. 1988. More Than Just a Communication System: Diversity in the Use of Electronic Mail. *Pages 344–353 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26–28 1988. Portland, Oregon.

- Mackay, WE, Malone, TW, Crowston, K, Rao, R, Rosenblitt, D, & Card, SK. 1989. How Do Experienced Information Lens Users Use Rules? *Pages 211-216 of: Proceedings of CHI'89 Conference on Human Factors in Computing Systems* Austin Texas, April 1989.
- MacLean, A, Carter, K, Lovstrand, L, & Moran, T. 1990. User-tailorable systems: pressing the issue with buttons. *Pages 175-182 of: Proceedings of CHI'90 Conference on Human Factors in Computing Systems* Seattle, April 1990. ACM Press.
- Malone, TW. 1983. How Do People Organise Their Desks? Implications for the Design of Office Information Systems. *ACM Transactions on Office Information Systems*, 1(1), 99-112.
- Malone, TW, & Lai, KY. 1988. Object Lens: A "Spreadsheet" for Cooperative Work. *Pages 115-124 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26-28 1988. Portland, Oregon.
- Malone, TW, Grant, KR, Turbak, FA, Brobst, SA, & Cohen, MD. 1987. Intelligent Information-Sharing Systems. *Communications of the ACM*, 30(5), 390-402.
- Malone, TW, Grant, KR, Lai, K-Y, Rao, R, & Rosenblitt, D. 1988. Semi-structured messages are surprisingly useful for computer-supported coordination. *Pages 311-331 of: Greif, I (ed), Computer Supported Cooperative Work: A Book of Readings*. Morgan Kaufmann.
- Malone, TW, Lai, K-Y, & Fry, C. 1992. Experiments with OVAL: A radically tailorable tool for cooperative work. *Pages 289-297 of: Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work* October 31 to November 4 1992. Toronto, Canada.
- Mander, R, Salomon, G, & Wong, YY. 1992. A 'Pile' metaphor for supporting casual organization of information. *Pages 627-634 of: Proceedings of CHI'92 Conference on Human Factors in Computing Systems* Monterey, May 3-7 1992. Addison-Wesley.
- Mantei, M. 1988. Groupware: Interface Design for Meetings (PANEL). *Pages 161-163 of: Proceedings of CHI'88 Conference on Human Factors in Computing Systems*.
- Mantei, M. 1989. Observation of Executives Using a Computer Supported Meeting Environment. *Decision Support Systems*, 5, 153-166.
- Markus, ML, & Connolly, T. 1990. Why CSCW Applications Fail: Problems in the Adoption of Interdependent Work Tools. *Pages 371-380 of: Proceedings of the Third Conference on Computer Supported Cooperative Work* October 7-10 1990. Los Angeles.
- Marsh, S. 1992. Trust and reliance in multi-agent systems: a preliminary report. In: Cesta, A, Miceli, M, & Conte, R (eds), *The fourth European workshop on modelling autonomous agents in an artificial world, San Martino al Cimino, Italy. July 29-31*. Institute of Psychology of the Italian National Research Council.
- Minneman, SL, & Bly, SA. 1991. Managing a Trois: A Study of a Multi-User Drawing Tool in Distributed Design Work. *Pages 217-223 of: Proceedings of CHI'91 Conference on Human Factors in Computing Systems* New Orleans, May 1991.

- Monk, AF. 1989. The Challenge of Interface Design for Communication Theory: from interaction metaphor to contexts of discovery. *Interacting with Computers: the Interdisciplinary Journal of Human-Computer Interaction*, 1(2), 190–196.
- Muller, MJ, Smith, JG, Shoher, JZ, & Goldberg, H. 1991. The Impact of Organisations on Computers. *SIGCHI Bulletin*, 23(1), 82–85.
- Murphy, AS. 1990 (October). The Multimedia Assistant. In: *IEE Colloquium on CSCW* October 24, 1990 London. Digest No: 1990/132.
- Nagasundaram, M. 1990. Style and Substance in communication: Implications for message structuring systems. *ACM SIGOIS Bulletin*, 11(4), 33–41.
- Neuron Data, Inc. 1992. *Portable graphical user interfaces across all windowing standards: Product Information on the Open Interface system*. 488 NEURON DATA, 156 University Avenue, Palo Alto, CA 94301.
- Newman, J, Reynolds, C, & Wilson, P. 1990 (October). HICOM: An Interactive Scientific Community. In: *IEE Colloquium on CSCW* October 24, 1990 London. Digest No: 1990/132.
- Newman, W, & Wellner, P. 1992. A desk supporting computer-based interaction with paper documents. Pages 587–592 of: *Proceedings of CHI'92 Conference on Human Factors in Computing Systems* Monterey, May 3–7 1992. Addison-Wesley.
- Newman, WM, Eldridge, MA, & Lamming, MG. 1991. PEPYS: Generating Autobiographies by Automatic Tracking. In: Bannon, LJ, Robinson, M, & Schmidt, K (eds), *Proceedings of the 1991 European Community Conference on Computer Supported Cooperative Work* September 24–27 1991. Amsterdam.
- NeXT Computer, Inc. 1990 (August). *Software and Peripherals*. 900 Chesapeake Drive, Redwood City, CA 94063. N6023.
- Nielson, J. 1990a. The Art of Navigating through HyperText. *Communications of the ACM*, 33(3), 296–310.
- Nielson, J. 1990b. Trip Report: HyperText '89. *SIGCHI Bulletin*, 21(4), 52–61.
- Norman, DA. 1983. Design principles for Human-Computer Interfaces. Pages 1–10 of: *Proceedings of CHI 83*. New York: ACM Press.
- Norman, DA. 1987. Some observations on mental models. In: Baecker, RM, & Buxton, WAS (eds), *Readings in Human-Computer Interaction: A Multidisciplinary Approach*. Morgan-Kaufmann.
- Novick, DG, & Walpole, J. 1990. Enhancing the Efficiency of Multiparty Interaction Through Computer Mediation. *Interacting with Computers: The Interdisciplinary Journal of Human-Computer Interaction*, 2(2), 229–246.
- Ogura, A, & Gillespie, T. 1991. The Design and Maintenance of the Andrew Message System. *SIGCHI Bulletin*, 23(1), 44–47.

- Ohukubu, M, & Ishii, H. 1990. Design and Implementation of a Shared Workspace by Integrating Individual Workspaces. *Pages 142-146 of: Lochovsky, FH, & Allen, RB (eds), Proceedings of the Conference on Office Information Systems*. April 25-27 1990. Cambridge Mass.
- Opper, S. 1988. A Groupware Toolbox. *Byte Magazine*, December, 275-282.
- Palme, J. 1984. You Have 134 Unread Mail! Do You Want to Read Them Now? *Pages 175-184 of: Smith, HT (ed), Computer-Based Message Services*. IFIP, 1984. Elsevier Science Publishers B.V. (North-Holland).
- Patterson, JF. 1991. Comparing the demands of single-user and multi-user applications. *Pages 87-94 of: Proceedings of the ACM Symposium on User Interface Software and Technology*. Hilton Head, South Carolina, USA. November 11-13, 1991. ACM Press.
- Pollock, S. 1988. A Rule-Based Message Filtering System. *ACM Transactions on Office Information Systems*, 6(3), 233-254.
- Prinz, W, & Pennelli, P. 1991. Relevance of the X.500 Directory to CSCW Applications — Directory support for computer based communication. *Pages 267-283 of: Bowers, JM, & Benford, SD (eds), Studies in Computer Supported Cooperative Work: Theory, Practice and Design*. Elsevier Science Publishers (North-Holland).
- Quarterman, JS. 1989. *The Matrix: Computer Networks and Conferencing Systems Worldwide*. Digital Press.
- Remde, JR, Gomez, LM, & Landauer, TK. 1988 (March). SuperBook: An Automatic Tool for Information Exploration - HyperText? *In: Hypertext '87*, University of North Carolina, Chapel Hill, North Carolina, November 13-15 1987.
- Rettig, M. 1992. Interface design when you don't know how. *Communications of the ACM*, 35(1), 29-34.
- Rhyne, JR, & Wolf, CG. 1992. Tools for supporting the collaborative process. *Pages 161-170 of: Proceedings of the 1992 ACM Symposium on User Interface Software and Technology*. Monterey, CA. November 15-18. New York: ACM Press.
- Robbins, M. 1990. Software Synergy. *Practical Computing*, May, 68-70.
- Robertson, SP, Olson, GM, & Olson, JS (eds). 1991. *Proceedings of CHI'91 Conference on Human Factors in Computing Systems* New Orleans, May 1991. Addison-Wesley.
- Robinson, M. 1989. Double level languages and co-operative working. *Pages 79-114 of: Support, society and culture: Mutual uses of cybernetics and science*. March 27th-April 1st. Amsterdam.
- Rodden, T. 1991. Survey of CSCW. *Interacting with Computers: the Interdisciplinary Journal of Human-Computer Interaction*, 3(3), 319-353.
- Rodden, T, & Sommerville, I. 1991. Building Conversations Using Mailtrays. *Pages 159-172 of: Bowers, JM, & Benford, SD (eds), Studies in Computer Supported Cooperative Work: Theory, Practice and Design*. North-Holland.

- Romiszowski, A, & Jost, KL. 1990 (May). Computer Mediated Communication and Hypertext: An Approach to Building Structure into Distance Seminars. *In: Third Symposium on Computer Mediated Communication* University of Guelph, Guelph, Ontario, Canada. May 15-17, 1990.
- Root, RW. 1988. Design of a Multi-Media Vehicle for Social Browsing. *Pages 25-38 of: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26-28 1988. Portland, Oregon.
- Roseman, M, & Greenberg, S. 1992. GroupKit: A groupware toolkit for building real-time conferencing applications. *Pages 43-50 of: Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work* October 31 to November 4 1992. Toronto, Canada.
- Sarkar, M, & Brown, MH. 1992. Graphical fisheye views of graphs. *Pages 83-91 of: Proceedings of CHI'92 Conference on Human Factors in Computing Systems* Monterey, May 3-7 1992. Addison-Wesley.
- Sathi, A, Morton, TE, & Roth, SE. 1988. Callisto: An Intelligent Project Management System. *In: Greif, I (ed), Computer Supported Cooperative Work: A book of readings.* Morgan Kaufmann.
- Scheifler, RW, & Gettys, J. 1986. The X Window System. *ACM Transactions on Graphics*, 5(2), 79-109.
- Scrivener, SAR, Clark, S, Harris, D, Smyth, M, & Rockoff, T. 1992. Designing at a distance: Experiments in remote-synchronous design. *Pages 44-53 of: Rees, MJ, & Iannella, R (eds), Interface Technology: Advancing Human-Computer Communication. Proceedings of OZCHI 1992. CHISIG Annual Conference.* Bond University Australia, Queensland. November 26-27.
- Searle, JR. 1969. *Speech acts: an essay in the philosophy of language.* Cambridge University Press.
- Searle, JR. 1979. *Expression and meaning: Studies in the theory of speech acts.* Cambridge University Press.
- Shackel, B. 1987. An Overview of Research on Electronic Journals. *In: Proceedings of the Second International Conference on Human-Computer Interaction.* Honolulu, Hawaii. August 1987. Elsevier Science Publishers.
- Shackel, B. 1991. *Blend-9: Overview and appraisal.* Tech. rept. 82. British Library Reports.
- Shepherd, A, Mayer, N, & Kuchinsky, A. 1990. Strudel: An Extensible Electronic Conversation Toolkit. *Pages 93-104 of: Proceedings of the Third Conference on Computer Supported Cooperative Work* October 7-10 1990. Los Angeles.
- Shipman III, FM, Chaney, RJ, & Gorry, GA. 1989. Distributed hypertext for collaborative research: the virtual notebook system. *In: Hypertext '89 proceedings.* Pittsburgh, Pennsylvania November 1989. ACM Press.

- Shneiderman, B. 1987. Direct manipulation : a step beyond programming languages (excerpt). *Pages 461-467 of: Baecker, RM, & Buxton, WAS (eds), Readings in Human-Computer Interaction: A Multidisciplinary Approach.* Morgan Kaufmann.
- Shneiderman, B. 1993. Beyond intelligent machines: Just do it. *IEEE Software*, January, 100-103.
- Shuckburgh, ES. 1889. *The histories of Polybius.* Macmillan. London. Translated from the text of Hultsch, F.
- Smith, DC, Irby, C, Kimball, R, Verplank, B, & Harslem, E. 1982. Designing the Star user interface. *Byte Magazine*, 7(4). cited in pages 653-661 of (Baecker and Buxton, 1987).
- Smith, RB, O'Shea, T, O'Malley, C, Scanlon, E, & Taylor, J. 1991. Preliminary Experiments with a Distributed, Multi-Media, Problem Solving Environment. *Pages 31-47 of: Bowers, JM, & Benford, SD (eds), Studies in Computer Supported Cooperative Work: Theory, Practice and Design.* North-Holland.
- Sproull, L, & Kiesler, S. 1991. *Connections: New ways of working in the networked organization.* The MIT Press.
- Stallman, R. 1986. *GNU Emacs Manual.* Fifth edition, Emacs Version 18. edn. Free Software Foundation, 1000 Mass Ave, Cambridge, MA02138, USA.
- Stefik, M, Bobrow, DG, Foster, G, Lanning, S, & Tatar, D. 1987. WYSIWIS Revised: Early Experiences with Multiuser Interfaces. *ACM Transactions on Office Information Systems*, 5(2), 147-167.
- Stefik, M, Foster, G, Kahn, K, Bobrow, DG, Lanning, S, & Suchman, L. 1988. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. *Pages 334-366 of: Greif, I (ed), Computer Supported Cooperative Work: A Book of Readings.* Morgan Kaufmann.
- Tang, JC. 1991. Findings from observational studies of collaborative work. *International Journal of Man-Machine Studies*, 34, 143-160.
- Tang, JC, & Leifer, LJ. 1988. A Framework for Understanding the Workspace Activity of Design Teams. *In: Proceedings of the Second Conference on Computer Supported Cooperative Work* September 26-28 1988. Portland, Oregon.
- Tatar, DG, Foster, G, & Bobrow, DG. 1991. Designing for Conversation: Lessons from Cognoter. *International Journal of Man-Machine Studies*, 34, 185-209.
- Thimbleby, H. 1990a. *User Interface Design.* ACM Press, Addison-Wesley.
- Thimbleby, H, Cockburn, A, & Jones, S. 1992. HyperCard: An object-oriented disappointment. *Pages 35-55 of: Gray, P, & Took, R (eds), Building interactive systems: architectures and tools.* Springer-Verlag.
- Thimbleby, HW. 1986. *Equal opportunity interactive systems.* Tech. rept. YCS.80. Department of Computer Science, University of York, Heslington, York. YO1 5DD.

- Thimbleby, HW. 1990b (October). Liveware: A Personal, Distributed CSCW. *Pages 6/1-6/4 of: IEE Colloquium on CSCW* October 24, 1990 London. Digest No: 1990/132.
- Thimbleby, HW. 1991. Heuristic Strategies for HyperText. *In: Mindtools: Cognitive Strategies for Modeling Knowledge*, NATO ASI Series F, Proceedings NATO Advanced Research Workshop on Mindtools and Cognitive Modelling.
- Thimbleby, HW, & Greenberg, S. 1991. *The Weak Science of HCI*. Department of Computing Science, University of Stirling, Scotland.
- Thimbleby, HW, Anderson, S, & Witten, I. 1990. Reflexive CSCW: Supporting long-term personal work. *Interacting with Computers: the Interdisciplinary Journal of Human-Computer Interaction*, 2(3), 330-336.
- Utting, K, & Yankelovich, N. 1989. Context and Orientation in Hypermedia Networks. *ACM Transactions on Office Information Systems*, 7(1), 58-84.
- Wilson, P. 1987. Key Issues in Structured Messaging. *In: Working Conference on Messaging Systems* IFIP 6.5. Munich. April 1987.
- Wilson, P. 1988. Key Research in Computer Supported Cooperative Work (CSCW). *In: Sperth, R (ed), EUTECO 1988 Research into networks and distributed applications. 22nd April. Vienna.*
- Winograd, T. 1987. A Language/Action Perspective on the Design of Cooperative Work. *Human-Computer Interaction*, 3(1), 3-30.
- Winograd, T. 1988. Where the Action Is. *Byte Magazine*, December, 256-260.
- Winograd, T, & Flores, F. 1986. *Understanding computers and cognition: a new foundation for design*. Norwood, N. J.; Ablex.
- Witten, IH, Thimbleby, HW, Coulouris, G, & Greenberg, S. 1991. Liveware: A new approach to sharing data in social networks. *International Journal of Man-Machine Studies*, 34(3), 337-348.
- Woo, CC. 1990. SACT: A Tool for Automating Semi-Structured Organizational Communication. *Pages 89-98 of: Lochovsky, FH, & Allen, RB (eds), Proceedings of the Conference on Office Information Systems*. April 25-27 1990. Cambridge Mass.
- Woods, D. 1993. The price of flexibility. *Pages 19-25 of: Gray, WD, Hefley, WE, & Murray, D (eds), Proceedings of the 1993 International Workshop on Intelligent User Interfaces, Orlando, Florida. January 4-7*. New York: ACM Press.
- Yakemovic, KCB, & Conklin, EJ. 1990. Report on a Development Project Use of an Issue-Based Information System. *Pages 105-118 of: Proceedings of the Third Conference on Computer Supported Cooperative Work* October 7-10 1990. Los Angeles.
- Yoder, Y, Akscyn, R, & McCracken, D. 1989. Collaboration in KMS, a shared hypermedia system. *Pages 37-42 of: Proceedings of CHI'89 Conference on Human Factors in Computing Systems* Austin Texas, April 1989.

Appendix A

mnd Manual page

MND(1)

MND(1)

NAME

mnd - reminder

SYNOPSIS

Communicate event reminders to oneself and/or others. Reminders can be optionally attached to files, and various forms of notification are supported.

DESCRIPTION

mnd provides timely reminders of events and commitments. Reminders can be sent to oneself and/or others, and can be optionally attached to files.

When establishing a reminder the activation date (day of notification) must be specified.

Recorded reminders can be browsed and previewed in a variety of ways, described below.

When a reminder becomes current (its activation date passes), notification is made in one of three ways:

1. providing navigational aid to the associated commitment file (the **-r** option);
2. naming the commitment file and its path;
3. displaying the text of general (non-file-associated) reminders.

In the following expressions 'date' is of the form *dd/mm/yy*.

Valid expressions are:

mnd [**file_list**]

Show all currently active commitments. If the `file_list` argument is present, then only those active commitments which are linked to the named files are shown. The details shown about each commitment are: the path and name of the linked file, the date/time at which the reminder was set-up, and the text of the reminder.

mnd -s date [file] "reminder text"

Set up a reminder, optionally linking it to the named file. If the `file` argument is absent then a general reminder is created with the default file name, `reminderdd/mm/yy`, where `dd/mm/yy` is the activation date. Several commitments can be linked to a file, and are numbered (1 to n) when listed.

mnd -c username [username_list] date "reminder text"

Send a reminder to the named user(s). The reminder is installed in the named users' commitment file together with their personal reminders. Notification is made as for personal reminders (the `-s` option). `mnd` uses email to communicate reminders between users. On receipt of reminder messages, the message is autonomously filtered from the mailbox by the shell `mrc`. This process is silent to the recipient until notification on or after the activation date. Recipients without access to `mrc` receive `mnd` reminder messages as normal email.

mnd -d file [commitment_number_list]

Delete reminders associated with the named file. If `commitment_number_list` argument is absent all commitments linked to the file are removed. To remove a general reminder (one not linked to a file) the default file name (`reminderdd/mm/yy`) is used.

mnd -r

Add `@!` to the names of all files linked to active reminders. All parent directories also have `@!` affixed to the start of their name (aiding navigation to commitment files).

mnd -u file [file_list]

Remove `@!` from the named files. If all `@!` files are removed from a directory then the `@!` is removed from the parent directory name.

mnd -ua

Remove appended `@!` from all files in all directories.

mnd -p start_date end_date

Display all reminders set to be activated between `start_date` and `end_date`.

EXAMPLES

To establish a personal reminder that a monthly report (contained in file **project**) is due on the 15th of December:

```
mnd -s 15/12/90 project "midway report due today."
```

(it would probably be more appropriate to set the activation date some time before the due date.)

To remove the second commitment associated with the file **project**:

```
mnd -d project 2
```

To view all reminders set for the first two weeks in August:

```
mnd -p 1/8/90 14/8/90
```

To view all active reminders associated with "C" programs (assuming the standard **.c** naming conventions):

```
mnd *.c
```

SEE ALSO

mrc - filters the mailbox, extracts **mnd** reminder messages, and executes the **mnd** command contained therein.

FILES

the reminders are held in the file **\$HOME/bin/commfile**

Appendix B

Mona: User guide

B.1 Introduction

This document describes the use of *Mona*, an enhanced email system. *Mona* supports a window-based graphical user interface to email, and provides a variety of powerful email management facilities, primarily conversation based relationships between messages.

Mona's enhanced facilities are available for use at any time, but there is no requirement to do so—new users can continue to use email exactly as before, but through an improved, graphical, user interface. It is intended that with increasing familiarity, users will recognise and draw on the value of *Mona*'s conversational management facilities.

This user-guide is task oriented, so it can be used as a reference guide for assistance with the job at hand such as sending mail or reviewing a conversation; an index eases access to individual task descriptions. With the exception of section B.2, that describes the one time set-up process for *Mona*, reading this document from start to finish describes a “typical” day's work with email through *Mona*.

B.2 Installing *Mona*

Mona runs under the X window system in the Unix environment.

Mona requires a set of files and directories for its operation. These are created by the command:

```
set_up
```

The directories and files created by this command, and their purpose are described in section B.11.

Mona is best used as a continuously running system, started when you log-on and terminated when you log-off. To have *Mona* automatically start each time you log-on, add the line

```
mona -iconic&
```

to your `.x11start` file (in your home directory).

B.3 Getting started

Mona is started with one of the following commands:

`mona&` — after a short delay the outline of a window will appear on the screen. Moving the mouse moves the window outline. Click the left mouse button to place the window where the outline appears.

`mona -iconic&` — one of *Mona*'s icons (figures B.2) will appear on the screen.

B.3.1 *Mona*'s inbox

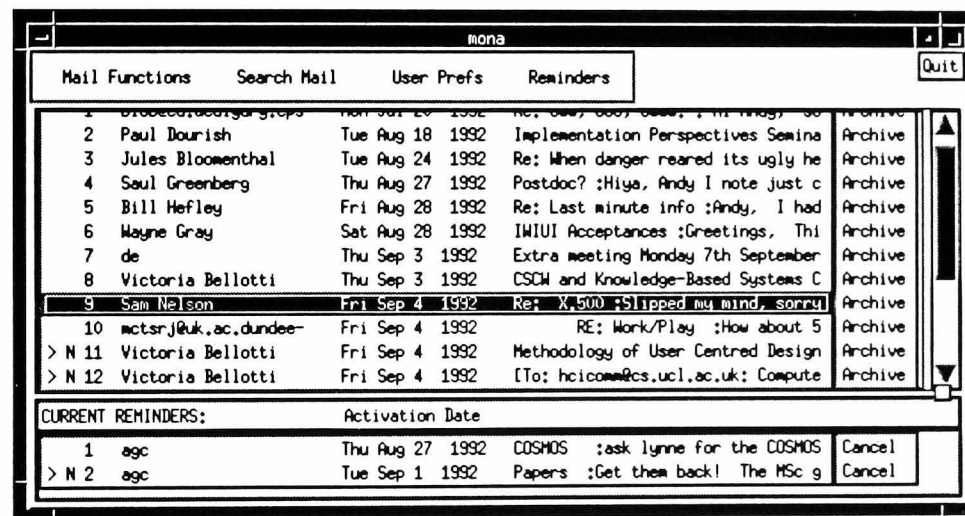


Figure B.1: The main *Mona* window.

Mona's main window (figure B.1) consists of three parts, described from the top of the window down:

- main menu bar — accessing many of *Mona*'s facilities. Menu items are “selected” by moving the mouse over the menu title (eg **Reminders**), pressing the left mouse button (a list of options now appears), moving the mouse down until the desired option is highlighted, and finally letting the mouse button go.
- mail inbox — each line in the inbox represents a mail message summary showing whether it is new or unread, its numerical ordering in the inbox, the name (or address) of the sender, the time and date at which it was sent, its subject (if any), and the first few words of the message. The **Archive** button alongside each message, when clicked, removes the message from the inbox and places it in the mail archive (see section B.4.5). As the mouse passes over a message summary (or the associated **Archive** button) the message summary is highlight—in figure B.1 the mouse-pointer is over message number 9;



Normal icon



Notification of new mail

Figure B.2: *Mona*'s inbox icons.

- reminder message display — the bottom of the window shows the currently requested reminder messages. By default the requested reminders are the currently active ones. Reminder messages are described in section B.6.

B.3.2 Sizing window fields

The relative sizes of the mail-inbox and reminder message display sections can be altered by the “field-separator”, a small box at the right hand side of the window, between the mail inbox and reminder message display. As the mouse moves over this box, the cursor changes to a cross-hair. By “dragging” the cursor (pressing the mouse button, holding it down, moving the mouse, and finally releasing the mouse button) the division of window space can be altered.

B.3.3 Iconifying the inbox

The inbox window can be iconized by clicking the dot by the top right corner of the window.¹ The inbox can be brought back from its iconized state by “double-clicking” (two rapid clicks on the left mouse button) on the icon. The inbox icons are shown in figure B.2—the left hand icon shows that there is no new mail, and the right hand icon notifies that new mail has arrived.

B.4 Reading mail

To read a message in the inbox click on the relevant message summary (message summaries in the inbox are highlighted as the mouse moves over them). A new window outline will appear, and clicking the left mouse button will place the message window in the position of the outline.

Mail message windows, figure B.3, consist of three parts (described from top to bottom):

- mail message menu bar — accessing a variety of facilities, primarily relating to the current message;
- the message header — showing, by default, a summary of the message header information, including the sender, receiver and Cc names, the date sent, and the subject;

¹Any *Mona* window can be iconized in the same way.

- the message body — displaying the text of the message. Messages may be too long to display in one window; the text of the message can be scrolled forward and back a page at a time (using the “Prev” and “Next” keys) or one line at a time using the cursor movement (arrow) keys. A summary of the commands available for editing and browsing text in windows is provided in section B.12.

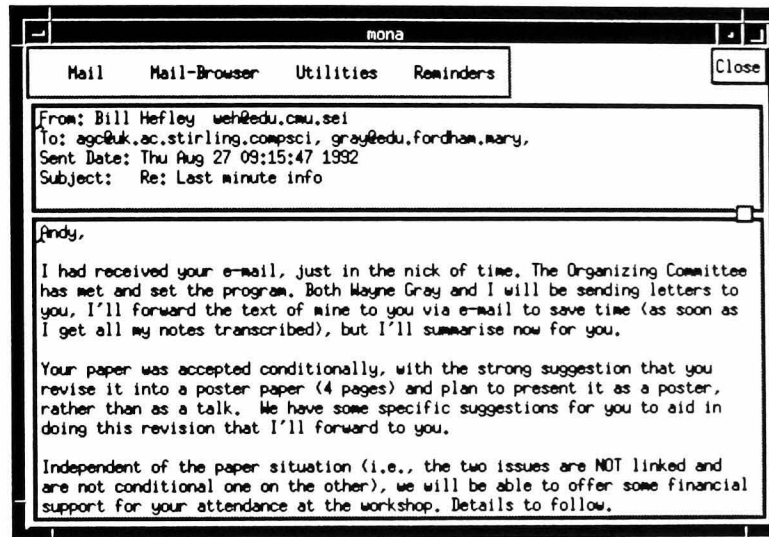


Figure B.3: A mail message window.

The division of window space between header information and message text can be altered by moving the field-separator (see section B.3.2).

Any number of messages can be simultaneously displayed. To avoid the problems of window clutter, see section B.8.

B.4.1 Header information

The whole message header can be displayed in the message header field by selecting **Full Header** from the **Mail** menu on the message’s menu-bar.

To restore the header information summary, select **Basic Header** from the **Mail** menu.

B.4.2 Saving a message

A message can be explicitly saved in a named file using the **Save** option in the message’s **Mail** menu.

A dialogue box (figure B.4) appears, requesting the file name. To enter a file name, move the mouse into the text entry area (the central field of the dialogue box) and type the name. Having typed the file name, click on the **OK** button. If a file of that name already exists *Mona* slightly modifies the file name, and reports the resultant name where figure B.4 shows **File Name (Hit OK when ready)**.

To close the dialogue box click the **Close** button.

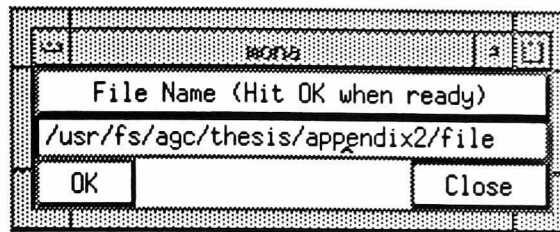


Figure B.4: The **Save** message dialog box.

B.4.3 Printing a message

Laser printed copies of a message can be produced by selecting one of the options from the hierarchical **Laser Print** menu (figure B.5). A selection is made from the hierarchical menu by raising the **Mail** menu as normal, moving the mouse to the **Laser Print** option, and then moving the mouse over the arrow at the right hand side; the hierarchical menu appears (figure B.5 and the particular type of header required can be selected by releasing the mouse button.

B.4.4 Closing a message

To close a message and make its window disappear, click on the **Close** button in the window's top-right hand corner.

B.4.5 Archiving messages

To avoid a cluttered main-inbox, it is advisable to remove messages from the mail-inbox when they are no longer urgently required. Messages can be moved back into the inbox (section B.7.7), or hidden temporarily by attaching reminder activation dates (section B.6).

To remove a single message from the inbox, placing it in the mail archive, click on the **Archive** button along-side the message summary line in the inbox. As the mouse moves over the **Archive** button, the button and its associated message are highlighted, confirming which message will be archived when the mouse button is clicked.

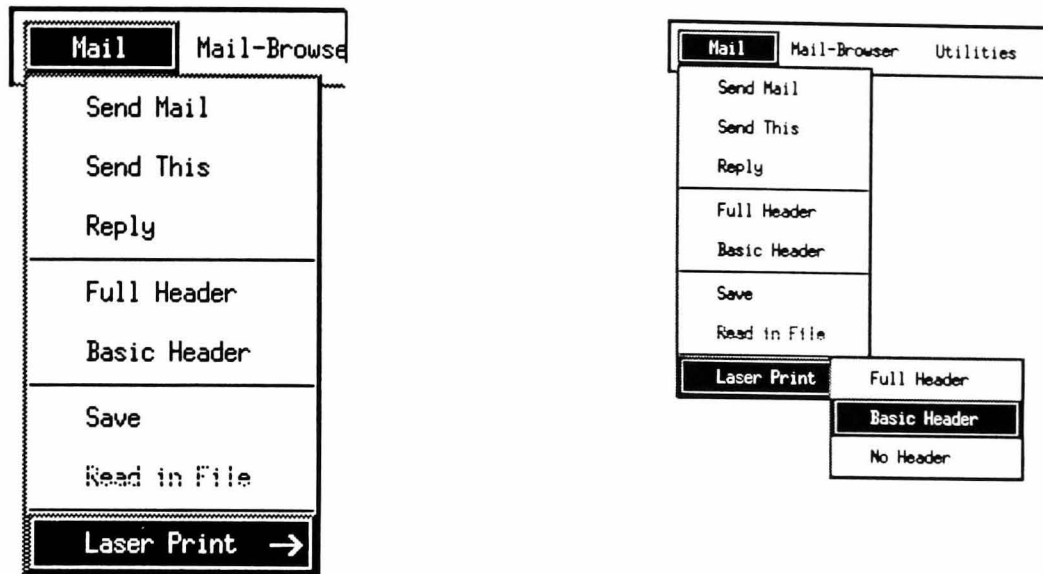
To archive several inbox messages at a time, select

Archive Mail from the **Mail Functions** menu in *Mona*'s main window.

A dialogue box will appear, and the numbers of the messages to be archived can be entered—for further information on how to use this dialogue box, see section B.7.8.

B.5 Sending messages

Mona supports several alternative ways of sending messages. The basic method is to select **Send Mail** from either the **Mail Functions** menu in *Mona*'s main window or from any



Initial selection of the
Laser Print menu

Hierarchical menu off the
Laser Print option

Figure B.5: Selection from a hierarchical menu.

message's **Mail** menu.

A blank mail message window appears with the default message-header prompts (**To:** and **Subject:**). The default header fields can be altered and extended by one of *Mona's* preference settings (see section B.5.8).

To send the message there are two things that *must* be done:

- at least one address must be present in the **To:** field. If the **To:** field is left empty the message will not be sent and an error report will be presented in a dialogue box similar to that in figure B.4;
- the **Send** button (below the **Close** button in the top-right of the window) must be clicked. If the window is closed (using the **Close** button) before the message is sent, a dialogue box will appear, asking if the message should be saved in the file `dead.letter`.

B.5.1 Addressing conventions

Multiple recipients can be addressed in the message header. Individual name/address combinations should be separated with spaces or commas. Several lines of **To:** fields are allowed. Any of the following are legal addressing schemes:

To: `agc@uk.ac.stir.cs`

To: `agc@uk.ac.stir.cs, gwh@uk.ac.stir.cs, salisbur@edu.washington.cs`

To: `agc@uk.ac.stir.cs gwh@uk.ac.stir.cs salisbur@edu.washington.cs`

To: agc@uk.ac.stir.cs, gwh@uk.ac.stir.cs
To: salisbur@edu.washington.cs malinow@de.siemens.zfe.dia8
To: joelle@fr.imag, begg@ca.mpr.mprgate

B.5.2 Message body

Although it is not essential to have any content in the message body, it is obviously sensible to do so! To type the message move the mouse into the message body part of the window and type. (see section B.12 for a summary of the text editing commands).

B.5.3 Dispatching (multiple) messages

Clicking the **Send** button sends the message to the list of recipients in the **To:** and **CC:** fields. Successfully dispatched mail is notified by a beep and the label on the **Send** button changes to **Sent**.

The same message can be sent again (to the same or different recipients) by clicking again on the **Sent** button.

B.5.4 Replying

To reply to a message select

Reply from the message's **Mail** menu.

Using **Reply** automatically fills in the return address in the **To:** field, and copies the message's **Subject:**.

The rest the message sending process is as described above (section B.5).

B.5.5 Forwarding

To forward a message to a (group of) recipient(s), select

Send This from the message's **Mail** menu.

A **Send** button will appear under the **Close** button (top-right of the message window). Both the header and message body parts of the window become editable, allowing you to add, delete, and alter their content.

To send the message, click the **Send** button.

B.5.6 Sending and including files

Files can be read into message windows containing a **Send** button. Select

Read in File from the message's **Mail** menu.

A dialogue box will appear requesting the name of the file to be read (figure B.4). Having typed the file name, click the **OK** button. Provided the file is found, its contents will be added after whatever text is already in the message body field.

If the file is not found another dialogue box will appear reporting the failure to find the file. Having checked the Unix file name, try re-typing the file name with its complete path name.

B.5.7 Spell checking

To check the spelling in messages (preferably prior to sending) select **Spell Check** from the message's **Utilities** menu.

A window will pop-up (figure B.6) displaying all the incorrectly spelled words in your message. Click this window's **OK** when your corrections have been made.

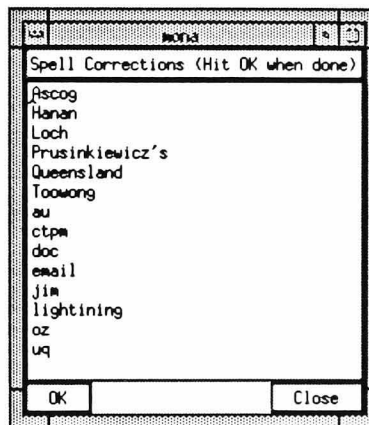


Figure B.6: Spelling correction window.

B.5.8 User-preferences when sending

Mona supports three preference settings relevant to the sending of messages, these are **Header Prompts**, **Footer Message** and **Address File**.

Header Prompts

The message header automatically provided when a message send window is requested (using one of **Send**, **Send This**, or **Reply**) can be modified by selecting **Header Prompts** from **User Prefs** on *Mona*'s main window menu.

A window similar to figure B.6 requests the entry new default header prompts. To save the new header prompts click the **OK** button.

Footer Message

A "signature" message can be automatically attached to the end of every message sent—for instance stating your name, postal address, telephone and fax numbers. To do so, select **Footer Message** from the **User Prefs** menu in *Mona*'s main window.

A window similar to figure B.6 displays the current footer message and allows it to be edited. To save the new footer message click the **OK** button.

Address aliases

Aliases for email addresses can be recored in an address file. Using aliases simplifies and eases addressing messages. For example, by adding the line

```
steve:mctsrj@uk.ac.dundee-tech
```

to the address file, the **To:** field in the message header need only contain

```
To: steve
```

rather than

```
To: mctsrj@uk.ac.dundee-tech
```

To alter the address file, select

Address File from the **User Prefs** menu in *Mona's* main window.

A window similar to figure B.6 displays the current address file and allows it to be edited. To save the modified address file click the **OK** button.

B.6 Reminders in *Mona*

Reminders in *Mona* are used for a variety of purposes, the most common being personal memory aids of events and actions that must be carried out—upcoming birthdays, project deadlines, and so on.

Reminders also serve a role in managing email. It is inconvenient and distracting to have messages lying about in your inbox just because they are pending some future action—by attaching reminder dates, such messages can be hidden until relevant.

Reminder messages are similar to normal mail messages, but have two additional fields in their window (figure B.7). These fields are the reminder activation-date field and the activation-date-calendar field.

Reminders become “active” when their activation-date becomes current (or has passed).

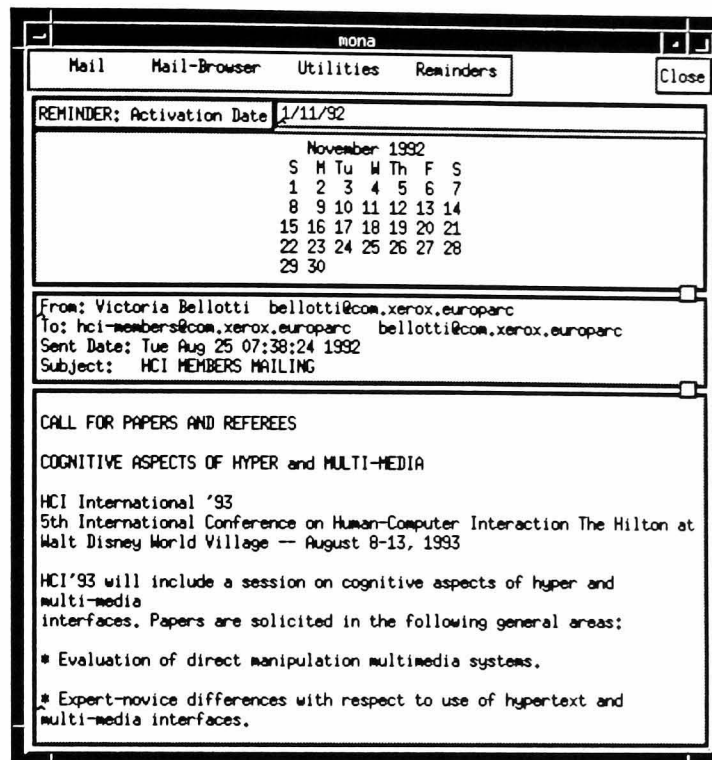
By default, the currently active reminders are displayed in the bottom field of *Mona's* main window (figure B.1).

Incoming reminder messages from other *Mona* users will only be notified when the associated activation date becomes current. Non-*Mona* users will receive reminder messages exactly as normal email.

B.6.1 Posting a reminder

To post a reminder select

Post from the **Reminders** menu of either *Mona's* main window or any message window.

Figure B.7: A reminder message in *Mona*.

A blank reminder message window will appear with the activation-date set to the current day, and the activation-date-calendar displaying the current month.

Change the activation-date as required—when the mouse leaves the activation-date display the activation-date-calendar field is refreshed, ensuring the calendar month matches the activation-date.

Write and send the message as for normal mail messages (see B.5)—you must supply the name/address combination in the **To:** field of the message header. To send the reminder to yourself, put your own user name in the **To:** field.

B.6.2 Reviewing (upcoming) reminders

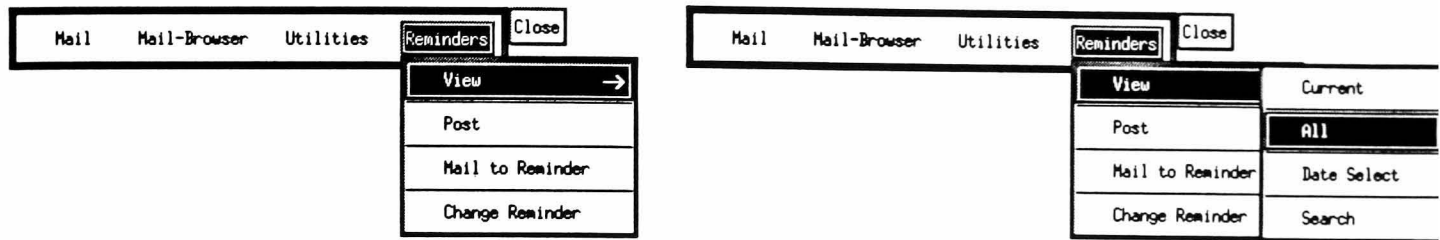
By default only the currently active reminders are displayed in the reminders field of *Mona*'s main window (figure B.1).

The title bar of the reminders field, in *Mona*'s mail window, states the selection of reminders displayed. This will be one of the following:

CURRENT REMINDERS: — the default selection, only currently active reminders are displayed;

ALL REMINDERS: — displaying all reminders stored. To display all reminders select **All** from the hierarchical menu displayed by moving the mouse over the **View** option under the **Reminders** menu in either *Mona*'s main window or a message window (see figure B.8);

PERIOD REMINDERS: — the reminders becoming active over a specified period. Having selected



Initial selection of the
View menu

Hierarchical menu off the
View option

Figure B.8: Selecting **All** reminders.

Date Select from the **View** hierarchical menu, the period date entry window appears (figure B.9). Type the period start and end dates into the relevant fields and click the **Search** button. Any reminders set to become active during that period will be displayed in the reminders field of *Mona*'s main window.

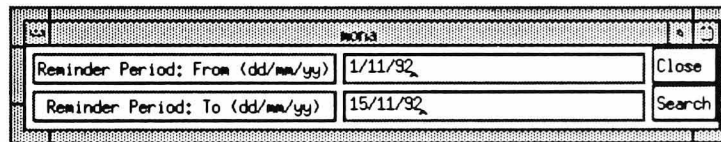


Figure B.9: The reminder period-entry window.

A search template for reminders satisfying a wider set of properties can be requested through the **Search** option of the **View** menu. This template is used in the same way as the mail archive search template—for a description of this facility see section B.7.6.

B.6.3 Converting normal mail into reminders

Any mail item can be turned into a reminder by assigning an activation date to it.

Either select

Mail to Reminder from the message's **Reminder** menu

or

Move to Reminder from the message's **Utilities** menu.

A window requesting an activation date will appear (figure B.10). Fill in the date field and click the **OK** button.

If the message is currently in *Mona*'s inbox it will be removed, and will appear in the reminder display when it becomes active.

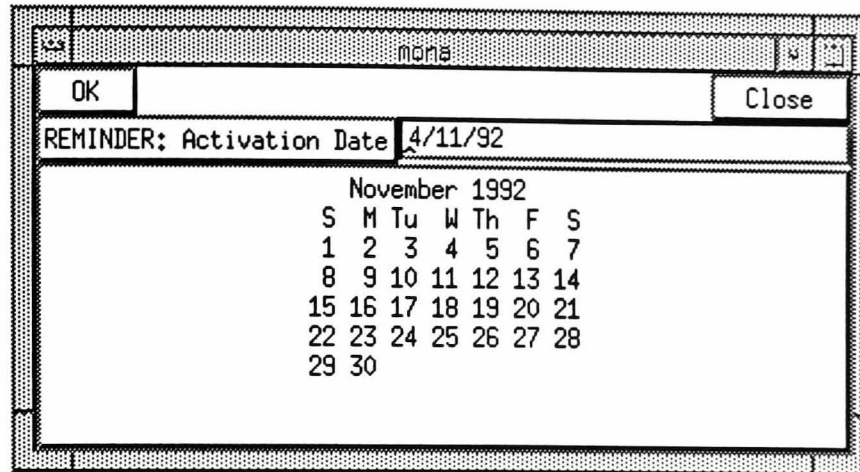


Figure B.10: Assigning an activation date to a normal mail item.

B.6.4 Changing reminders

Both the activation date and the text of reminder messages can be changed. Before changes take effect the reminder must be re-sent. If the reminder activation date is to remain unchanged, it is probably easier to change the reminder text using the **Edit File** option of the **Utilities** menu (see section B.7.9).

Select **Change Reminder** from the message's **Reminders** menu.

All fields in the window become editable (activation date, header, and reminder text). A **Send** button appears under the **Close** button in the top-right hand corner of the message window.

Having made the alterations click the **Send** button.

Note: the message will be sent to all users addressed in the **To:** field.

B.7 Mail management

All mail sent and received through *Mona* is automatically archived in an email database. *Mona's* mechanisms for accessing the archive provide effective email management enabling new email uses—the relation and organisation of information in *Mona* creates *new* information, thus enhancing collaborative work.

In this section *Mona's* facilities for accessing and managing the archive are described, together with hints on how to most effectively use them.

B.7.1 Mail in context

Mona infers the conversational context of all incoming and outgoing mail messages. In two party interactions, the relationship between messages is often clear—*A* writes to *B*, and this prompts *B* to write to *A*, and so on. *Mona* automatically establishes this type of conversational context, allowing easy access to a message's causes and responses.

Imagine receiving a message from a colleague that says “Yes, I’ll do that”. You might not be able to remember what “that” is... perhaps you’ve been on holiday...

Using *Mona*’s conversational relation facilities the context of messages can be quickly and easily accessed.

Mona’s conversational representation is not limited to two party interactions. Group discussions are also automatically related, and can be reviewed and browsed through a graphical “web” representing the conversation (see figure B.11). The conversational web enhances email’s ability to mediate group projects and decisions: the causes and results of past decisions can be reviewed; competing views, positions, and arguments can be examined and used as a basis for new decisions; and so on.

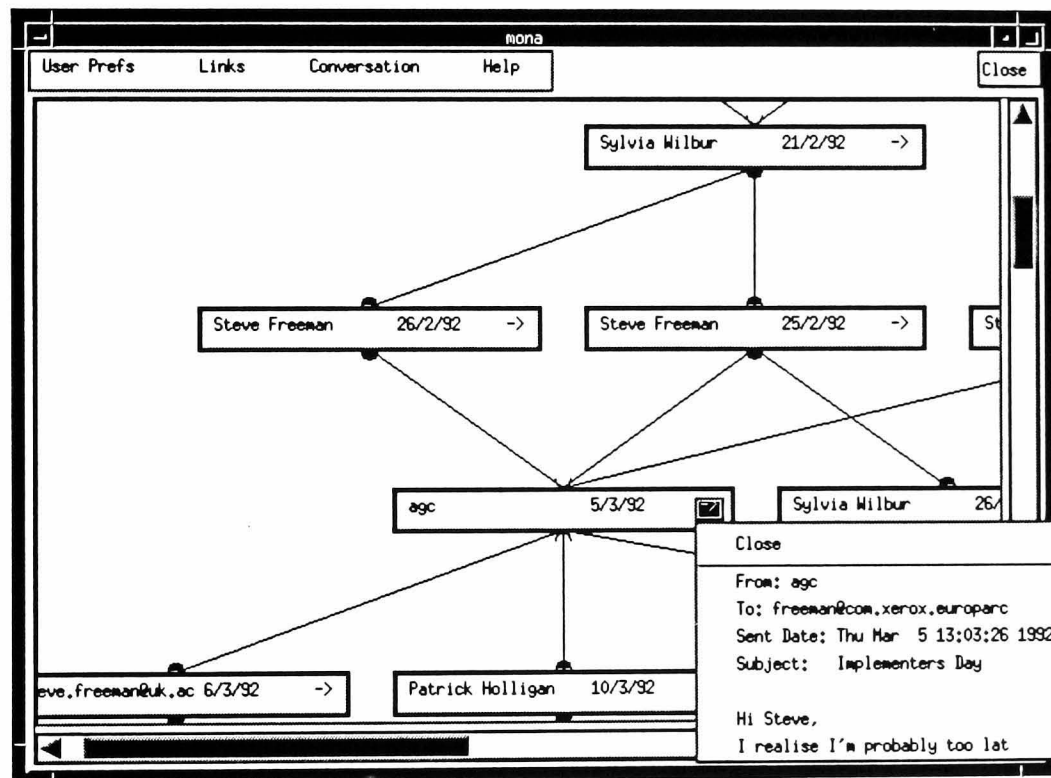


Figure B.11: An automatically inferred conversational web.

Mail links

Four message link types are supported by *Mona*. **Previous by Same** and **Next by Same** form a time ordered list of messages from a particular email source. **Cause** and **Response** links reveal the conversational context contributing-to and resulting-from any particular message.

Naturally, the conversational context inferred by *Mona*’s **Cause** and **Response** links does not necessarily match that perceived by the user. For this reason these links can be modified to reflect personal interpretations of the conversation; **Previous by Same** and **Next by Same** links are particularly useful in this modification process (see B.7.3).

Previous and Next links

The previous and next message from an email source can be directly accessed and displayed in its own mail window (figure B.3) by selecting either

Previous By Same or **Next By Same** from the **Mail-Browser** menu of the message's mail window.

Previous By Same is unavailable in the oldest message from a particular source. In these messages the menu option is written in faded text and can not be selected. Similarly, the **Next By Same** option is unselectable in the most recent message from a particular source.

Previous and next links cannot be explicitly modified by the user—deleting a message will, however, cause the links to be modified, maintaining the time-based ordering of messages.

Cause and Response links

Cause and Response links are automatically inferred when messages are sent/received; they allow email conversations to be examined and reviewed. From each message there may be any number of Cause or Response links. Although acting as a guide, the inferred message links do not necessarily agree with the user's interpretation of conversation structure, the links can therefore be changed (see B.7.3).

There are two ways of examining Cause and Response links:

- direct message access — similar to Previous and Next links (described in section B.7.1), a single Cause or Response is accessed and displayed directly. This method is suitable for two party interactions:
select **Cause** or **Response** from the message's **Mail-Browser** menu.
- the conversational web display — see section B.7.2.

Mailing lists

For *Mona* to assess the conversational context of mailing list messages it must be informed of the lists the user belongs to. Failing to do so will not cause adverse effects, but the system will fail to infer conversational relationship between messages posted to mailing-lists.

To inform *Mona* of mailing list membership, select

Join Mailing List from the **User Prefs** menu in the main *Mona* window.

A dialogue box (like figure B.4) will request the name of the new list, click **OK** when this has been typed.

To leave a mailing list, select

Leave Mailing List from the **User Prefs** menu in the main *Mona* window.

A dialogue box (like figure B.4) will request the name of the list, click **OK** when this has been typed.

To view the mailing lists that *Mona* has been informed of, select **View Mailing Lists** from the **User Prefs** menu in the main *Mona* window. A dialogue box (like figure B.6) will display the mailing list names, click **OK** when ready.

B.7.2 Conversational webs

To view, browse, and modify the conversational web relating to a particular message, select **Show Mail Web** from the message's **Mail-Browser** menu.

A web-window (figure B.11) showing a single “web-node” (see figure B.12) will appear.

Web nodes

Each web node is made up of the message sender's name, the date the message was sent, an arrow (preview) button, and two icons showing the availability of unexplored Cause and Response links—Causes branch out of the top of the node, Responses from the bottom. Closed icons (shown at the bottom of the web-node in figure B.12) represent exhausted paths in which all available links are displayed, open icons (top of figure B.12) represent unexplored links.

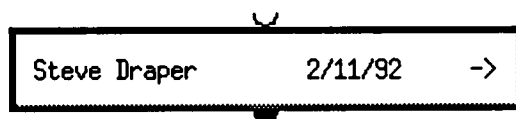


Figure B.12: A web node.

Pressing the mouse in the centre of a web-node displays the web-node menu (figure B.13), and pressing on the preview arrow (– >) provides a summary of the message content (figure B.13).

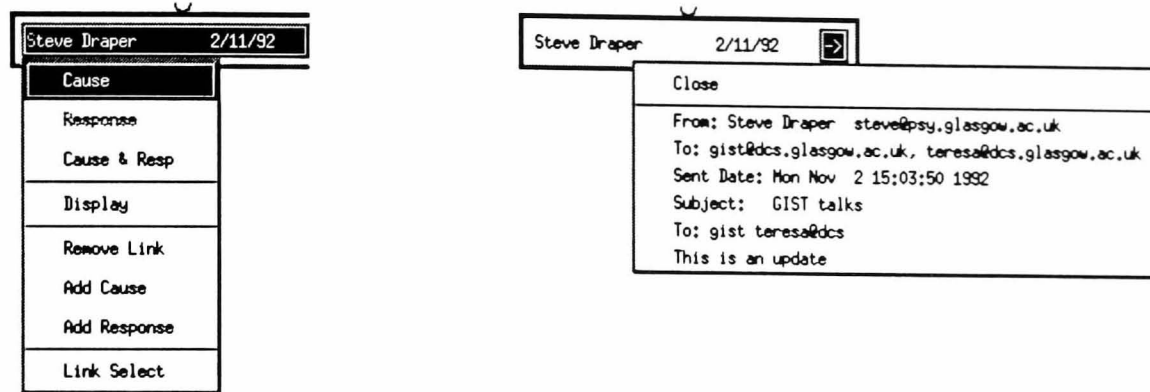
Web-nodes always display the sender's name, but the date, subject, and icons can be optionally displayed by altering the **Activate Icons**, **Activate Date**, **Activate Subject** options in the **User Prefs** menu in the web-window.

Browsing the web

The conversational web can be expanded and browsed using one of the three web-browsing options in the web-node menu (figure B.13), these are:

Cause — reveals a set of web-nodes representing all the messages causing the message;

Response — reveals a set of web-nodes representing all the messages responding to the message;



Web-node main menu

Web-node message preview

Figure B.13: Web-node menus.

Cause & Resp — the causes and the responses are revealed in one selection.

By default, selecting one of these options reveals one generation of the conversation—only the direct causes and responses are displayed. To speed up the growth of the conversational web, the number of conversation generations revealed in each selection can be increased. To do this, select

Reveal Layers # from the web-window's **User Prefs** menu.

A dialogue box will pop-up requesting the number of generations to be revealed with each Cause/Response selection. Click **OK** when this value has been set.

Note: When using **Reveal Layers #** values greater than one *Mona* checks for possible loops in the conversation structure. If a message is encountered twice in the web, a dialogue box stating that the **Reveal Layers #** value has been re-set to one is presented, and the web-nodes causing potential loops are highlighted.

To hide a particular web-node, select **Close** from the message preview icon (– >).

Displaying the node's message

To display the message represented by a web-node, select

Display from the web-node menu.

A message window will appear.

Naming conversations

Conversations that are frequently examined can be accessed directly by attaching a name. This is carried out in two steps:

1. select **Name This Conversation** from the web-window's **Conversation** menu. A dialogue box similar to figure B.6 appears requesting the name to be assigned (click the **OK** button when ready). The dialogue box now announces that you must **Link Select on Conversation Node**—this prompts step 2;
2. select
Link Select
 from the main menu of the web-node to be accessed as the main conversation node.

Accessing and removing named conversations

Immediate access to named conversations is provided by selecting **View Conversation** from either the **Conversation** menu of the web-window, or the **Mail Functions** menu of *Mona*'s main window.

A dialogue box (figure B.14) requests the name of the conversation to be displayed. Having typed the relevant conversation name, click **OK** to reveal to web-window with the conversation's main node.



Figure B.14: Requesting a named conversation.

To remove a named conversation (removing the name associated with the conversations, NOT the mail items themselves), select **Remove** from the dialogue box in figure B.14.

B.7.3 Modifying the web

The Cause and Response links automatically established by *Mona* can be altered; existing links can be deleted and any message can be attached to the conversation.

Adding new links

There are two steps in adding links to a web-node:

1. select either **Add Cause** or **Add Response** from the main menu of the web-node (figure B.13)—having done so, the cursor changes to a spider (figure B.15);
2. select
Link Select on the message to be attached (from either the web-node’s main menu or the message’s **Mail-Browser** menu)—the cursor now returns to its normal, arrow, state.

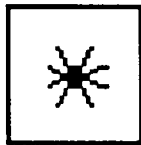


Figure B.15: The web modification “spider-cursor”.

Having selected either **Add Cause** or **Add Response**, the cursor changes to a spider (figure B.15). This reminds the user that *Mona* is waiting to be told which message is to be attached to the conversation. Any of *Mona*’s usual facilities can be carried out, but until step 2 (the **Link Select**) is carried out the cursor will remain a spider.

To cancel a link addition having selected **Add Cause** or **Add Response**, select **Cancel Link** from the web-window’s **Links** menu. The cursor will change back to its normal, arrow, state.

Removing existing links

There are two steps in deleting links (causes or responses) attached to a web-node:

1. select **Remove Link** from the main menu of the web-node (figure B.13)—having done so, the cursor changes to a spider (figure B.15);
2. select
Link Select
on the web-node at the other end of the link (from the web-node’s main menu)—the cursor now returns to its normal, arrow, state.

Mona’s usual functions can be carried out as usual between step 1 and step 2 of this process, but the cursor remains a spider to remind users of the pending **Link Select** action.

To cancel a link removal having selected **Remove Link**, select **Cancel Link** from the web-window’s **Links** menu. The cursor will change back to its normal, arrow, state.

B.7.4 Sharing web views

Views of a conversational web can be shared by users having common access to a Unix mail archive directory.

By default, the mail archive directory is held within the user's own file store, but this can be changed by selecting

Change Maildump from the **User Prefs** menu in the main *Mona* window.

A dialogue box (like figure B.4) is displayed, and having typed the new mail archive directory path click **OK**.

The mail in the changed mail archive can now be browsed in exactly the same way as normal.

B.7.5 Changing the inbox file

The file that *Mona* examines for new mail can be changed by selecting

Change Inbox from the **User Prefs** menu in the main *Mona* window.

A dialogue box (like figure B.4) is displayed, and having typed the new mail file path click **OK**.

B.7.6 Searching the archive

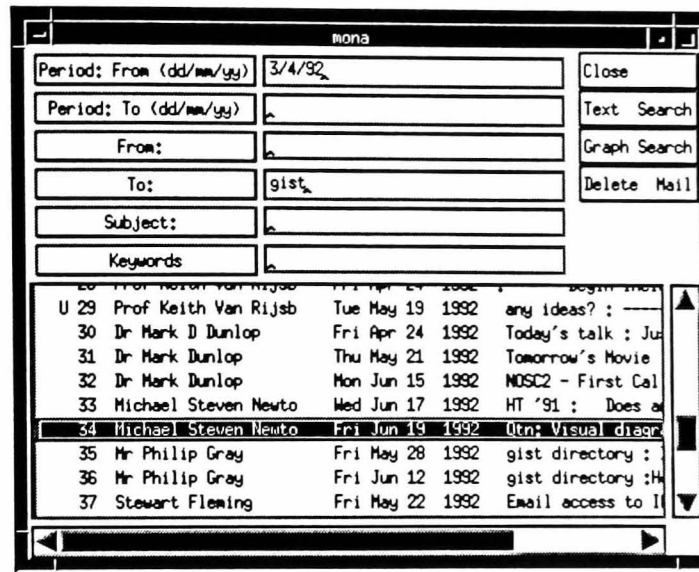
Items of mail can be selectively retrieved from the archive through the mail-search template (figure B.16). The mail-search template is requested by selecting **Search Mail** from any of the following menus:

- the **Search Mail** menu of *Mona*'s main window;
- the **Mail-Browser** menu in any message-window;
- the **Links** menu of the web-window.

A search-template will appear (figure B.16). The top-half of the search-template window contains a set of empty search-fields, used to selectively access mail in the archive. The results of each search are numbered and displayed in the bottom-half of the window. Clicking on any message displayed in the results will pop-up the relevant message window in a similar style to *Mona*'s main inbox (see section B.3.1).

The number of search-fields used in any search is optional, but at least one must be non-empty. The search-fields are:

- **Period: From (dd/mm/yy)** — the date at which the mail search should start. Mail items received prior to this date will not be selected;
- **Period: To (dd/mm/yy)** — the date at which the mail search should end. Mail received after this date will not be selected;

Figure B.16: *Mona*'s search template.

- **From:** — the name(s) or email address(es) of message sender(s). Several names may be listed, for example, john agc spm harold;
- **To:** — the name(s) or email address(es) of message receiver(s). Several names may be listed;
- **Subject:** — select messages with particular words in the subject field, for example, CSCW mail groupware;
- **Keywords** — select messages with particular words anywhere within the message.

Having filled in the relevant fields, click the **Text Search** button to initiate the search—this can take some time, and the cursor changes to a watch. You can continue to use *Mona* as normal while the search is underway.

Within each search-field several messages may be extracted (for instance, a **Keyword** search for **groupware** **CSCW** may find several messages, some matching **CSCW** and others **groupware**). Within any single search-field, then, messages resulting from the search are brought together (“ORed”). Across search-fields, however, messages are “ANDed”, that is, messages must satisfy each non-empty search-field to be included in the final result. In figure B.16 it is only those messages sent after 3/4/92 AND sent to **gist** that are displayed in the search results.

B.7.7 Moving mail to the inbox

Messages that have been moved out of *Mona*'s inbox (by clicking their **Archive** button), can be moved back into the inbox.

To replace messages in the inbox, display the message (using any of the available methods: message browsing links; conversational webs; search-template), and select **Move To Inbox** from the message window's **Utilities** menu.

The message will now appear as the most recent arrival (at the bottom) of the mail inbox.

B.7.8 Deleting messages

All incoming and outgoing mail is automatically recorded in the mail archive, thus with prolonged active use the archive becomes cluttered. *Mona* provides several types of assistance in deleting unwanted email items from the archive.

Deleted messages are moved to a deleted-mail-archive and stored for one week before final removal. If, having deleted an item, you wish to restore it, this can be achieved during the week between user-deletion and actual removal (see section B.7.8).

Single messages

Single messages can be deleted by selecting

Delete This from the message window's **Utilities** menu.

Multiple messages

From the inbox

A group of messages can be deleted from the mail-inbox by selecting **Delete New Mail** from the **Mail Functions** menu in *Mona*'s main window.

A dialogue box appears (similar to figure B.4), requesting the inbox numbers of the messages to be deleted—message numbers are displayed at the start of each message's line in the inbox.

In the dialogue box the message numbers must be entered in ascending order, and optionally separated by commas or spaces. A range of numbers (say, m to n) can be entered using the format $m - n$.

Any of the following techniques would delete messages 2, 3, 4, 7, 8, and 9.

2, 3, 4, 7, 8, 9

2 - 4, 7 - 9

2 3 4 7 - 9

From the search-template

The same dialogue box technique can be used to delete multiple (numbered) messages from the results of a search.

The search-template (figure B.16) allows the results of searches to be deleted by clicking the **Delete Mail** button in the search-template's top-right-hand edge. A dialogue box similar to that described above appears, and the numbers of the messages in the search template can be entered for deletion.

Cleaning up the archive

The least used items of mail in the archive can be displayed by selecting **Clean-Up Maildump** from the **Search Mail** option in *Mona's* main window.

A dialogue box requesting the number of least accessed mail items to display is presented. Having typed this number, and clicked the **OK** button, the least accessed mail items are displayed in a similar way to the results of a search-template. The numbers of the messages to be deleted can then be entered as above.

Recovering deleted items

When messages are deleted in *Mona* they are moved to a deleted-mail-archive where they are kept for one week before actual removal from the system. Messages in the deleted-mail-archive can not be read or displayed, but they can be searched for, and consequently moved back into the normal mail archive.

To search the deleted-mail-archive, select **Deleted Mail Search** from **Search Mail** in *Mona's* main window.

A search template like figure B.16 appears, and messages are selected from the deleted-mail-archive in exactly the same manner as the normal mail archive (see section B.7.6).

To restore messages to the normal mail archive from the search results, click on the **Un-Delete** button at the top-right-hand-side of the search-template. A message number dialogue box appears and is used to restore messages (see section B.7.8).

B.7.9 Editing messages in the archive

Mona allows any Unix file to be edited within a text-edit window (figure B.17). To edit a file, select

Edit File from either the **Search Mail** menu of *Mona's* main window, or the **Utilities** menu of any message window.

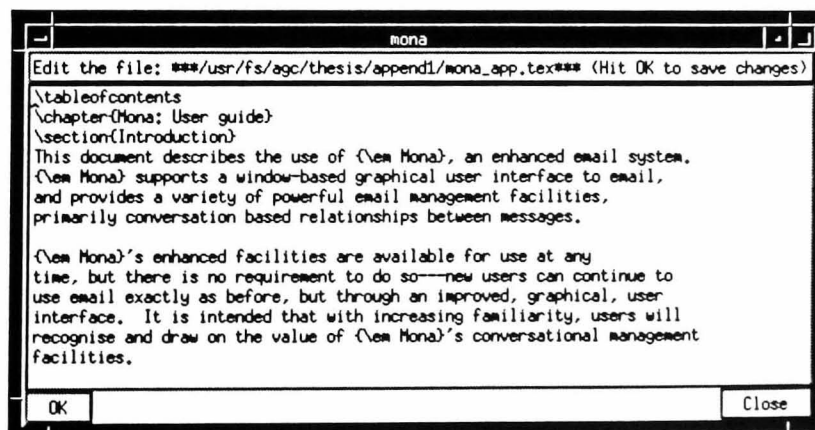


Figure B.17: A text-edit window.

A dialogue box, similar to figure B.4, requests the path and filename of the file to be edited. Type the path/file name and click the **OK** button to display the file. Having edited



Mail message icon



Search template icon



Web window icon

Figure B.18: *Mona's* window icons.

the file, click the **OK** button to save the changes.

When selecting

Edit File from a message window's **Utilities** menu, the name of the file in the dialogue box defaults to that of the mail message itself. Clicking **OK** without changing the file name allow the message to be edited directly. This technique is particularly useful for changing the text of reminders (maintaining an up-to-date “To Do” list, for example).

B.8 Avoiding window clutter

Like any window based system, without organised use, *Mona* can lead to a disorganised and cluttered screen. It is up to the user to control this, closing windows when no longer required and iconifying them when not immediately needed. *Mona* also includes features to reduce the confusion caused by window clutter.

B.8.1 Closing and iconizing windows

All windows, with the exception of the main window, contain a **Close** button; clicking on this button causes the window to disappear.

In the border of all windows (at the top-right-hand corner) there is a dot symbol that, when clicked, iconizes the window—shrinking it to postage-stamp size and displaying an identifying symbol. Double clicking on an icon causes the window to reappear in its previous position.

Mona uses a set of icon symbols, allowing the user to tell at glance the type of window the icon represents. *Mona's* inbox icons are shown in figure B.2, and its others in figure B.18.

B.8.2 Raising and hiding windows

When viewing messages in context, mail messages and other windows on the screen compete for the limited screen space. *Mona* eases the problems of overlapping messages and windows by allowing all message windows to be raised to the front of the screen. Select

Raise All Mail from the **Search Mail** menu in the main *Mona* window.

All message windows can be closed by selecting

Hide All Mail from the **Search Mail** menu in the main *Mona* window.

B.9 Quitting *Mona*

To shut-down *Mona*, click the **Quit** button in the top-right-hand corner of the main *Mona* window. A dialogue box will appear asking for confirmation—if you really want to quit, click the **OK** button in the dialogue box and all *Mona* windows and icons will disappear.

B.10 Common problems

B.10.1 My typing doesn't appear anywhere

The mouse must be in a text entry region before text can be typed.

Only out-going messages can be altered. You can only type into a message window if it has a **Send** or **Sent** button in the right-hand corner. If you are altering a message that will be forwarded later either use the **Send This** option (section B.5.5) or the **Reply** option (section B.5.4). If you really wish to alter a mail message that you have received use the **Edit File** option under the message's **Utilities** menu (section B.7.9).

B.10.2 The mail-window doesn't appear when I request it

Check to see if the message has been iconized. Double click on the icon (figure B.18) to read the message.

B.10.3 Saving and reading mail messages doesn't work

Try typing the entire path name in the save/read dialogue box. Also, when saving files, *Mona* will slightly modify file names (as required) to ensure that old-versions of files with the same name are not over-written.

B.10.4 Absent file error

This error message will appear when *Mona* fails to find a requested mail message in the mail-archive. This occurs when “special” messages have been deleted (for example, the main nodes in named conversations), or when trying to read messages resulting from a deleted-mail-archive search.

Before displaying messages in the deleted-mail-archive they must be **Un-Deleted** (see section B.7.8).

B.11 *Mona*'s files and directories

Mona uses a set of files in its processing and management of email. These files can be automatically created by the command

set_up

when typed in the user's home directory.

This command creates the following directories and files:

- **maildump.d** — the mail-archive directory containing all incoming and outgoing mail in separate files;
- **del_maildump.d** — the deleted-mail-archive directory; mail items explicitly deleted in *Mona* are moved to this directory for one week before deletion from the Unix file store;
- **.inbox** — *Mona*'s record of the messages currently in the inbox;
- **.convers** — a list of the explicitly named conversations and the name of the file representing the central conversation node;
- **.defheader** — the default header information presented to the user whenever sending mail;
- **.delsearchbox** — the results of the last search in the deleted-mail-archive;
- **.maillist** — a list of the mailing lists the user belongs to, and the file names of the messages sent to those lists;
- **.reminbox** — the reminders currently displayed in the reminders inbox (by default containing the currently active reminders);
- **.searchinbox** — the results of the last search in the mail-archive.

B.12 *Mona* text editing functions

forward 1 character	Ctrl F →
backward 1 character	Ctrl B ←
forward 1 word	⟨ <i>Meta Key</i> ⟩ F
backward 1 word	⟨ <i>Meta Key</i> ⟩ B
forward 1 paragraph	⟨ <i>Meta Key</i> ⟩]
backward 1 paragraph	Ctrl [
go to beginning of line	Ctrl A
go to end of line	Ctrl E
go to next line	Ctrl N ↓
go to previous line	Ctrl P ↑
go to next page	Ctrl V Next
go to previous page	⟨ <i>Meta Key</i> ⟩ V Prev
go to beginning of text	↖
go to end of text	Shift ↖
scroll text 1 line upwards	Ctrl Z
scroll text 1 line downwards	⟨ <i>Meta Key</i> ⟩ Z
delete next character	Ctrl D Delete char
delete previous character	Ctrl H Back space
delete next word	⟨ <i>Meta Key</i> ⟩ D
delete previous word	⟨ <i>Meta Key</i> ⟩ H
kill word	Shift ⟨ <i>Meta Key</i> ⟩ D
backward kill word	Shift ⟨ <i>Meta Key</i> ⟩ H
kill selection	Ctrl W
kill to end of line	Ctrl K
kill to end of paragraph	⟨ <i>Meta Key</i> ⟩ K
newline and carriage return	Ctrl J Insert line
newline and back up	Ctrl O
newline	Ctrl M

	Return
redraw text	Ctrl L

On HP workstations the Meta Key is the Extend char key

Appendix C

TELEFREEK: User guide

C.1 Introduction

This document describes how to use TELEFREEK, a platform providing access to and information about various types of communication tools and facilities. It increases awareness of “social presence” (who’s about), eases the initiation of interactions, tells users when “interesting” events occur, and much more.

TELEFREEK is a customisable system. The facilities it provides can be easily extended and modified to suit the individual’s (or group’s) communication requirements.

This user-guide describes TELEFREEK’s standard facilities, and demonstrates techniques used to extend or personalise the support it provides.

Although this document provides goal-oriented instruction on how to use and customise TELEFREEK, the best way to become familiar with it is to experiment: press the buttons, see what they do, define your own, and share your ideas with others. To borrow the advertising slogan considered for Apple Computer’s HyperCard system (Nielson, 1990b):

TELEFREEK: you figure it out!¹

C.2 Installing TELEFREEK

TELEFREEK runs under the X window system in the Unix environment.

It requires a set of files for its operation. These are created in a new directory `$HOME/telehost` by the command:

```
set_up_telefreek2
```

¹That said, a word of warning. TELEFREEK is a communications platform. Randomly clicking buttons, although a quick way to learn, may have an impact on others. It is recommended that, while experimenting, you keep yourself (or friends) as the currently named user (see section **Community List** below).

²In the environment at Stirling, the full path to this command must be supplied:
`/usr/fs/agc/telehost/set_up_telefreek`

The directory and files created by this command, their purpose, and their formats are described in section C.8.

TELEFREEK is best used as a continuously running system, started when you log-on and terminated when you log-off. To have TELEFREEK automatically start each time you log-on, add the line

```
telefreek -iconic&
```

to your `.x11start` file (in your home directory).

C.3 Getting started

TELEFREEK is started with one of the following commands:

`telefreek3` — after a short delay the outline of a window will appear on the screen; moving the mouse moves the window outline. Click the left mouse button to place the window where the outline appears;

`telefreek -iconic&` — one of TELEFREEK's icons (figures C.2) will appear on the screen.

C.3.1 TELEFREEK's main window

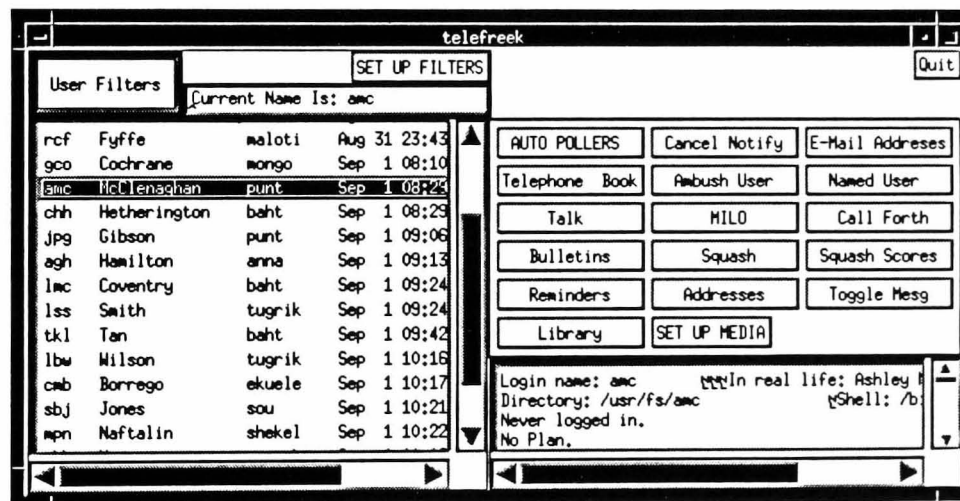


Figure C.1: The main TELEFREEK window

TELEFREEK's main window (figure C.1) consists of four primary components.

User filters — At the top-left of the window, the **User Filters** menu allows awareness of particular groups currently logged onto the network: see section C.4. Along-side the **User Filters** menu is a **SET UP FILTERS** button (see section C.7.1), and beneath this is a display stating “**Current Name Is:xxx**” that shows the currently “selected” person (used in many of the facilities described below.)

³At Stirling, the entire path to should be used: `$HOME/telehost/telefreek`



Figure C.2: TELEFREEK's icons

The community list — The scrollable list on the lower left of TELEFREEK's window shows an optionally filtered view of the user community revealing who (of interest) is logged onto the network. This display is updated every few minutes, maintaining awareness of the network community. Each line, such as

```
agc Cockburn      doodah   Nov 5 09:51 ttya
```

shows a user name, their real name, which machine they are on, and when they logged on. This information can be extended and customised (see sections C.7.1 and C.8).

When the mouse pointer moves over a line it is highlighted, and clicking the (left) mouse button “selects” the person shown (see section C.5.1).

Button utilities — the button-field on the right-hand side of figure C.1 accesses a variety of facilities and functions (usually, but not necessarily, concerning communication), some of which are described below. New functions can be added to this field (see section C.7.2).

The information display — at the bottom-right of the window is a read-only scrollable information display that is used to show the results of many TELEFREEK operations.

C.3.2 Iconifying TELEFREEK

To save screen space, TELEFREEK can be iconified by clicking the dot symbol in the top-right-corner of the window border.⁴ Windows can be restored from iconic states by double-clicking (two rapid clicks on the left mouse button) on the icon.

The TELEFREEK icons are shown in figure C.2. The left-hand icon shows the normal state, and the right-hand icon provides notification that an “event” (such as new mail, a new bulletin, the arrival of a friend, and so on) has occurred. Event notification requests are described in sections C.4.2 and C.7.3. The context of event notification can usually be determined by looking in the information display.

C.3.3 Quitting TELEFREEK

To stop TELEFREEK click the QUIT button in the top-right-hand corner of the main window. This action kills all processes initiated by the system.

⁴Any TELEFREEK window can be iconified in the same way.



Figure C.3: A typical user filter menu: releasing the mouse button at this point would select the **C Experts** group

C.4 Awareness of who is about

TELEFREEK's community list (on the left-hand-side of its window) shows the current filtered view of users logged onto the local network. The list is updated every couple of minutes, accounting for the arrival and departure of members of the current filter community. The current filter is changed using the **User Filters** menu.

C.4.1 User Filters

Particular community filters are selected through a pop-up menu—a typical **User Filters** menu is shown in figure C.3. Filters needn't limit the set of users shown: a special case filter could be **All Users** (see section C.7.1).

To select a filter, pop-up the menu (by moving the mouse pointer over the **User Filters** button and holding down the left hand mouse button), drag the pointer down (by moving the mouse, while still holding the mouse-button down), and release the mouse button when the desired filter is highlighted.

By default, after start-up, TELEFREEK displays the user-community filtered by the first (top) filter in the menu.

Customising the **User Filters** menu (adding/removing groups, changing their membership, and altering their order) is described in section C.7.1.

C.4.2 Tell me when he/she/they arrive

TELEFREEK can be requested to inform the user when particular people or group members log onto the network—called a “user ambush”. To request an ambush, move the mouse pointer over the **Ambush User** button, and click the left hand mouse button. An outline of a window will appear; moving the mouse causes corresponding movements of the outline. Click the left hand mouse button when the window-outline is in the position you wish it to appear (see figure C.4). This window allows the `.ambushnames` file to be edited (see section C.9 for a description of editing commands; see section C.8 for details of the `.ambushnames` file).

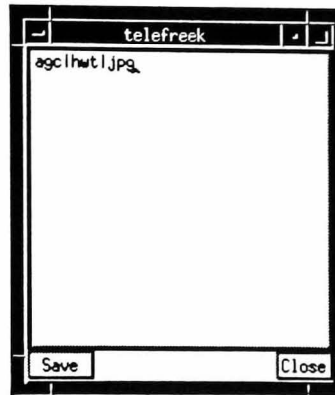


Figure C.4: The ambush users pop-up window

To ambush a user, or group of users, type their user-names (separated by the | character), and click the **Save** button in the bottom-left of the window. The window will disappear, and the system will silently wait for any of the users to arrive.

TELEFREEK announces the arrival of a member of the ambush group by activating its icon (see figures C.2), by providing an audible beep, and by sending a message saying who has arrived to the information display.

Any of the following lines would be legal user groups for ambushing:

```
jpg|hwt|sam|gco|chh|amc
agc
spm|gwh
```

Cancelling notification

Once TELEFREEK has announced a user's arrival the ambush is cancelled. Further ambushes can be requested by re-defining the ambush user names, as above.

The icon remains in its active state until the **Cancel Notify** button is clicked (reverting the icon to its normal state). Whenever the TELEFREEK icon is active, it can be de-activated by clicking **Cancel Notify**.

C.5 Queries about people

TELEFREEK can access and present a variety of information about individuals, groups, organisations, and so on. Without any modification or customisation it is likely that several such facilities will be supported by TELEFREEK (this is not necessarily the case as system administrators can alter the facilities automatically provided.) User's can customise and extend the utilities for accessing information: see section C.7.2.



Figure C.5: The Named User pop-up dialogue box

C.5.1 Information about those on the community list

Clicking on lines in the community list causes information about the person selected to be displayed in the information display.⁵ The name of the selected user is appended after the **Current Name Is:** display. The name shown in this display becomes the default for further enquiry or interaction.

C.5.2 Information queries through the button field

The button field is primarily intended to access customised facilities. The current implementation (at the University of Stirling) does, however, support a limited range of utilities that are automatically provided without any user customisation. These facilities are described below.

Setting the default user

The default user for enquiry or interaction (as displayed in **Current Name Is:** *xxx*) can be set explicitly as well as by selection from the community list. Consequently, many information and interaction facilities are eased, even when the user is not currently on the network.

To set the current default to a specific user-name, click the **Named User** button. This raises a pop-up dialogue box shown in figure C.5.

Move the mouse pointer into the middle section of the box and type the user-name (TELEFREEK will refuse names shorter than two characters, notifying refusal with a beep). Now click the **OK** button in the lower left-hand corner. Check the information in the information display to confirm the correct user is selected. The user-name also appears in the **Current Name Is:** *xxx* display.

If no information is shown in the display, the user-name entered is unknown on the network. Try looking-up the person using their real name, as described below.

The dialogue box remains visible until the **Close** button is clicked.

User lookup

The **Named User** button for setting the default user also assists in finding user-names.

⁵By default, this information is generated by the Unix `finger` command.

If unsure about a user-name, click the **Named User** button; a pop-up dialogue box appears (figure C.5). Move the mouse pointer into the middle section of the box and type the surname or first name. Now click the **OK** button.

If the search is successful, the information display will show some details about the person (full name, user-name, last login, etc).⁶ Now set the default user-name by typing the displayed user-name (first, backspace over the last query) into the dialogue box, and click the **OK** button again.

If the search is unsuccessful, you might try variant spellings of the user's name (first and last). If all these fail to return the correct user-name, it is likely that the person does not have an account on the network. This does not, however, mean that TELEFREEK can be of no service. Many TELEFREEK information services do not require an explicit user-name. Type the user's surname into the dialogue box, click **OK** (setting the "Current Name Is: " field to the surname), and try the facilities required, for example the **Telephone Book** described below.

Telephone directory

TELEFREEK can be used to look up telephone numbers. Currently, this facility is built into its functionality, and accesses the University of Stirling Campus directory).

Select a user-name, or enter a surname (as described above), and click the **Telephone Book** button. The results are displayed in the information display. If this is blank, no entry was found. Try an alternative spelling of the name.

Information browsers

As well as directed searches for information about particular people, TELEFREEK also supports general access to information sources, allowing users to browse for interesting topics, people, and so on. In some cases the information can be edited.

An example supported by unmodified TELEFREEK versions is the ability to directly access and edit the email alias file (`.mailias`).

Email addresses and aliases can be recorded in an address-alias file called `.mailias` stored in your `$HOME` directory. Using aliases simplifies and eases message addressing. For example, by adding the line

```
steve:mctsrj@uk.ac.dundee-tech
```

to the alias file, email can be addressed by the line:

```
To: steve
```

rather than

⁶In the current implementation, provided by the Unix `finger` command.

Figure C.6: Editing the `.mailias` file

To: `mctsrj@uk.ac.dundee-tech`

To alter the `.mailias` file, click the **E-Mail Addresses** button. A text-edit window (figure C.6) displays the current address-alias file and allows it to be edited (for editing commands, see section C.9). To save the file with modifications, click the **Save** button.

To quit the editor without saving any changes, click the **Close** button.

Customised information browsers are discussed in section C.7.2.

C.6 Establishing communications

The sections above have described techniques for establishing who is *available for communication* on the network (through the community list, or perhaps an “ambush”) and how to access *information about communication* (such addresses, addressing schemes, and naming conventions). Frequently, these actions will be followed by *initiating communication*. This too, can be eased through TELEFREEK.

TELEFREEK can launch a variety of communication channels or groupware applications. It is largely the user’s decision which channels and applications to access through TELEFREEK: customising these facilities will be described in section C.7.2. Without any customisation, the present version of TELEFREEK can access and initiate conversations using the Unix `write` facility.

To initiate a text-based conversation with the current named user, click the **Talk** button. After a short delay, a window outline appears under the mouse pointer. Move the window-outline to the position you require, and click the left-hand mouse button.

If, having placed the window, it immediately disappears **Talk** has failed: the user may have messages disabled, or may not be logged on. Try an alternative communication channel: email, for instance.

If the window remains, any text typed into it is sent to the named user whenever the

return key is pressed. To avoid confusion and garbled text simultaneous typing should be prevented: it is recommended that each line be kept short, and that a protocol of `-o` at the start of a new line should represent “over to you”, and `-oo` at the start of a new line for “over and out”.

To terminate the conversation, and kill off the talk-window, press the **Control** and **D** keys simultaneously.

C.7 Extending and customising TELEFREEK

Much of TELEFREEK’s power comes from its ability to be extended and customised by users. The interface for modifying the system is simple and easy to use, and much can be achieved by copying and adapting the mechanisms presented to users during customisation.

Give it a go!

C.7.1 User Filters

Customising the **User Filters** menu allows the user to maintain awareness of the people and groups of personal interest: friends, colleagues, project members, and/or those with specific areas of expertise.

To modify the **User Filters** menu, click the **SET UP FILTERS** button at the top-centre of the TELEFREEK window. A text edit window will appear showing the `.filters_file`, see figure C.7. The format of the `.filters_file` is further described in section C.8.

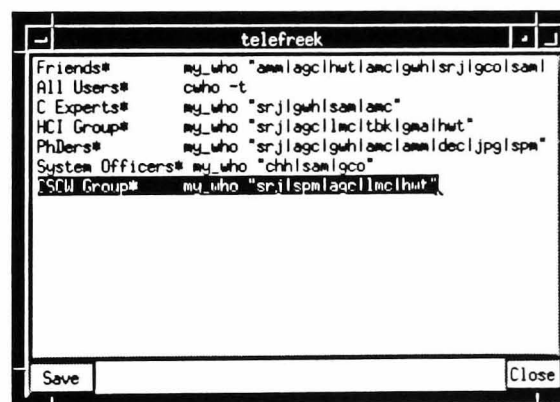


Figure C.7: The `.filters_file` edit window.

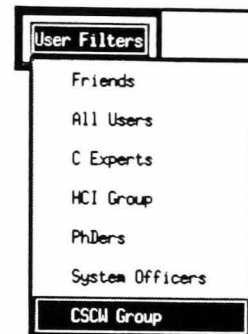
Lines in the `.filters_file` are of the form:

`OPTION LABEL* my_who user_list`

- **OPTION LABEL** is the tag that will appear on the menu option;
- ***** separates the option label from the command;
- **my_who** is a Unix shell script that extracts members of the `user_list` from the entire community of network users. In Stirling (the current TELEFREEK implementation site)



An example filters menu



Additional menu option resulting from changes to the filters form (see figure C.7)

Figure C.8: Modifying the filters menu

the entire network community is yielded by the command `cwho` (a variant of Unix `rwho`), see the second line (`All Users`) of the `.filters_file` in figure C.7;

- *user_list* consists of a list of one or more user names, enclosed in quotation marks, and separated by the character |

The first line in the filters file (and consequently the first option in the **User Filters** menu) is the default selection that TELEFREEK monitors after system start-up. The ordering of the lines can be altered though the text editing commands (section C.9). Members of user-lists can be added and removed, and new filters can be added.

To record changes made to the `.filters_file` and consequently modify the **User Filters** menu, click the **Save** button at the bottom left of the `.filters_file` edit window. The system briefly reconfigures, and the **User Filters** menu will be altered appropriately.

To cancel the alterations made without saving, click the **Close** button on the bottom-right of the window.

The change to the **User Filters** menu brought about by adding the highlighted line shown in figure C.7, is shown in figure C.8.

C.7.2 Button utilities

Button utilities enable easy access to a wide range of useful operations. If you carry out a command or command sequence frequently, consider adding it to TELEFREEK's button utilities. By clicking a button, operations that are hard to remember and burdensome to type can be executed.

The results of button utility operations can be displayed by TELEFREEK in a variety of ways (described below). TELEFREEK can also periodically check for events, and notify you when they occur: use of this facility is described in section C.7.3.

Button utilities are customised through the `.button_utils` file that is accessed and edited by clicking the `SET UP MEDIA` button in the button field (middle-right of the main TELEFREEK

window, figure C.1). The window for editing the `.button_utils` file is shown in figure C.9. Each line in the `.button_utils` file corresponds to a button in the button-field, and has the form:

```
BUTTON LABEL * command
```

The button label must be separated from the command it invokes by the `*` symbol. The *command* format is described in the following sections.

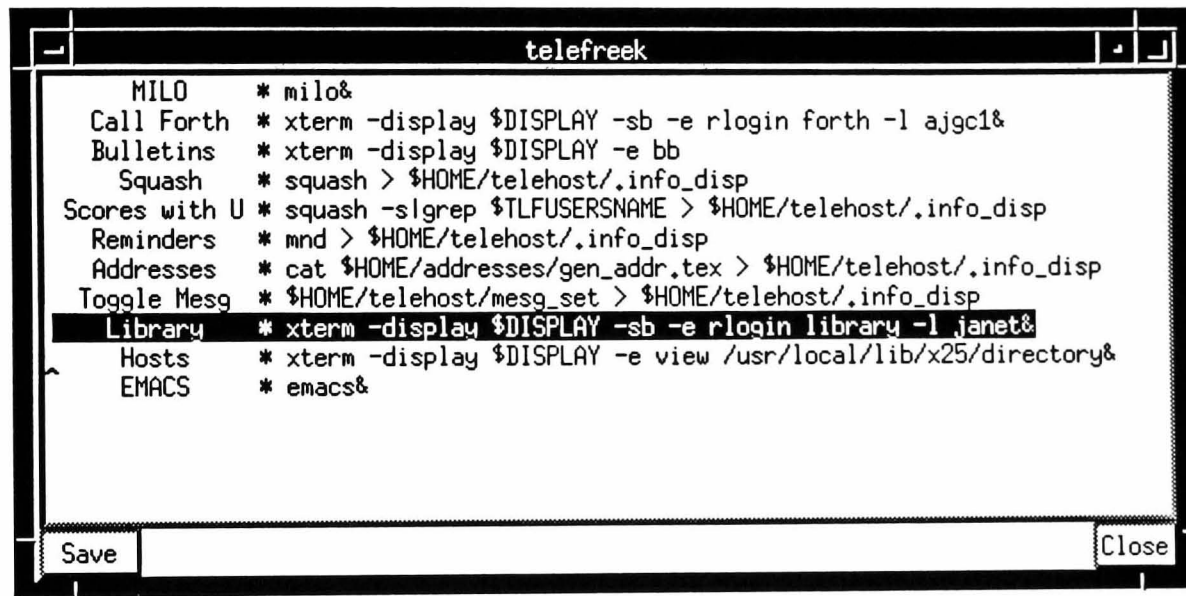


Figure C.9: Adding a “call library” function to TELEFREEK’s `.button_utils` file

When the `.button_utils` file has been modified, click the **Save** button to record the changes; the buttons will be re-drawn, and will include the new or modified facilities. To cancel the changes click the **Close** button prior to a **Save** operation.

Several example button utilities are given below. For a tutorial on the use of shell scripts, in which button utilities are written, see (Bourne, 1983).

Launching independent-window applications

Many applications create their own independent windows when launched (X-Windows applications for instance). Such applications can be added very simply to TELEFREEK’s `.button_utils` file.

Here are some examples:

```
MILO * milo&
EMACS * emacs&
```

MILO is a collaborative authoring X-Windows application available at the University of Stirling, Department of Computing Science (Jones, 1992b). Emacs is a powerful “extensible, customisable self-documenting display editor” (Stallman, 1986) that (usually, in an X-Window environment) creates an independent window when called.

The command to launch such applications is simply the name of the application. Consequently, this is the *command* necessary for TELEFREEK to launch them.

The **&** at the end of the *command* is necessary to enable TELEFREEK to continue operating while the new application is running.

Quitting an application activated through TELEFREEK automatically kills the window in which it was run.

Displaying information

Passively presenting useful information in the information display is another simple customization to TELEFREEK's button utilities. The output of Unix commands can be re-directed to the `.info_disp` file, causing it to be passively displayed in the scrollable information display (at the lower-right of TELEFREEK's main window).

Some example lines in the `.button_utils` file of this type include:

```
Reminders * mnd > $HOME/telehost/.info_disp
Squash    * squash > $HOME/telehost/.info_disp
Addresses * cat $HOME/addresses/gen_addr.tex > $HOME/telehost/.info_disp
```

The command `mnd` reveals the user's currently outstanding commitments (see (Cockburn, 1991b)); `squash` shows a departmental squash challenge ladder; `cat $HOME/addresses/gen_addr.tex` displays the contents of the file `gen_addr.tex` (a personal address file) held in the `$HOME/addresses` directory.

Each of these commands is followed by `> $HOME/telehost/.info_disp`. This re-directs the output of the preceding command segment to the `.info_disp` file. Information in the `.info_disp` file is automatically displayed in TELEFREEK's information display; it can be scrolled and browsed, but cannot be edited.

Most TELEFREEK button utilities send a message to the information display (sometimes as a side-effect of their primary action) to inform users of what action is being executed, and why.

Launching other applications and facilities

Although useful, the user-support enabled by launching independent windowing applications and by passively viewing documents is limited.

Another powerful use of TELEFREEK buttons is to initiate commands in a new window. This technique greatly extends the range of facilities accessible through TELEFREEK at the click of a button: editing frequently accessed files in a separate window; establishing communication links to remote sites; launching communication applications; and so on. Example lines in the `.button_utils` file include:

Call Forth * `xterm -display $DISPLAY -sb -e rlogin forth -l ajgc1&`
 Opens a new window, and establishes a login prompt on a remote machine.

Bulletins * `xterm -display $DISPLAY -e bb&`
 Launches a bulletin browser application in a new window (new bulletins are announced through TELEFREEK event capture: see section C.7.3).

Library * `xterm -display $DISPLAY -sb -e rlogin library -l janet&`
 Establishes a connection to a University library in a new window.

Hosts * `xterm -display $DISPLAY -e view /usr/local/lib/x25/directory&`
 Opens a useful (but hard to find and remember) file within a read-only editor in a new window.

All these commands are of the form `xterm -display $DISPLAY -e command&`.

- `xterm -display $DISPLAY` — invokes a new window;
- `-sb` — this optional part of the command (as in the Call Forth example) causes a scroll-bar to appear in the new window;
- `-e command` — executes *command* in the new window;
- `&` — allows TELEFREEK to continue operating while the new window and command are in operation.

When the command is terminated, the window will automatically close.

It must, again, be stressed that experimentation with these facilities is the best way to learn.

User-dependent facilities

The power of the functions described above can be further extended by making their operation dependent on the currently selected user.

To support and ease user-dependent facilities, TELEFREEK provides two “environment variables” called:

\$TLFUSERID — contains the currently selected user’s name as shown in the “Current Name Is: *xxx*” display at the top of the main TELEFREEK window. Although this is usually a login user-name (for example, *agc* or *jpg*), this is not necessarily the case as users can be selected explicitly through their real names (see section C.5.2);

\$TLFUSERSNAME — contains (whenever possible) the surname of the currently selected user.

Through appropriate use of these variables many user-specific operations become available, for example: communication channels can be launched with user addresses automatically supplied, and the details of a particular user can be selectively extracted from information sources. Examples include:

Email Named User * `xterm -display $DISPLAY -sb -e mail $TLFUSERID&`

Opens the Unix mail application in a new window with the user-name address automatically supplied;

User Squash Info* `squash -s|grep $TLFUSERSNAME > $HOME/telehost/.info_disp`

Displays (in the information display) the departmental squash ladder results in which the named user participated;

Another Talk * `xterm -disp $DISPLAY -e write $TLFUSERID&`

Initiates a Unix write conversation with the currently selected user in a new window.

C.7.3 Event monitors

TELEFREEK can be requested to notify the user when specific events occur. A simple event notification request is enabled by the **Ambush User** facility (see section C.4.2) that is built into TELEFREEK's functionality.

The range of events monitored by TELEFREEK can be extended through the use of shell scripts (for instance checking for the arrival of email, or the posting of new bulletins). These scripts can be automatically started whenever TELEFREEK is launched by placing the name of the script in the `.autopoller` file. To access the `.autopoller` file click the **AUTO POLLERS** button; a file edit window similar to figures C.7 and C.9 will appear.

In the `.autopoller` file, each call to an event capturing shell script should be on a new line, and should state the full path to the script file. For example:

```
$HOME/telehost/poll_bulletins&
```

Again, the line must end with the `&` symbol to ensure the process is run in the background (allowing TELEFREEK to continue operating). To save the changes made to the `.autopoller` file, click the **Save** button. To cancel the changes, click **Close**.

Event-capturing scripts announce events by sending output to the file `.autoout`. When `.autoout` is non-empty, event notification is made by activating the TELEFREEK icon, by making an audible beep, and by announcing the event in the information display.⁷

Writing event-capturing shell scripts requires some experience. An example is provided below: it traps the arrival of new bulletin board notices, and uses the existing commands:

⁷At event notification TELEFREEK automatically copies the contents of the `.autoout` file to the `.info_disp` file (see the file descriptions in section C.8).

bi — a command stating whether new bulletins have been posted. If no new bulletins are available **bi** gives the message `No new bulletins.`;

diff — shows the differing lines between two files;

echo — copies its argument to the output stream;

The script of `poll_bulletins` is:

```
x=1
while [ $x = 1 ]
do
    echo "No new bulletins." > $HOME/telehost/.bi_file
    bi > $HOME/telehost/.bi_file2
    diff $HOME/telehost/.bi_file $HOME/telehost/.bi_file2 >
        $HOME/telehost/.autoout
    sleep 120
done
```

Once invoked, this shell script sets a variable `x` to an unchanging value, making the `while` loop cycle continuously. The normal output of the `bi` command (`No new bulletins.`) is placed in a file `.bi_file`, and the actual result of the `bi` command is placed in the file `.bi_file2`. Any difference between the two files is directed to the file `.autoout`, causing event notification, and is also displayed in the information display.

The difference between the `No new bulletins` and `New bulletins` states, as displayed in the information display, would be similar to:

```
< No new bulletins.
---
> New bulletins have been posted to the system...
> 1 new bulletin posted to bulletin board 'general'.
```

If there is no difference between the files (both contain `No new bulletins`) no output is sent to `.autoout`.

After the file comparison, the process sleeps for 120 seconds before repeating the comparison.

Event notification is cancelled (resetting the icon to its normal state) by clicking the **Cancel Notify** button.

C.8 TELEFREEK files and variables

To assist customising TELEFREEK several files and variables are supported for the development of particular facilities and interface mechanisms. The files are held in the directory `$HOME/telehost`. This directory, the files (with default values), and a set of default TELEFREEK shell script commands are created by the command `set_up_telefreek` (see section C.2).

The shell scripts automatically provided by `set_up_telefreek` can be used as guides for the development of button utilities.

The files, their purpose, and their formats are described below. Note that these files begin with a period (.) and, consequently, will not be listed by Unix `ls` unless the option `ls -A` is used.

.comm_list contains the current filtered view of the network community as created by commands in the **.filters_file** (variations of the Unix `rwho` command). Information about each user must be on a separate new-line, and arguments in each line are separated by spaces. The first argument must be the user-name, and the second should be (but not essentially) the machine name. The remainder of the line is unconstrained. For example, any of the following are legal lines in the **.comm_list** file:

```
dme dinar Mar 9 08:03 ttys1
agc penny The cat sat on the mat
gma doobry
srj
```

The list is updated automatically every couple of minutes by the current **User Filters** selection.

.filters_file controls the options available under the **User Filters** menu. It can be accessed and edited in a separate window (see figure C.7) by clicking the **SET UP FILTERS** button, see section C.7.1.

Changes to this file (through the **Save** button in figure C.7) cause corresponding alterations to the **User Filters**. Example lines in the **.filters_file** include the following:

```
Friends* my_who "sam|gco|chh"
Andy* my_who "agc"
All Users* cwho -t
```

The ***** symbol separates the menu label from the command.

The command used (currently `cwho` and `my_who`) may be changed, but it must produce output in the format required by the **.comm_list**.

.info_disp information directed to this file is automatically displayed in the scrollable information display at the bottom right of the TELEFREEK window, see figure C.1.

Button utilities can put information into this file by using the Unix re-direction operator, > (for example, SAY HELLO * echo "hello" > .info_disp).

.button_utils contains your customised utilities appearing in the button field at the upper-right-hand side of the TELEFREEK main window, (figure C.1). **.button_utils** can be accessed and edited directly by clicking the SET UP MEDIA button. Buttons can be added, removed, and their behaviour changed by altering this file. The highlighted line in figure C.9 show the addition of a new command to access a University library. For further information, see section C.7.2.

.autopoller contains the functions automatically called by TELEFREEK at system start-up. Typically, these require periodic checks for state changes (the arrival of mail, posting of new bulletins, etc). The **.autopoller** file is accessed by clicking the AUTO POLLERS button. The scripts named in this file are responsible for re-calling themselves (allowing them to control their own frequency of re-initiation) and, if providing event notification, should direct their output to the **.autoout** file to cause event announcement. See section C.7.3.

.autoout Information in this file triggers a TELEFREEK notification event: icon activation (see figure C.2), an audible beep, and the contents of the **.autoout** file being displayed in the information display.

.ambushnames contains the user-names of those people tracked under the last (or current) ambush. The file is accessed by clicking the "Ambush User" button, and initiated by clicking the Save button in the resultant edit window. See section C.4.2.

TELEFREEK also supports two environment variables to assist the user's customisation of facilities. These are:

\$TLFUSERID — contains the currently selected name (as displayed by Current Name Is: *xxx*);

\$TLFUSERNAME — contains the surname (if available) of the currently selected user.

Use of these variables and files is further discussed and exemplified in section C.7.2.

C.9 TELEFREEK text editing functions

forward 1 character	Ctrl F →
backward 1 character	Ctrl B ←
forward 1 word	⟨Meta Key⟩ F
backward 1 word	⟨Meta Key⟩ B
forward 1 paragraph	⟨Meta Key⟩]
backward 1 paragraph	Ctrl [
go to beginning of line	Ctrl A
go to end of line	Ctrl E
go to next line	Ctrl N ↓
go to previous line	Ctrl P ↑
go to next page	Ctrl V Next
go to previous page	⟨Meta Key⟩ V Prev
go to beginning of text	↖
go to end of text	Shift ↖
scroll text 1 line upwards	Ctrl Z
scroll text 1 line downwards	⟨Meta Key⟩ Z
delete next character	Ctrl D Delete char
delete previous character	Ctrl H Back space
delete next word	⟨Meta Key⟩ D
delete previous word	⟨Meta Key⟩ H
kill word	Shift ⟨Meta Key⟩ D
backward kill word	Shift ⟨Meta Key⟩ H
kill selection	Ctrl W
kill to end of line	Ctrl K
kill to end of paragraph	⟨Meta Key⟩ K
newline and carriage return	Ctrl J Insert line
newline and back up	Ctrl O
newline	Ctrl M

	Return
redraw text	Ctrl L

On HP workstations the Meta Key is the Extend char key