Cooperative Work in the

Andrew Message System

Nathaniel S. Borenstein
Information Technology Center
and Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Chris A. Thyberg
Academic Computing
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The Andrew Message System, a distributed system for multi-media electronic communication, has a number of special features that support cooperative work. After a brief discussion of the system itself, these features are described and discussed in more detail. Examples of how organizations actually use these features are then presented and discussed, with particular attention paid to the "Advisor" system for electronic consulting.

Introduction

The effort to make computer systems better support human collaboration is a many-faceted, ongoing endeavor. Such efforts are always at least partially limited by the available tools. Not only do sophisticated tools (or their absence) define the set of possible experiments that can be conducted, but they also provide a framework for our own thinking about such systems. Thus, any substantial improvement to the existing base of application tools holds the promise of opening up fertile ground, both for new approaches to, and for empirical studies of, human collaboration.

The Andrew Message System (AMS) appears to represent just such a substantial improvement. It extends the facilities of previous electronic mail and bulletin board (bboard) systems in several directions relevant to cooperative work. This paper is an initial explanation of how the AMS has been used, in its early deployment, to facilitate cooperative work.

Background: Andrew & Its Message System

The Andrew Project [1, 2] is a collaborative effort of IBM and Carnegie Mellon University. The goal of the Andrew project is to provide a good environment for university computing. That is, particular emphasis is paid to the needs of the academic and research communities.

As the project evolves, it has concentrated on three main parts. The Andrew File System [3,4] is a distributed network file system designed to provide the illusion of a uniform central UNIX file system for a very large network (10,000 workstations was the design goal). The Andrew Toolkit [5] is a window-system-independent programming library to support the development of user interface software. It currently supports a number of applications, including a multi-media editor that allows seamless editing of text, various kinds of graphics, and animations.

The third main piece of Andrew is the Andrew Message System, or AMS. The AMS, which makes heavy use of the file system and the toolkit, provides a very large-scale mail and bulletin board system. It transparently supports messages which include text, pictures, animations, spreadsheets, equations, and hierarchical drawings, while also supporting "old-fashioned" text-only communication with low-end machines such as IBM PC’s and with the rest of the electronic mail world. The Andrew Message System has only recently become widely available; the "results" discussed in this paper are really observations of the first large test installation, the Carnegie Mellon campus, where thousands of students, faculty, and staff have been using the system during its development over the last few years.

A detailed description of the Andrew Message System is beyond the scope of this paper and can be found elsewhere [6, 7]. This paper will concentrate on those parts of the system that are of particular relevance to issues of cooperative work.

AMS Features for Cooperative Work

Multi-Media Objects

When the Andrew Toolkit and Message System were designed, the desirability of multi-media objects seemed clear, and indeed experience has borne this out, inasmuch as the multi-media features seem to be widely used and appreciated. However, the effects of the multi-media functionality on the way people actually communicated were largely unanticipated.

When the multi-media features were first introduced, they were introduced as upgrades to an existing system; most of the Andrew Message System had been in place for over a year, and the long-promised introduction of the ability to send graphics and animation through the mail didn't seem to make much of a splash on the campus. Indeed, the question of "why don't people make more use of the ability to send pictures through the mail?" seemed for a while to be something requiring serious investigation.

What actually seemed to happen, however, was that a gradual raising of consciousness took place after the introduction of the new functionality. For example, it was months after these capabilities existed that someone thought to include, in his suggestion for changes to a user interface program, a line drawing of what he thought the interface should look like. Similarly, it seemed a major conceptual breakthrough when, instead of describing a bug, a user simply took a snapshot (screen dump) of his screen when the bug was clearly visible, and included the image of the screen as part of his bug report message (Figures 1-2).

Thus it seems that a picture is only worth a thousand words when people are fully aware that pictures are an option. Now that such awareness is more common, illustrations are becoming less of a novelty and are increasingly relied on to clarify prose, and particularly to speed the diagnosis of bug reports.

In addition, animations, initially perceived merely as a novelty when included in mail messages, have also proven to be useful for clarifications. For example, animations have been used on several occasions to clarify a convoluted prose description of a complex *process*. While it is difficult to include a proper animation example in an article to be published on paper, Figures 3-5 give some frames from such an animation.

Magazines

One special feature of the Andrew Message System that is itself an example of cooperative work is the electronic *magazine*. Magazines are user-edited, publicly-readable bulletin boards. For example, a user with a strong interest in music might read a dozen or so music-related bboards. (If this sounds excessive, it might help to know that at Carnegie Mellon there are currently over 1700 bboards in the public tree, and that some individuals subscribe to nearly 400 of them.) If he has volunteered to edit a "music magazine" he can, as he reads these bboards, simply cross-post the best messages onto his magazine with a single keystroke or menu selection. Those less interested in wading through the masses of information on the other bboards can instead peruse his magazine alone.

As it turns out, this mechanism is not merely an example of cooperative work, but strongly supports other cooperative efforts. One member of the AMS Group, for example, reads approximately a dozen bboards of strong relevance to electronic communication. The other members of the group do not read those bboards, but instead read an electronic mail magazine prepared by the person who does read them all. In this way, the system makes it easy for one member of a group to serve as an information filter for the other members. There are currently over 30 such electronic magazines on the Andrew system at Carnegie Mellon, and a few of them regularly appear on the "top 40" list of the most widely-read bboards on the system. For these particularly popular magazines, over a hundred people are choosing to read the magazine instead of the source bboards from which the magazine’s contents are derived.

Private Bulletin Boards and New Bulletin Board Creation

The Andrew Message System supports a rich and flexible set of protection and configuration options that facilitate group communication. In particular, the protection mechanisms permit the creation of public bboards, private bboards (readable & postable only by members of a group), official bboards (readable by all, postable only by a few), administrative and advisory bboards (postable by all, readable by only a few), and various hybrids thereof. In addition, the protection mechanisms can be (and are) used to allow, for example, a secretary to read and process someone else’s electronic mail. (Indeed, a secretary could create something like a magazine for his employer, containing only those pieces of his mail that he thought his employer would really want to see.)

The system is also configurable to specify the control of the creation of new bboards. Since the bboard database is structured as a tree, one can specify, for any point in the tree, whether some or all users are allowed to created sub-nodes in that tree. On the Carnegie Mellon campus, the system has been liberally configured in most parts of the tree, to allow any users to create new bboards when they so desire. Although this has created occasional annoyances that needed to be cleaned up by the system administrators (notably when two people created similar but differently-named bboards, or when people frivolously created nuisance bboards in inappropriate locations), these rare annoyances have been more than offset by the freedom it has given to the participants in group discussions. In general, the users seem to have reasonably clear perception of when it is appropriate to start a new bboard, and the knowledge that they can do so without troubling overworked system administrators encourages them to do so quite often. In fact, the proliferation of bboards has led to the creation of an automatic "bboard bboard" which reports daily the names of all the bboards that have been created or deleted in the last 24 hours, and this bboard is itself subscribed to by dozens of users.

Active Messages

Another aspect of the AMS that is relevant to cooperative work is its support for *active messages*. Active messages are messages which, when read, prompt a message-specific interaction with the user. The "active message" feature was not designed as a feature in its own right; instead, the concept of active messages has evolved out of several specific types of active messages that the system now supports. A worthwhile goal for future research and implementation is to generalize this notion to enhance its utility.

In the Andrew Message System now, there are four specific types of active messages, each of which will be described briefly here.

Folder Announcements are messages that invite subscriptions to a new bulletin board. For example, if there is a bboard called "*andrew.gripes*" and someone creates a new bboard called "*andrew.gripes.angry*," the system will automatically post a folder announcement message on andrew.gripes. Anyone who subscribes to andrew.gripes will see the message, which will describe the new bboard and show the first message on that bboard, and then ask the user whether or not he wishes to subscribe. In addition to these automatically-generated folder announcements, users may themselves easily post folder announcement messages whenever they deem it appropriate.

Vote Messages are messages that put a question to a vote. The reader of the message can see the text of the message, typically an explanation of the question being voted on, and can then choose from a multiple-choice option list. The creator of the vote determines the vote choices, and whether or not write-in votes are to be permitted. The reader of the message always has the option of not voting. In order to prevent undetectable "ballot box stuffing," all votes are subject to the full authentication of the AMS Message Delivery System, so that votes are about as far from secret (anonymous) balloting as could be imagined.

Vote messages seem to have great utility in settling arguments. Often it is impossible to tell from normal bboards what the majority opinion is, because minorities are so often quite vocal. A simple voting mechanism has proven useful in moving discussions beyond stagnant debate. For example, it was used to finally settle two of the most burning issues in the Andrew development project, the question of bottled water versus tap water and the question of which brands of pop should be stocked in the pop machine. (See Figures 6-7.) Settling less, but perhaps more interesting, the university's Public Relations department has been using the voting mechanism to conduct informal surveys of campus opinion. (See Figures 8-9).

In addition, the vote mechanism has been used in at least two ways that the developers of the system did *not* expect. One way that it is used is to facilitate quick answers to questions. One user recently sent out a message asking for advice in choosing between several ways to work around a bug (Figure 10). By making the message call for a vote, he made it particularly easy for the recipients of his mail to express their opinions immediately. Another unexpected use of the vote mechanism has been to play a word game known as "dictionary," in which a moderator chooses an obscure word, each participant submits a fake "definition," and then everyone has to try to choose the correct definition from among the fakes (Figures 11-12).

Return-Receipt Messages are simply messages that are marked as requesting confirmation. When the user actually reads the message, he will be asked if he is willing to send an acknowledgment to the sender. By answering positively, he causes a confirmation to be sent automatically. This mechanism is a clear analogue to those provided by physical mail and is similarly useful. In particular, the mechanism is used regularly by people who communicate across distant network connections that tend to lose mail.

Enclosure Messages are messages that enclose additional data. For example, two users at remote sites may have no way to exchange data except via mail. Using enclosures, they can easily separate the message headers and body from the object being enclosed. When a user receives an enclosure message, he is given options for processing it, such as writing the enclosure (not the full message) to a file or running it through a filtering program. This mechanism streamlines some of the processes involved in cooperative efforts by remote users. For example, two co-authors of a paper who exchange drafts via mail typically, in existing systems, have to edit out all of the mail headers each time they receive a draft from a co-author. Using enclosures, the active nature of the message causes the same editing to be performed essentially automatically.

Extension Mechanisms

The Andrew Message System provides several types of extension mechanisms, each of which has been relied on heavily by those who use the system to coordinate group activities. The extension mechanisms that have proven most useful in the AMS to date will be described here briefly.

Automatic filtering of incoming messages has proven to be immensely valuable. The AMS provides a language for writing specifications of what will happen to new pieces of mail -- that is, where they will be placed for user viewing. For example, messages with certain key words can be routed directly to appropriate personnel for handling them. This kind of functionality has been particularly useful for advisory and bug-handling organizations, as described later in this paper, and for the support of an extensive bulletin board system. (It should, in fact, be noted that this mechanism has proven so useful that has recently been rewritten in a more generalized and powerful way, including an embedded LISP interpreter and support for running other programs from within the classification program.)

Boilerplate message templates are useful for processing routine requests for information or action. It is straightforward in the AMS to create a draft message that is the basic answer to a common request, and to then bring up that draft with a single keystroke or menu selection.

Compound commands are a mechanism for reducing the number of actions to be taken in a common situation. For example, using compound commands, one can easily create a single menu item which, when selected, will take the message currently being composed, add a few designated header lines to it, send it to its designated recipients, and place a copy on a private bboard somewhere. It has been the authors' experience that this is so useful that users will put up with an amazingly ugly syntax for designing the commands. Nonetheless, a more friendly syntax and mechanism for this purpose is being planned.

How the AMS Is Used In Real Cooperative Efforts

The Andrew Message System has proven to be exceptionally popular with its user community in general. Weekly statistics indicate that nearly 2000 people use it at Carnegie Mellon to read bulletin boards regularly. Several thousand Andrew users read their personal mail with the system. The AMS is also in use at several other universities and research sites. This would be indication enough that system is a success. However, the greatest enthusiasm has in fact been found among those who are using the AMS for substantial cooperative activity. Most notable among these devoted users are the people who provide support services on Andrew. The Andrew Advisor is a singular example of real-life cooperative work, conducted with the Andrew Message System.

The Advisor System

Campus Context

Carnegie Mellon and the Andrew project face some unique problems in supporting its computing constituency Three factors contribute to the challenge. First, although the system has been widely deployed and promoted, Andrew is an ever developing, rapidly changing environment. Second, campus computing expertise is widely, but unevenly, distributed. The users span the entire spectrum from technophobe to technophile. Third, the cast of characters involved in fixing bugs, adding new features, providing system administration, and answering users' questions is diverse and distributed among several organizations and buildings. The following organizational schema gives a sense of the range of people that might be involved in handling any particular user request:

Information Technology Center: Developers who create, document, and distribute the Andrew system.

Andrew/Unix Development: Systems programmers who maintain and support UNIX and Andrew on several machine types.

Andrew Systems Administration: Systems administrators and operators who run the Andrew file servers and other essential central services.

Networking and Communications:- Programmers and technicians who maintain the complex campus network.

Academic Computing: Professional and student consultants who provide technical support, public computing facilities, documentation, training, and publications, for Andrew and other computing systems, to the entire campus community.

Andrew Support Group: A sub-group of Academic Computing specializing in Andrew.

To cope with this complexity, members of the Andrew Support Group (ASG), with the help of the AMS group, have developed an extensive electronic mail consulting service called "Advisor." Advisor is designed to create the illusion of a single, private, and personal help resource for every conceivable Andrew problem. The user simply mails a query to Advisor’s account. In 24-48 hours private mail comes back to the user. In fact, however, Advisor is the front-end of a vast network of bboards that enlist the cooperative efforts of all the professional staffs in the organizations listed above.

Ancient Advisor History

Advisor has been in use since January, 1985. In the earliest days, it was simply another Andrew account. One person read the incoming mail, handled it with whatever limited tools were available (mostly paper lists and a good memory for the status of a given request), and sent out a reply to the user. This worked reasonably well when Andrew was in pilot deployment to about 100 carefully selected users and the Andrew consultant had an office in the Information Technology Center.

In the spring of 1986, the first Andrew cluster opened and Andrew accounts were available to the campus. Immediately, Advisor was overwhelmed with mail. An additional consultant picked up Advisor duties, but there were always problems with how to divide the work between the two staff members and how to keep track of the status of any given message. Classifying messages was possible, but the mechanism was extraordinarily clumsy, labor-intensive, and not too useful, because all the messages were still lumped together in one big flat mail directory. The combination of the large volume of the easy questions and the genuine difficulty of the hard questions made it tough to process Advisor mail in a timely fashion. The staff clearly required some efficient method of getting almost immediate assistance from the right people in the other Andrew groups.

In the fall of 1986, the first version of what is now the Andrew Message System was released to campus. Though conceptually distinct, personal mail and bboards were no longer different in kind. One's private mailbox and a public bboard were both examples of message databases, albeit with different levels of protection. Furthermore, since messages databases were built on top of the Unix hierarchical directory structure, bboards could now be nested within each other. This "paradigm shift" made it possible to think of using bboards as folders for classifying Advisor's "personal" mail. Armed with a suite of semi-private bboards (postable by the whole community, but readable only by those in the Andrew organizations) and an extremely primitive stack-oriented language for automatically filing messages, the Advisor staff set out to do things differently.

Advisor I

Tom Malone, in his discussion of the Information Lens system [8], identifies three fundamental approaches to handling the twin problems of being overwhelmed by useless electronic junk mail, and yet frequently being unaware of vital information available only electronically. The first approach, which he calls *cognitive filtering*, attempts to characterize the contents of a message and the information needs of the recipient. The system uses these profiles to match messages about XYZ with readers who have expressed an interest in XYZ. The second approach, which he calls *social filtering*, focuses on organizational relationships between the sender and the recipient. In addition to the message's topic, the status of the sender plays a role in the reader's interest in the message. The final approach, which he calls *economic filtering*, looks at implicit cost-benefit analyses that come into play when one decides what to do with a piece of electronic mail. The first incarnation of the Advisor system relied very heavily on both cognitive and social filters. Automatic message classification was the primary implementation technique.

Each message to Advisor that did not come from a member of a known set of Advisor "helpers" was judged to be from a user needing help. The message was then placed on a bboard called "*advisor.open*." The Advisor staff subscribed to this bboard and used it as an inbox for new questions. A copy of mail from the user was also placed in *advisor.trail*, to assist the staff in keeping track of requests. Thus, the first criterion for sorting the mail was a social one - is the sender a helper or a user?

An incoming question from a user was also copied to one of a series of subject-specific bboards, according to keywords in the subject line. For example, if a subject line was "mail bug," the message was copied to *advisor.mail*. These bboards, though not open to the public, were readable by the developers, system administrators, etc., who subscribed to whichever bboards covered their areas of interest and responsibility. To continue the example, the AMS group members subscribed to *advisor.mail*, thereby increasing the likelihood of seeing only those messages generally relevant to them. Uninformative or nonexistent subject lines caused the message to be copied to *advisor.misc*. All good Advisor helpers were expected to subscribe to advisor.misc, in addition to their other subscriptions. Here one finds a clear example of cognitive filtering.

Cognitive and social filtering were combined at several critical junctures. For example, when the Advisor staff requested more information from the user, Advisor received a blind carbon copy of that request. Because the message was from Advisor, it did not go into advisor.open. Instead it went to advisor.trail and to the relevant subject-specific bboard. Another example was in the processing of contributions from Advisor helpers. A helper would see a question on some topical bboard. By choosing the "Reply to Readers" option (which prepends "Re:" to the same subject line as the user’s initial post), the helper sent the answer, not to the user, but directly back to that subject-specific bboard. Mail from helpers never went into advisor.open, but only to some topic-oriented bboard. And when a final answer was sent to the user, the blind carbon receipt once again bypassed advisor.open and ended up on advisor.trail and the correct topical bboard. The Advisor staff member would remove the question from advisor.open and append to it the copy of the answer. These question-answer pairs went to an *advisor.qa* bboard, which acted as a repository of useful past work.

To summarize: the Advisor staff answered questions from advisor.open as they were able. They kept an eye on the relevant subject-specific bboards for help with the difficult problems. Having collected the information from the helpers, the Advisors sent polished answers back to the users. As far as the users could see, they had sent mail to Advisor and received an answer from Advisor. The fact that there was additional internal consultation was kept behind the scenes.

Other Consulting Venues

Though Advisor is designed to preserve the semi-confidentiality of a user’s mail, the Andrew Support Group also explored publicly readable bboards for asking questions and receiving help. The staff thought that there were many kinds of queries that a user would be willing to put forward in public. If other users could see these questions asked and answered, they might not need quite so much assistance from Advisor. Two bboards were created: *andrew.hints* and *andrew.gripes*. The former is a free-market of user-contributed advice. It serves that purpose reasonably well. The latter was intended as an official information channel parallel to Advisor mail. That is, the Advisor staff would monitor andrew.gripes, get correct answers and post them back to the user and to the bboard. Though this was the clearly stated intention of the bboard (the welcoming subscription post spelled it out in plain English), within a week of its inception, andrew.gripes had become a free-for-all of misinformation, *ad hominem* argument, and general rudeness. The Andrew Support Group has given up treating andrew.gripes as a service responsibility, though they monitor it for questions about Academic Computing policy. Andrew.gripes continues to be a heavily subscribed and apparently popular forum for the hackers and the developers.

Evaluation of Advisor I

The key feature of the first Advisor mechanism was the automatic filing of messages into subject-specific bboards. The positive effect of this was two-fold. First, messages came to the immediate attention of the other technical groups. Often, the Advisor staff found that someone in another group had already answered the question before Advisor had even looked at it. This kind of proactive assistance was extremely appreciated. Second, because requests for more information and final answers passed back to the subject-specific bboards, the other groups could provide problem-solving advice and assure technical accuracy.

However, the negative effects outweighed the positive. First, poorly phrased questions from the users led to many "misclassifications." The message filing algorithm worked quite well, but so many subject lines were virtually contentless, e.g., "Help!," that far too many messages ended up on *advisor.misc*. The authors estimate that close to fifty percent of all mail to Advisor was filed into *advisor.misc*. Without better characterization of the message's content in the subject line, the Advisor staff were helpless to get the right mail to the right parties. The Advisor designers considered the possibility of also searching the body of a message for sort keys, but the filtering language was not powerful enough to support free-text information retrieval techniques. Advisor settled for pattern matching on the subject line, rather than suffer too many false keyword hits.

Second, with every question going to a subject-specific bboard, the Advisor helpers had no easy way to distinguish between the questions the Advisor staff knew how to answer and those they didn't. Hence, they wasted time answering some questions unnecessarily and neglected other questions for which help really was required. In retrospect it seems like a truism, but actual use of the mechanism vividly showed that cooperative work disintegrates if what is expected and from whom are not clearly articulated. Electronic methods only exacerbate the problem of undefined expectations.

Third, because every blind carbon from Advisor and every message from an Advisor helper also went to the subject-specific bboards, these soon got too cluttered to be of much use. On the one hand, helpers got tired of wading through them. On the other hand, Advisor had no way to show a message and all the replies to it in a single chain, so it was sometimes very hard to find the answers that were already available. There is nothing so deadly to cooperation as seeming to ignore another's efforts. Despite Advisor's best intentions, this problem appeared far too often.

Advisor II

In the current version of the Advisor system, the only *automatic* sorting of incoming mail is by the day it arrives. Mail goes into one of *advisor.inbox.monday*, *.tuesday*, etc. Student Advisors are each responsible for a particular day's worth of Advisor mail. They handle all that they can -- which is most of the messages -- and then cross-post the tough questions on topic-oriented bboards with names like "*advisor.helpbox.mail*." These "helpboxes" are very similar to the "magazines" described above -- they are, in fact, magazines compiled by the Advisor staff of just those questions that require the help of some other group to answer. The technical staffs subscribe to appropriate helpboxes and to the parent bboard, *advisor.helpbox*. Posts to the parent bboard notify Advisor helpers of the creation of a new helpbox, give a synopsis of its purpose, and invite them to subscribe. All this is done automatically, via the folder announcements, as described above.

In addition to the helpboxes, there are *advisor.questions* and *advisor.trail*, for rudimentary measurement and tracking, *advisor.outbox*, for question-answer pairs, and *advisor.discuss*, for meta-Advisor debate and general Advisor information. (Figure 13). Some people in other organizations subscribe to these ancillary bboards to get a sense of how things "feel" in the Andrew world. Advisor welcomes such observers, especially to the extent that they are able to influence resource allocation in behalf of Advisor’s work.

Evaluation of Advisor II

By putting human intelligence to work at the heart of the system, the Advisor designers solved in one stroke several of the problems mentioned above. First, Advisor can support a far more fine-grained suite of helpboxes than it could with automatic filing. Poorly phrased subject lines are less of a concern because humans read the mail and digest its contents before passing it to a topical bboard. Second, when an Advisor staff member puts a question on a helpbox bboard, everyone knows that this means that help is genuinely needed. Third, because clutter does not automatically accumulate in the helpboxes, these have become "high-content" bboards that the programmers and administrators feel are worth reading regularly. The payoff for Advisor is a much more reliable information resource. And just in case there are a number of items pending on a given helpbox, the Advisor now has a "Show Related Messages" option which puts a marker beside all the messages in a given reply-chain. Advisor rarely misses a helper’s contribution in the new scheme.

In summary, though the new scheme lacks the proactive help and the quality assurance that was evident in Advisor I, the Advisor staff feels that they are better equipped to handle the load than before. Currently, Advisor receives an average of 30 messages each day. Note that these are new requests from users; the total number of messages that pass through the Advisor system, including help from Advisor helpers, requests for more information, and replies to users is close to 100 messages per day. The student Advisors do an admirable job of performing triage on incoming mail. Fulltime consultants now function much more as Advisor supervisors, taking areas of technical responsibility, expediting helpbox requests, and insuring that the answers that go out from Advisor are timely and accurate. In sum, messages now filter up "manually" through different levels of expertise: the simplest questions are answered by the students, the harder ones are answered by the fulltime consultants, and the hardest are tackled by the programmers and administrators themselves. At each level, humans must work diligently and efficiently to minimize time-delays inherent in the system. But all parties involved feel that the Advisor scheme focuses and streamlines their efforts.

Advisor’s Future

The designers of Advisor have made a tactical retreat from the more automated porcessing of Advisor I. To begin with, they have had to solve new problems which required them to use other tools in the AMS kit. For example, sorting Advisor mail by day creates the problem of how Monday’s Advisor continues a dialog with a user on Tuesday, without getting in the way of the Tuesday Advisor. This problem is solved by the use of compound commands, as described earlier. Advisor now has a suite of customized message sending/replying commands, one for each day of the week. (Figure 14). These commands, which are on the menu and bound to keys, insert a special message header on the outgoing mail. That mail, and all mail in reply to it, get sorted into the correct day’s inbox by virtue of that header. So even though the followup reply from the user comes in on Wednesday, it still goes to the Monday inbox, where Monday’s Advisor is waiting for it. The ASG is capturing other complicated, but repetitive, Advisor actions as compound commands.

The Andrew Support Group has also begun to connect the Advisor system to other help groups on campus. The most mature example to date is a bridge between the *advisor.helpbox.datacomm* bboard and a suite of bboards attached to a special userid, dc0m, belonging to the Network and Communications group. Rather than have these folks subscribe to the Advisor helpbox as a second source of input to their group, the Advisor designers created a "hot link" between the two groups. When Advisor puts mail into its datacomm helpbox, it is automatically resent to dc0m with a special header. When someone in Data Communications replies to that mail, by virtue of that header, it comes back directly to Advisor's helpbox, just where the Advisor expects to find it. There are similar links to the group that handles public cluster issues and to a Student Advisory Committee for policy matters. The ASG plans to provide additional hot links to Advisor-like systems that they've already exported for academic use. In this way, the ASG hopes to help these groups become largely support themselves, while still providing a fast channel by which their support staff can communicate with the Advisor staff. It is our belief that a large part of Academic Computing's future role is to enable distributed support.

Finally, Advisor still handles a huge load of routine items like requests for more disk quota. These are matters that rarely require attention from the Advisor staff, save to pass them along to a systems administrator with an acknowledgment of receipt. It would be nice if the students did not have to do very much to handle such requests. Thus, the plans for Advisor include judiciously reintroducing certain automatic mail forwarding features. However, before doing so, the ASG plans to explore two areas of development, the results of which should fend off the problems encountered in the days of Advisor I.

The first work item is to build of a suite of Advisor-request templates. Advisor has long used boilerplates for sending answers back to the users. But rather than leaving it wholly up to the user what his mail to Advisor should look like, Advisor will also provide various forms that are pre-addressed, pre-titled, and internally organized into fields, some of which have the content dynamically supplied. The possibilities are almost endless for creating highly sophisticated forms using multi-media objects.

Recently, the AMS Group made available a much more powerful and easy-to-use extension language called "FLAMES" (Filtering Language for the Andrew MEssage System) and a set of common extensions in the FLAMES library. The language is essentially Common LISP, with special primitives for manipulating the AMS database. Thus, the second work item is to acquire FLAMES expertise in the ASG. FLAMES will make it possible to create very powerful mail-handling routines that process the fields in the various Advisor-request templates. With semi-structured input from the users and LISP-based filters to process it, the future for automatic message handling looks promising again.

For example, by the time this paper is published, the ASG expects to have a prototype quota-request form. This form will have an appropriate subject line, and will contain dynamically-generated information about current disk use, a stock message from Advisor about current policies for quota, and some fields to complete in that elicit the reasons for the increase. When Advisor gets this mail, a canned acknowledgment will go to the user automatically, and the form will be sent to the person who handles quota increases. The message will now have an additional header so that both the user and Advisor are notified when the user’s quota has been increased, Advisor is also notified. The request and the receipt can then be saved, if statistics on disk quota request handling are desired, or it can be deleted. The Advisor developers hope to follow this prototype with templates and parsers for bug reports, requests for new features, and the like.

Another area of development for the ASG is to continue work they’ve begun on non-workstation interfaces to Advisor. Student Advisors are particularly eager to do their advising from low-end machines in their rooms. Currently the group has one sample interface, using an unsupported package that runs under the Emacs text editor. Of course, there are several AMS interfaces that run on low-end machines, but only the flagship Messages interface, on workstations, has all the customization features that Advisor now depends on.

Finally, it should be noted that the Advisor staff, who do not view themselves as "hackers", are nonetheless able to develop customized compound commands, hot links between support systems, Advisor-templates, and alternative interfaces independently, in large measure, of the AMS developers. This seems to be clear evidence of the maturity, power, and flexibility of the Andrew Message System.

Academic Uses

The Andrew Message System is heavily used by academic courses at Carnegie Mellon. As of the spring of 1988, there were over 100 academic bboards in use by more than a dozen different departments, including relatively non-computerized departments such as English, History, and Architecture. The extent and nature of its use varies substantially from one class to another. In some classes, it is used simply to post assignments and other "official" notices. In other classes, however, substantial portions of the class discussion takes place on the class bboards (Figure 15). In a few classes, a significant portion of the grade has been based on bboard participation.

In addition, it is becoming increasingly common for classes to take advantage of the protection mechanisms to create several different types of bulletin boards. Quite a few courses now have an "admin" bboard, which any student can post to but only the teachers and teaching assistants can read. This provides an easy way for students to contest grades, ask questions without fear of looking "stupid" in front of their peers, etc., and without requiring an office appointment with the professor himself. As such, the mechanism is popular among the teaching staff as a time-saving feature. In at least one case, private bboards have been used within an academic course, as competing teams in a Software Engineering course discussed their designs independently, using both private per-team bboards and larger whole-class bboards, whichever seemed more appropriate for a given discussion (Figure 16).

Two particular instances of course-related bboards warrant detailed description. The first is a system for supporting the two-semester sequence, "Fundamental Structures of Computer Science, I and II."

Approximately 300 students per semester are enrolled in one or the other course. In addition, the classes are taught on the Andrew system. To provide better service for the students, the ASG, with the cooperation of the instructors, teaching assistants, and the AMS group, created a suite of bboards for the two classes. The initial bboards created were:

academic.cs.211: The "root" bulletin board for the class, on which sub-bboards were announced automatically.

academic.cs.211.announce: A bulletin board which the students could read but only the staff could post on, used to announce due dates, changes to assignments, and other important news.

academic.cs.211.help: An open bboard for students and course staff, where students were encouraged to post questions about the course or its use of Andrew facilities. Students were encouraged to assist each other on this bboard, so that an answer might be provided either by the staff or by another student.

academic.cs.211.discuss: Another open bboard, for discussion of technical issues germane to the *content* of the course.

academic.cs.211.admin: A private bboard, readable only by the course staff. Students could post on this bboard in confidence simply by sending mail to "CS-211"

The teaching staff went on to create two additional bboards, *academic.cs.211.admin.handled*, for internal tracking, and *academic.cs.211.discuss.grades*, "for people who wish to complain [publicly] about the grading system of the homework, tests, etc."

The second example is an Advisor-like system for the Computer Skills Workshop (CSW). Each fall, most of the 1200 or so entering students take CSW as their introduction to the computing facilities at Carnegie Mellon. As part of the course, all students subscribe to *academic.csw*, the top level bboard for the course.

In the fall of 1987, the Advisor staff tried an experiment. They created another Advisor system, called "CSW-Advisor" solely for the support of the CSW class. (Historical note: CSW-Advisor was actually a beta-test of Advisor II. Much of the organization of the current Advisor was tried out here.) CSW students were told to address questions about Andrew to CSW-Advisor. In case they forgot and sent mail to Advisor, the Advisor system trapped their mail and sent it over to CSW-Advisor. People who were neither CSW, nor part of the CSW staff, nor CSW-helpers had their mail to CSW-Advisor automatically routed over to Advisor. An initial suite of private bboards under "csw-advisor" included bboards for discussions among the course staff, an inbox and outbox for tracking help messages from students, and a suite of "help" bboards much like the current advisor "helpboxes".

In the spring semester, the CSW students got into the act and created a number of public bboards under academic.csw, dedicated to topics ranging from the "CMU work ethic," to underage drinking on campus, to the desirability of DOD-funded research institutes on campus.

Conclusions And Future Work

The developers of the Andrew Message System did not set out to investigate tools for cooperative work *per se*, but simply to build a better system for electronic communication. However, they gradually found themselves drawn in to the issues germane to electronically mediated group work. Much of the evolution of the system has been driven by the requirements of the user groups described above. An even larger part of the future plans for the AMS are geared towards supporting such work.

The real hope is that providing a higher level of functionality in a widely available message system will further raise the level of consciousness and expectations regarding electronic communication in general. One quickly gets used to the kinds of features the AMS provides, however surprising and delightful they might seem at first, and it seems inevitable that people will, for example, come to regard integrated graphical objects as a basic and necessary part of electronic mail. Once more people have done so, the next step in the future of electronic communication may be easier to discern.

Acknowledgments

The Andrew Message System was designed and implemented by Nathaniel Borenstein, Craig Everhart, and Jonathan Rosenberg. Substantial additional work was done by Adam Stoller, Mark Chance, Bob Glickstein, Sue Pawlowski, and Aaron Wohl. The Advisor system was developed by Chris Thyberg, Pierette Maniago, Wallace Colyer, and a staff of student Advisors. Andrew is a big project, and the AMS could not exist without the work of dozens of people who cannot be named here due to space limitations. We would, however, be remiss if we did not acknowledge the unique vision of Jim Morris, without whom Andrew as we know it could never have become a reality.

Special thanks are also due to Chris Haas, Jim Morris, and Adam Stoller for their comments on earlier drafts of this paper.

References

[1] Morris, et al., "Andrew: A Distributed Personal Computing Environment", C*ommunications of the ACM*, March, 1986.

[2] Morris, James H., "'Make or Take' Decisions in Andrew", *Proceedings of the USENIX Technical Conference*, February, 1988.

[3] Howard, John. H., "An Overview of the Andrew File System", *Proceedings of the USENIX Technical Conference*, February, 1988.

[4] Kazar, Michael Leon, "Synchronization and Caching Issues in the Andrew File System", *Proceedings of the USENIX Technical Conference*, February, 1988.

[5] Palay, et al., "The Andrew Toolkit: an Overview", *Proceedings of the USENIX Technical Conference*, February, 1988.

[6] Rosenberg, et al., "An Overview of the Andrew Message System", *Proceedings of SIGCOMM '87 Workshop*, Frontiers in Computer Communications Technology, Stowe, Vermont, August, 1987.

[7] Borenstein, et al., "A Multi-media Message System for Andrew", *Proceedings of the USENIX Technical Conference*, February, 1988.

[8] Malone, et al., "Intelligent Information-Sharing Systems", *Communications of the ACM*, May, 1987.