Automatic Conversational Context: Avoiding Dependency on User Effort in Groupware

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

Relating individual messages to their on-going conversations enhances the value of electronic mail as a medium for collaborative and coordinated work. Some groupware systems have offered these facilities, but their ability to determine conversational context is dependent on explicit user actions — being told — and the use of specific systems by all users involved.

This paper describes **Mona**, an email system that provides an automatic hypertext representation of conversational context. Mona is novel in that conversation facilities are provided *without* requiring any user effort or the use of particular systems by other collaborators. This lack of requirements and independence is made possible by *inferring* conversational context with heuristics from information inherent in all email communications.

Mona's heuristics are described, together with its central design motivation: that the cost/benefit disparity resulting from dependency on user actions is liable to cause system rejection.

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

Introduction

Despite its wide distribution and low-technology entry level, email (electronic mail) is a medium that fails to meet its full potential. Inadequacies in message management, and problems arising from information overload commonly frustrate its use in collaborative and coordinated work (Bullen and Bennet, 1990; Mackay, 1988). However, with adequate management utilities the value of email for supporting collaborative work could be greatly enhanced, Belew and Rentzepis (1990) 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 support 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. Beyond passively organising email into conversations, theories of conversation have allowed systems to take active roles in project management: coordinating activities, providing reminders and identifying commitment defaulters. Conversational email 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 (Bullen and Bennet, 1990; Grudin, 1988). We contend that this failure is largely due to system requirements for additional information from users.

Interactive systems depending on user-supplied information are susceptible to the user (or anyone else) failing to supply it or supplying incorrect (even malicious) or merely out of date information. Thus, any inferences based on user explicit guidance may be unreliable; consequently users become less motivated to provide the information the system relies on for its success. Problems are exacerbated in collaboration support as each user relies on all others. Everyone must be motivated. Furthermore, systems rely on a 'critical mass' of sufficient users doing sufficient activity before users' effort begins to pay off. In short, reliance on explicit user-supplied information has severe limitations. However, much relevant information is often implicit in the structuring of the tasks performed. Although this information is weaker than the best that can be

explicitly provided by users, it can be obtained automatically, costs the users nothing, and is necessarily correct, timely, and available.

This paper details an automatic approach to cooperative work using email, and shows that inferred conversational context is more useful than might be expected. Mona, an operational email system establishes conversation context without explicit user action. Through this automatic approach Mona avoids the twin problems of enhanced email systems: dependency on additional work by users, and inter-system incompatibility. The wider potential of dependency avoidance is also mitigated, since there is no critical mass before Mona's benefits become available.

Background

Groupware Experiences with Email

Message filtering systems were initially developed to improve information management and reduce information overload. Examples include the Information Lens (Malone *et al*, 1988), and ISCREEN (Pollock, 1988) in which 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. More generally, the power of message filtering schemes can be used to support toolkits for the development of coordination applications, examples include the Object Lens (Malone and Lai, 1988) and Strudel (Shepherd *et al*, 1990).

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

gIBIS (Conklin and Begeman, 1988), Strudel (Shepherd *et al*, 1990), WHAT (Hashim, 1991), and SIBYL (Lee, 1990) use variations of the IBIS (Kunz and Rittel, 1970) 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. Through similar explicit statement of message purpose, systems like the Coordinator (Flores *et al*, 1988; Winograd, 1987) take an active role in the process of project management, providing reminders, identifying commitment defaulters.

Dependency on Structure and Guidance

All the applications discussed above require additional information from the user or from messages (other users) to provide their augmented facilities. We call this extra work *guidance*. When sending a message, the user must select an appropriate message type and fill in the associated fields. Some systems additionally require that messages are split to distinguish 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?'

Though additional work of this nature may be acceptable when personal benefits are provided, typically providing guidance results in no direct personal gain; someone else, the receiver(s) get the benefit (Grudin, 1988). It may be *impossible* for some users to supply guidance, particularly when working at different locations without the relevant systems; even when using the same system there may be incompatibilities between message structures (Lee and Malone, 1990). The guidance required by email applications imposes a cost/benefit disparity at one of the following 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.

- If the sender decides that the cost of providing guidance is unwarranted and fails to do so, then, for the benefits to be realised, a third party must execute the actions on the sender's behalf. The consequences of omitted guidance are serious for systems taking an active role in the coordination process. If the additional work is not carried out the system's knowledge of commitment status will become non-current, causing problems such as redundant or mis-timed reminders.
- The sender carries out the relevant actions, but problems such as incompatible systems require a third party to repeat the actions.

Adoption and Critical Mass

Goodman and Abel (1987) state that, 'People will use new communication systems to the extent that they require no more effort than existing ones.' While this may be mitigated by the enhancement of work tasks, there is a vicious circle restricting the realisation of system borne benefits (figure 1). The factors in this chain of dependencies are the level of *benefit* gained from the system, its potential for *adoption by individuals,* and its *achievement of critical mass.* The key determinant in the realisation of each of these factors is the level of *effort* imposed on users.

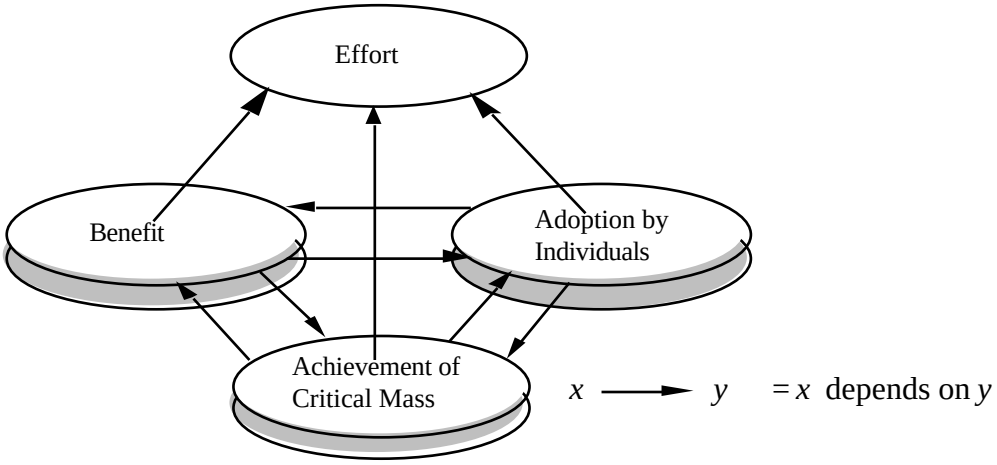

Figure 1: The vicious circle of dependencies in groupware adoption

It is conceivable that groupware applications could have been successful had they achieved the necessary 'critical mass' of users for group benefits to realised. However, failure to provide initial incentive to adopt systems at a personal level, and the effort required to use the system deters people from incorporating new systems into their working methods (for a further discussion see Cockburn and Thimbleby, 1991; Cockburn and Jones, 1992). Mona, the system described in this paper, is designed to both avoid the dependencies in this vicious circle, and provide the zero-cost personal benefit 'kick-start' necessary to overcome it.

Mona: Conversational Context for Free

Mona runs under Unix in the X window system (Scheifler and Gettys, 1986). It provides a graphical user interface to Internet mail conforming to the standard specified in RFC822 (Crocker, 1982) and a variety of independent augmented email facilities, including a hypertext representation of conversations. In contrast to systems providing similar facilities, 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.' Adopting Mona, therefore, causes minimal impact on existing working methods; its augmented management facilities are accessible immediately, but their are no requirements for their use.

Design Considerations in Mona

Mona's provision of conversation facilities follows three design considerations attending to the limitations of previous conversation support applications.

1. *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).
2. *Avoid a Disparity in Cost/Benefit.* Heeding Grudin's (1988) 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).
3. *Allow Flexibility and Personalised Views of Conversations.* The structure of conversations is open to personal interpretation (Romiszowski and Jost, 1990). 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.

The emphasis on *personal* benefit and satisfaction in these design considerations is a consequence of the desire to promote the personal adoption. This is necessary to overcome the vicious circle hindering groupware's ability to overcome 'critical mass'.

Inferring Conversational Context

In inferring conversational context Mona uses RFC822 header information that is independent of user actions and guarantied 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 a knowledge of previous incoming and outgoing mail items contained in a local archive, Mona infers the probable relationships between messages, forming a *web* of conversational relationships (figure 2).

Mona attempts to establish four link types 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 message cause(s) of a message
— **cause**;

the inferred message response(s) to a message
— **response**.

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

$$a_{\textbf{previous by same}} = b \textbf{ when } \textbf{prev\_part}(a,b) \qquad c\textbf{: } \textbf{prev\_part}(a,c) \qquad c_{\textbf{send time}} \leq b_{\textbf{send time}}$$
where
$$\textbf{prev\_part}(a,b) = a_{\textbf{sender}} = b_{\textbf{sender}} \qquad a_{\textbf{send time}} > b_{\textbf{send time}}$$

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

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

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.

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 2). 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' 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:

$$a_{\mathbf{cause}} = b \ \mathbf{when\ conv\_part}(a,b) \quad c\text{: } \mathbf{conv\_part}(a,c) \quad c_{\mathbf{receive\ time}} \leq b_{\mathbf{receive\ time}}$$
$$\mathbf{where}$$
$$\mathbf{conv\_part}(a,b) = a_{\mathbf{sender}} \quad b_{\mathbf{receiver}} \quad b_{\mathbf{sender}} \quad a_{\mathbf{receiver}} \quad b_{\mathbf{receive\ time}} < a_{\mathbf{receive\ time}}$$

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:

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

The number of **cause**s, 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 condition $a_{\mathbf{sender}} \quad b_{\mathbf{receiver}}$ cannot be satisfied when the receiver is a mailing list. 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 **cause**s. 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

Mona's heuristics provide 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 the user will learn to use effectively, despite its limitations in conversational classification.

It can be argued that the best *any* system can provide is no more than a guide to conversation structure — Romiszowski and Jost (1990) observed that individuals tend to differ in their interpretations of conversational context. 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." To allow for discrepancies between interpretation of conversational context, and to further enhance personal satisfaction Mona supports 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 supports selective retrieval of messages satisfying a variety of combined properties. Any message retrieved by these, or other methods can be included in the conversational web. Each node the hypertext 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 `->` symbol) and a separate window displaying the complete message can 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.

Combining inferred conversational context with 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 (Cockburn and Thimbleby, 1991). Mona can 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.

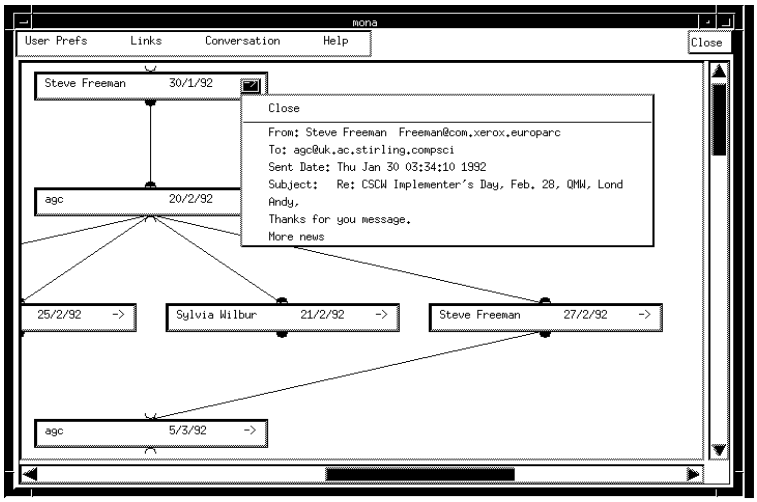

Figure 2. Automatically constructed conversational web.

Limitations of Mona

Mona provides an improved interface to email and several advanced features which augment management of email communications. However, despite these enhancements we consider Mona to be a prototype because of our design decision to limit its functionality in order to better expose the principles it was built to test.[1].

While Mona satisfies the requirements of groupware for completeness, usability and augmentation (Borenstein and Thyberg, 1991), it intentionally disregards benefits available through guidance dependant mechanisms, thus emphasising potential alternatives. Combining augmentation available through guidance-dependant schemes with free inferencing, users would be supplied with compatible and flexible conversation support, allowing social protocols rather than system requirements to mediate styles of use (Dykstra and Carasik, 1991).

Mona's present heuristics are simple and deterministic, allowing users to quickly become familiar with their behaviour — this furthers our aim to encourage system adoption through the provision of immediate benefit. More complex heuristics, though tempting, would most likely fail in obscure ways and in unfamiliar circumstances. Complex heuristics might attempt to predict what the user will do or prefers based on the statistics of past activity. Over a prolonged period, a statistical system may have better performance than a deterministic system such as Mona — but by that time, users may already have rejected the system! 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*

---

[1] Regardless of its prototype status, Mona is reliable and available as C source code from `agc`.

as reliable by the user. In similarity with combined free and guidance dependant approaches described above, the true value of heuristic schemes is likely to be realised when statistical and deterministic heuristics are used in conjunction.

Conclusions

The possible enhancements that groupware offers to email are impressive; they include intelligent filtering of messages, management and coordination of projects, and active assistance in managing commitments. While such facilities are *possible*, their realisation depends on user actions which are frequently omitted due to work pressure, lack of access to compatible tools, lack of motivation, mistakes, even laziness. Consequently, systems are often unable to provide "enhanced facilities" despite effort expended by some users — who then feel cheated. Unlike many systems, since the Mona user expends no effort, no users are ever cheated.

Indeed, Mona exemplifies the free approach to enhancing email as a medium for collaborative and coordinated work. While reducing effort is of obvious benefit to users, Mona's conversational inferencing eases problems arising from systems supporting incompatible information structures, and users failing to provide appropriate information.

Establishing the necessary critical mass of users is a major hurdle for conventional collaboration support tools. Difficulties in convincing users to change their working methods in order to satisfy system requirements is also 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 (even zero). 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 Steve Jones, Tom Kane, and Steve Marsh who commented on early drafts of this paper. This research was funded by the Isle of Man Board of Education.

References

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.

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.

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.

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.

AJG Cockburn and HW Thimbleby. A reflexive perspective of CSCW, ACM SIGCHI Bulletin, 23(3):63–68, July 1991.

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.

DH Crocker. Standard for the format of ARPA Internet text messages; RFC-822, August 1982, ARPANET Working Group Requests for Comments: RFC-822.

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

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.

GO Goodman and MJ Abel. Communication and collaboration: Facilitating cooperative work through communication, Office:Technology and People, 3(2):129–146, 1987.

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.

SH Hashim. WHAT: An argumentative groupware approach for organizing and documenting research activities, Journal of Organizational Computing, 1(3):275–302, 1991.

R Johansen. Groupware: Computer Support for Business Teams, Free Press, New York, 1988.

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.

L Lamport. Time, clocks, and the ordering of events in a distributed system, Communications of the ACM, 21(7):558–565, 1978.

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.

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.

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.

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.

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.

S Pollock. A rule-based message filtering system, ACM Transactions on Office Information Systems, 1(2):233–254, July 1988.

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.

RW Scheifler and J Gettys. The X Window System, ACM Transactions on Graphics, 5(2):79–109, April 1986.

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.

T Winograd. A Language/Action Perspective on the Design of Cooperative Work, Human-Computer Interaction, 3(1):3–30, 1987.