

REDUCING USER EFFORT IN COLLABORATION SUPPORT

Andy Cockburn and Harold Thimbleby

Department of Computing Science
Stirling University
STIRLING, Scotland, FK9 4LA
+44 786 66741 (FAX 64551)
agc@ & hwt@uk.ac.stir.cs

ABSTRACT

The value of electronic mail as a medium for collaborative and coordinated work can be enhanced by relating messages to conversations. While some groupware systems have offered such facilities, their ability to assess conversational context is dependent on explicit user action and the use of specific systems by all collaborators.

This paper describes **Mona**, a novel conversation based email platform. Mona provides a hypertext representation of conversational context *without* requiring any additional effort from the user or the use of specific email systems by other collaborators. Mona's lack of requirements and independence is made possible by inferring conversational context with heuristics using information inherently transferred in all email communications.

Mona's heuristics are described, as are its mechanisms for personalising conversation views.

KEYWORDS: email, free guidance, conversational context, heuristics, Mona (a CSCW system).

INTRODUCTION

If an interactive system requires information from its user, there is a danger that the user will not supply it or supply incorrect or out of date information. Any inferences the system makes based on user explicit guidance may be unreliable; the user is then less motivated to provide the information. When many users are involved, aiming to collaborate, the problems are exacerbated as each user relies on *all* others correctly entering and maintaining their information. However, in user support, much relevant information is implicit in the structuring of the tasks performed; although this information is weaker than forcing explicit information from users, it can be obtained automatically. This costs users nothing, and it will necessarily be correct and timely.

This paper considers an automatic approach to user support in the context of cooperative work using email (electronic mail), and shows that the inferred conversational context is more useful than might be expected. Our approach avoids the twin problems of enhanced email systems: dependency on additional work by users, and inter-system

incompatibility. We introduce Mona, an operational email system embodying this automatic approach. Mona establishes conversation context independently of user actions through the use of heuristic inferencing techniques which draw on information in standard email communications. We also discuss the applicability of Mona's design considerations to other areas of CSCW research.

General reasons for email failure

Email is a medium that fails to meet its full potential. Inadequacies in message management, and problems arising from information overload commonly force users to adopt radical strategies in handling their email [4, 24]. As a consequence of such difficulties, email does not fulfill its potential as a support tool for collaborative work, despite its wide distribution and low technology entry level. With adequate management utilities the value of email for supporting collaborative work could be greatly enhanced, Belew and Rentzepis [1] discuss promoting its value to that of "a form of literature, worthy of the same preservation and augmentation that is typical of traditional printed media."

Several groupware research projects have also recognised the potential of email. The common aim in many of these projects has been to enhance email management by supporting conversational relationship between messages, allowing the review of past collaborative efforts, enhancing the maintenance of a group focus, and providing a platform for basing decisions and arguments. Theories of conversation have also been used to provide facilities beyond information management. Flores *et al* [11] discuss supporting an *active* role in project management: coordinating activities, providing reminders and identifying commitment defaulters. Email support based on conversation therefore promises to enhance our ability to manage and review communications in a natural manner, it also holds potential for assisting with commitment management. Yet, in practice, conversation based systems fail to provide these benefits [4, 13].

BACKGROUND

Mackay [24] identifies two alternative and opposite management strategies, those of the 'archiver' and the 'prioritizer.' Archivers maintain large and frequently unmanageable mail stores, afraid to discard items that may become valuable. Prioritizers discard mail whenever possible, aware that potentially valuable information is being lost they feel this is a necessary price to avoid information overload.

To support the variety of message management styles, message filtering schemes have been developed in which users specify actions to be taken when messages arrive satisfying particular sets of properties. Incoming messages are filtered by the receiver's rules, for example, messages from Joe Bloggs with the subject squash ladder are automatically deleted, while those from Tom Smith are assigned to an Urgent directory.

Filtering schemes were initially developed to ease the user's plight in information overload, for example, the Information Lens [25], and ISCREEN [29]. More generally, the power of message filtering schemes can be used to support toolkits for the development of coordination applications, for example, the Object Lens [26] and Strudel [33].

While message filtering schemes ease the individual's difficulties in managing email, conversational email augments its role as a medium for collaborative and coordinated work. Application domains for conversation schemes include email management, project management and coordination, and distributed education [30].

gIBIS [7], Strudel [33], WHAT [14], and SIBYL [22] use variations of the IBIS [19] method to provide conversation structure. Discussions are advanced by message types such as 'Objects-To,' 'Specialises' or 'Supports' which explicitly state each message's role in the current conversation. The Coordinator [11, 35] takes an active role in the process of project management, providing reminders, identifying commitment defaulters, and so on; the Coordinator's approach is motivated by speech-act theory [32] which, by indicating actions within messages enables the system to maintain a knowledge of commitments.

All systems above are dependent on additional work. The nature and mechanisms of the dependency is discussed below.

Dependency on structure and guidance

For all the applications discussed above additional information is required from the user or from messages (other users) to provide the augmented facilities. We call this extra work *guidance*. (Our aim will be to build a successful guidance-free system.)

Thus, when sending a message, the user must select an appropriate message type and fill in the associated fields. Some systems require that messages are split to enable identification of individual conversational 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?'

Goodman and Abel [12] state that, "People will use new communication systems to the extent that they require no more effort than existing ones." While this can be mitigated by the enhancement of work tasks, it is unreasonable to expect all users to carry out additional work every time they communicate. It may be *impossible* for some to carry out the actions required, particularly when working at different locations without the relevant systems; even when using the same system there may be incompatibilities between message structures [23].

Though the users' additional work (that is, guidance) may be acceptable when personal benefits are provided, disparities between those doing the additional work and those gaining the benefit from it is a major cause of failure [13]. The guidance required by email applications imposes a cost/benefit disparity at one of three levels:

- Message senders have a cognitive burden in selecting the appropriate message type, a process which can be non-trivial when messages contain several conversational moves, or discuss inchoate ideas. The burden is increased when no suitable types are available, in which case a new type must be defined or an existing one adapted. This effort, in addition to the requirement of filling in relevant fields, is imposed on senders for the benefit of the eventual receivers.
- Should the sender decide that the cost/benefit disparity is unwarranted and omit the additional work, then, for the benefits to be realised, a greater disparity must be imposed on some other user who executes the actions on the sender's behalf. The consequences of messages arriving without the required structural information are serious for systems taking an active role in the collaborative or coordination process, such as the Coordinator. If the additional work is not carried out the system's knowledge of the status of commitments will become non-current, causing problems such as redundant or mis-timed reminders.
- The sender does execute the additional work, but the receivers fail to gain the intended benefit. This is possible if a receiver does not support the same system, or if the knowledge of commitments maintained by a project management system is corrupt.

MONA: CONVERSATIONAL CONTEXT FOR FREE

Mona runs under Unix in the X window system [31], it provides a graphical user interface to Internet mail conforming to the standard specified in RFC822 [8]. Mona provides a variety of augmented email facilities, including a graphical representation of conversations. In contrast to systems providing similar facilities, however, Mona assesses the conversational context relating messages without requiring *any* additional actions from the message sender or receiver — essentially, Mona provides an interpretation of conversation structure 'for free.' Users can therefore use Mona with minimal impact on their existing working methods, the augmented management facilities are accessible immediately, but the system imposes no requirements on their use.

Design considerations in Mona

Much of the motivation behind Mona's development resulted from a desire to examine four design principles for cooperative work support tools. These principles [5] are:

- maximise the likelihood of system acceptance at a personal level;
- minimise system imposed constraints on users;
- minimise system requirements imposed on the user;

- develop systems to be as hardware independent as possible.

While Mona's overall design adheres to these principles, its provision of conversation facilities follows refined design considerations attending to the limitations of previous conversation support applications.

Avoid a Dependence on User Actions. Even with the best of intentions users cannot be depended on to satisfy a system's requirements for additional actions. Mona, therefore, does not require any actions beyond those of ordinary mail systems — the provision of email address(es).

Avoid a Disparity in Cost/Benefit. Heeding Grudin's [13] attribution of failure in CSCW to the disparity between those executing additional work and those gaining the benefit, Mona ensures that all additional work is motivated by personal benefit. (See below for fuller discussion.)

Allow Flexibility and Personalised Views of Conversations. The structure of conversations is open to personal interpretation [30]. A rigid conversation structure dictated by systems is therefore unlikely to match the structure perceived by each individual user. Mona automatically provides a flexible, personalized view of conversations. It can also be used to support group consensus views, established through group negotiation.

INFERRING CONVERSATIONAL CONTEXT

Mona's ability to infer context draws upon information available through email communication, in combination with a knowledge of previous mail items held in an archive containing copies of both incoming and outgoing mail. Information about each message is extracted from the header in RFC822 format including source and destination fields showing the email name/address of sender(s) and recipient(s), and a date field containing the time the message was sent. As messages are routed through relay hosts they have additional fields attached, revealing the route and rate of message progression. Optional fields include subject, return path, reply-to, references, and various other components which *could* be useful in assessing a messages conversational context. Research and experience, however, indicate that additional fields are unreliable sources for assessing context [3]: for example, subject fields often inaccurately describe message contents. This mismatch can be caused by automatic reply facilities copying a subject field. The reply feature gives the recipient's address, consequently it is valid even when the copied subject field is unrelated to the current message.

Mona uses header information 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 communications, contained in the local archive, Mona infers the probable relationships between messages, forming a *web* of conversational relationships.

Mona attempts to establish four link types with each 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 message cause(s) of a message

— **cause**;

the inferred message response(s) to a message

— **response**.

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...* as message variables and *u...* as users we define the rules as follows:

$$\begin{aligned}
 & a_{\text{previous by same}} = b \text{ when} \\
 & \quad \mathbf{prev_part}(a,b) \\
 & \quad c: \\
 & \quad \quad \mathbf{prev_part}(a,c) \\
 & \quad \quad c_{\text{send time}} < b_{\text{send time}}
 \end{aligned}$$

where

$$\begin{aligned}
 & \mathbf{prev_part}(a,b) = \\
 & \quad a_{\text{sender}} = b_{\text{sender}} \\
 & \quad a_{\text{send time}} > b_{\text{send time}}
 \end{aligned}$$

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

$$\begin{aligned}
 & a_{\text{next by same}} = b \text{ when} \\
 & \quad b_{\text{previous by same}} = a
 \end{aligned}$$

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.

The pair of links, **cause** and **response**, provide the conversational relationship between messages. They can be browsed with a graphical display of the conversation web.

The naming of links may over-stress the relationship between messages, 'influence' and 'influences' would be more accurate than a causal relation! However, it is intended that users will develop personal interpretations of link meaning without close attention to the specific terms used by Mona.

Whenever messages satisfy the conversational heuristic, Mona derives the conversational context that users may remove, augment and modify to reflect *their* personal interpretation of the conversation. There is no limit to the number of **cause** and **response** links stemming from any message, however, Mona's criteria are designed to moderate the number of links to avoid a potential explosion of relationships in a conversation.

When new messages are processed by Mona (both incoming and out-going messages) the **cause(s)** are determined as

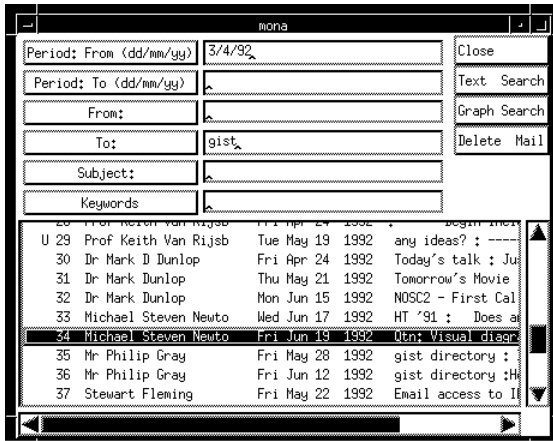


Figure 1. Selective retrieval of mail.

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.

A **cause** is defined:

$$a_{\text{cause}} = b \text{ when } \begin{array}{l} \text{conv_part}(a,b) \\ c: \\ \text{conv_part}(a,c) \\ c_{\text{receive time}} < b_{\text{receive time}} \end{array}$$

where

$$\text{conv_part}(a,b) = \begin{array}{l} a_{\text{sender}} = b_{\text{receiver}} \\ b_{\text{sender}} = a_{\text{receiver}} \\ b_{\text{receive time}} < a_{\text{receive time}} \end{array}$$

Cause and response links utilise message arrive-time (receive time) allowing quasi-causal effect to be based on events *observable* by the receiver [21]; 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:

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

The number of **causes**, under these rules, 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 a cause, a check is made to see if the message is addressed to a mailing list. Email addressing conventions make the heuristic inappropriate for inferring conversational context from mailing lists. The recipient fields of a mailing list need only contain the list name, while the sender field contains the name of the individual who posted the message. For this reason, the heuristic fails to find a **cause** because list names do not appear in the sender field — the condition $a_{\text{sender}} = b_{\text{receiver}}$ cannot be satisfied when the sender is a name in a mailing list.

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**. Responses are established as a consequence of each **cause**, according to the heuristic. To receive this separate mailing list inference of conversational context, Mona can be informed of each list the user belongs to.

USING AND MODIFYING CONVERSATIONAL CONTEXT

Through the use of heuristics, Mona provides what is *at worst* a zero cost, free, guide to conversational context, while remaining, *at best* an accurate reflection of the user's interpretation of conversational context. Due to the differences between individuals' interpretation of context [30], it may be argued that the best *any* system can provide is no more than a guide to conversation structure. Johansen [17] 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, while a search template (see Figure 1) supports selective retrieval of messages satisfying a variety of combined properties. The results of these searches may be displayed in a mail-inbox format or in a graphical format for easy inclusion in the conversation web.

Each node in the graphical conversation web (Figure 2) represents a single email item, identifiable through a combination of name and date. A pop-up mail summary is available through a preview key (represented by the \rightarrow symbol) and a separate window displaying the complete message may be requested through menu options associated with each node. Additional guidance 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 considerations — additional work is not *required*, but if carried out it provides personal benefit. In highly collaborative work, however, the distinction between personal and group benefit can become blurred [6]. 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 co-workers have a mutual understanding of their relevant commitments and responsibilities.

LIMITATIONS OF MONA

Mona is a working prototype providing an improved interface to email and several advanced features which augment management of email communications. For the purposes of practical use, Mona is only a prototype because of our design decision to experiment by deliberately limiting the functionality, in order to better expose the principles Mona was built to test. (Nevertheless, Mona is reliable and is available as C source code from agc.)

Borenstein and Thyberg [2] suggest the value of facilities provided by CSCW applications can only be substantiated if they achieve a “significant base of regular users.” Prototyping CSCW applications is therefore a formidable challenge — users will not incorporate new systems into their working routines unless they provide facilities already available in a usable manner *and* offer some additional incentive for adoption.

Mona satisfies requirements for completeness, usability and augmentation, however, it does not provide the structured and semi-structured mechanisms available in some other systems. Previous systems have depended on additional work by senders for the receivers benefit, Mona however, adopts the strategy of not supporting additional work in order to emphasise potential alternatives. While augmented facilities are made available through this scheme, the system fails to provide further enhancements for users wishing to carry out actions for the benefit of others. By combining structured and semi-structured schemes with free inferencing techniques, such as those described in this paper, users will be supplied with compatible and flexible conversation support, allowing social protocols rather than system requirements to mediate styles of use [10].

The heuristics Mona uses are (for the time being) fixed; however Mona can write out the archive of messages as a file of Prolog predicates, which can be analysed in any way,

and in principle could be used to extend the heuristics that Mona itself employs.

Although extensive heuristics might be a temptation, the design approach in Mona has been to be simple — Mona’s present heuristics are simple, allowing users to quickly become thoroughly familiar with their behaviour. More complex heuristics would most likely fail in obscure ways and in unfamiliar circumstances: normal email activity would not exercise them uniformly. In short, we believe that with more sophistication in guidance heuristics, the user would be worse off for presuming their usefulness based on current performance.

DISCUSSION

Reducing user effort through integration

“Seamless transition” across various dimensions of support for collaborative work has been widely cited as a method for reducing user effort [18, 20, 28]. The dimensions, or boundaries, which users must overcome include those between personal and group work, applications supporting various tasks undertaken as part of collaboration (which may or may not be computer supported), and modes of interaction which may be synchronous, asynchronous, co-located, or distributed. Integrated work environments such as the TeamWorkStation [16] reduce these boundaries, and consequently reduce effort. While Mona primarily uses heuristics to reduce dependency on user effort, it also adopts the “reflexive perspective” of CSCW [6] to increase integration across the personal/group work boundary.

TeamWorkStation’s method of “Video Fusion” allows the use of personally favoured applications, freeing users from the burden of learning (and remembering) separate interfaces. Providing for personal preferences in this manner satisfies another of the design considerations mentioned in this paper: that CSCW applications should maximise the likelihood of system acceptance at a personal level.

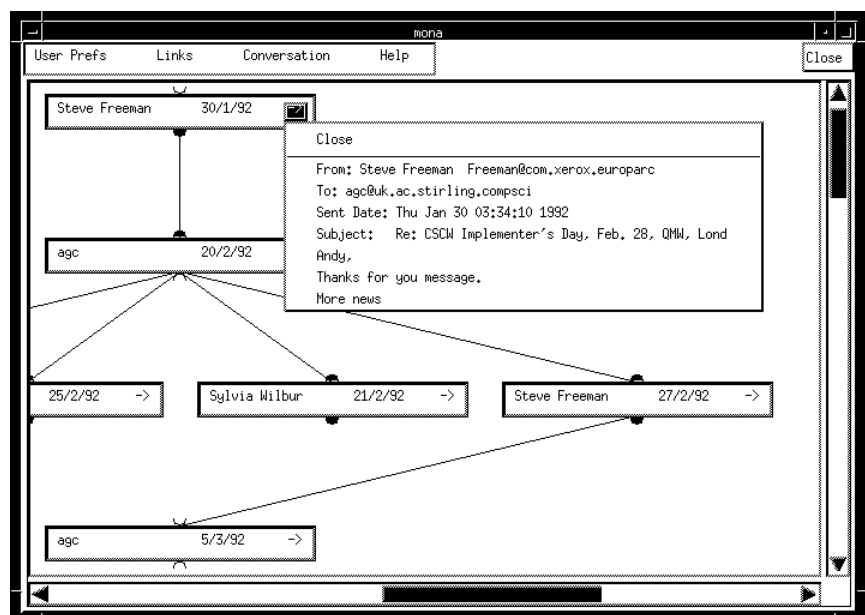


Figure 2. Automatically constructed conversational web.

Statistical versus deterministic heuristics

Mona's heuristics are deterministic: the way they affect features in the user interface is predictable knowing only the heuristics.

User effort may be reduced in many other ways, the most notable being prediction of what the user will do or prefers based on past activity. Such systems rely on heuristics that are statistical, and are based on group measures derived from information available to the system. Asymptotically, a statistical system may have better performance than a deterministic system such as Mona — but by that time, users may already have despaired of getting the system doing anything useful for them. In general, while a system is building up a statistical profile of the user, the variation in the features it provides will be too high for them to be *perceived* as reliable by the user.

Monk [27] describes one of many effort saving systems, in his case for HyperText systems. When a user repeatedly goes over the same hypertext link, the *Personal Browser* offers to establish explicit links and paths between the nodes. All systems of this sort, however, that rely on statistical analysis (Monk uses a simple frequency threshold) necessarily have behaviour that can only be properly understood statistically. A system may get “better and better,” but certainly in the short-run, its behaviour is unpredictable to all but the most sophisticated of users. If a system's behaviour is to be predictable to the user at all times, then its behaviour should not be dependent on any group statistic of the users' activities. (Thus, behaviour based on “most recent” is more predictable for the user than behaviour based on “most frequent”.) Mona does not base its features on statistical measures of the mail nor of users' actions, it is therefore predictable for the user at all times, and without assuming the user has a detailed knowledge of the mail archive. However non-functional features of a system can be usefully based on statistics: a simple example being a procedure to determine the telecommunications costs; a more sophisticated example being David Pullinger's statistical analysis of conversational currency (personal communication).

A specific application and survey of the wider implications of statistical systems is discussed in Darragh and Witten [9].

“For free” as a wider concept

The idea of performing useful actions for free involves tradeoffs. What if the user does not want these particular facilities? What if the user intentionally wanted to provide “guidance” in order to construct, short-circuit or fake a particular conversational context?

A similar set of tradeoffs arises in programming language design. Here, one generally wants the programmer (user) to be freed from certain detailed issues (such as garbage collection), since doing it “for free” ensures it will be done properly and that the programmer cannot accidentally cause certain sorts of error [34]. Thus garbage collection is a useful facility to provide for free, since doing so liberates the programmer from certain obligations (in this case, the obligations are primarily to the computer, rather than to other users — apart from the users of this program!). Yet

some programmers may be legitimately concerned with special forms of garbage collection: a scheme provided for free could limit their ability to express themselves. Conversely, it would limit their ability to make mistakes.

Providing facilities for free has its advantages, but is never without its critics, who want wider capabilities — and, we might add, have not always accepted the dangers and responsibility of the longer rope they now yield to hang themselves and others.

CONCLUSIONS

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. While such facilities are possible, their requirements and dependency on user actions can render them unrealistic. The realities of work pressure, lack of access to compatible tools, mistakes, even laziness, cause the omission of user-actions and a consequent loss in ability to provide enhanced facilities.

The inference of conversational context exemplified by Mona provides an alternative, free, approach to enhancing email as a medium for collaborative and coordinated work. While reducing effort is of obvious benefit to users, inferencing techniques also ease the problems arising from systems supporting incompatible information structures, and users failing to provide structured information.

The design principle of *minimising requirements imposed on users* that motivated the ‘free guidance’ approach, is applicable for a wide range of CSCW applications beyond augmented email. Establishing the necessary critical mass of users is a major hurdle for collaboration support tools. Similarly, difficulties in convincing users to change their working methods in order to satisfy system requirements is a major factor contributing to the failure of groupware. Minimising system specific requirements reduces dependency on critical mass — a system like Mona can offer 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.

Acknowledgements

Many thanks to colleagues who commented on early drafts of this paper. This research was funded by the Isle of Man Board of Education.

REFERENCES

1. RK Belew and J Rentzepis. Hyper Mail: Treating electronic mail as literature. In FH Lochovsky and RB Allen, editors, Proceedings of the Conference on Office Information Systems, April 25-27 1990. Cambridge Mass., pages 48–54, 1990.
2. NS Borenstein and CA Thyberg. Power, ease of use and cooperative work in a practical multimedia message system, *International Journal of Man-Machine Studies*, 32(2):229–259, 1991.

3. S Branskat. *How electronic mail is used: Implications for the design of e-mail systems*, November 1991. M.Sc Project Report: CSCW Class, University of Calgary. Alberta.
4. CV Bullen and JL Bennet. Learning from user experience with groupware. In *Proceedings of the Third Conference on Computer Supported Cooperative Work*, October 7–10 1990. Los Angeles, pages 291–302, 1990.
5. AJG Cockburn and SRA Jones. *Towards integrated environments for personal and collaborative work: four principles for design*. Working Paper, Stirling HCI Group. University of Stirling. Scotland, 1992.
6. AJG Cockburn and HW Thimbleby. A reflexive perspective of CSCW, *ACM SIGCHI Bulletin*, 23(3):63–68, July 1991.
7. J Conklin and ML Begeman. gIBIS: A hypertext tool for exploratory policy discussion, *ACM Transactions on Office Information Systems*, 6(4):303–331, October 1988.
8. DH Crocker. Standard for the format of ARPA Internet text messages; RFC-822, August 1982, ARPANET Working Group Requests for Comments: RFC-822.
9. J Darragh and IH Witten, *The Reactive Keyboard*, Cambridge University Press, 1992.
10. EA Dykstra and RP Carasik, Structure and Support in Cooperative Environments: the Amsterdam Conversation Environment, *International Journal of Man-Machine Studies*, 34:419–434, 1991
11. F Flores, M Graves, B Hartfield, and T Winograd. Computer systems and the design of organisational interaction, *ACM Transactions on Office Information Systems*, 6(2):153–172, 1988.
12. GO Goodman and MJ Abel. Communication and collaboration: Facilitating cooperative work through communication, *Office: Technology and People*, 3(2):129–146, 1987.
13. J Grudin. Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces. In *Proceedings of the Second Conference on Computer Supported Cooperative Work*, September 26–28 1988. Portland, Oregon., pages 85–93, 1988.
14. SH Hashim. WHAT: An argumentative groupware approach for organizing and documenting research activities, *Journal of Organizational Computing*, 1(3):275–302, 1991.
15. SR Hiltz and M Turoff. Structuring computer-mediated communication systems to avoid information overload, *Communications of the ACM*, 27(7):680–689, 1985.
16. H Ishii and N Miyake. Toward an open shared workspace: computer and video fusion approach of teamworkstation. *Communications of the ACM*, 34(12):37–50, 1991.
17. R Johansen. *Groupware: Computer Support for Business Teams*, Free Press, New York, 1988.
18. MA Keeler and SM Denning. 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, 1991.
19. W Kunz and H Rittel. *Issues as elements of information systems*, Technical Report Working Paper Number 131, Institute of Urban and Regional Development, University of California, Berkeley, California, 1970.
20. H Krasner, J McInroy, and DB Walz. Groupware research and technology issues with application to software process management. *IEEE Transactions on Systems, Man and Cybernetics*, 21(4):704–712, July/August 1991.
21. L Lamport. Time, clocks, and the ordering of events in a distributed system, *Communications of the ACM*, 21(7):558–565, 1978.
22. J Lee. Sibyl. A tool for managing group decision rationale. In *Proceedings of the Third Conference on Computer Supported Cooperative Work*, October 7–10 1990. Los Angeles, pages 79–92, 1990.
23. J Lee and TM Malone. Partially shared views: A scheme for communicating among groups that use different type hierarchies, *ACM Transactions on Office Information Systems*, 8(1):1–26, 1990.
24. WE Mackay. More than just a communication system: Diversity in the use of electronic mail. In *Proceedings of the Second Conference on Computer Supported Cooperative Work*, September 26–28 1988. Portland, Oregon, pages 344–353, 1988.
25. TW Malone, KR Grant, K-Y Lai, R Rao, and D Rosenblatt. Semi-structured messages are surprisingly useful for computer-supported coordination. In I Greif, editor, *Computer Supported Cooperative Work: A Book of Readings*, pages 311–331. Morgan Kaufmann, 1988.
26. TW Malone and KY Lai. Object Lens: A “Spreadsheet” for cooperative work. In *Proceedings of the Second Conference on Computer Supported Cooperative Work*, September 26–28 1988. Portland, Oregon., pages 115–124, 1988.
27. AF Monk. The Personal Browser: A tool for directed navigation in hypertext systems, *Interacting with Computers: the Interdisciplinary Journal of Human-Computer Interaction*. 1(2):190–196, 1989.
28. M Ohukubu and H Ishii. Design and implementation of a shared workspace by integrating individual workspaces. In FH Lochovsky and RB Allen, editors, *Proceedings of the Conference on Office Information Systems*. April 25–27 1990. Cambridge Mass., pages 142–146, 1990.
29. S Pollock. A rule-based message filtering system, *ACM Transactions on Office Information Systems*, 1(2):233–254, July 1988.

30. A Romiszowski and KL Jost. 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, May 1990.
31. RW Scheifler and J Gettys. The X Window System, *ACM Transactions on Graphics*, 5(2):79-109, April 1986.
32. JR Searle. *Speech acts: an essay in the philosophy of language*, Cambridge University Press, 1969.
33. A Shepherd, N Mayer and A Kuchinsky. Strudel: An extensible electronic conversation toolkit. In *Proceedings of the Third Conference on Computer Supported Cooperative Work*, October 7-10 1990, Los Angeles, pages 93-104, 1990.
34. H Thimbleby. A Literate Program for File Comparison, *Communications of the ACM*, 32(6):740-755, 1989.
35. T Winograd. A Language/Action Perspective on the Design of Cooperative Work, *Human-Computer Interaction*, 3(1):3-30, 1987.