MailCall: Message Presentation and Navigation in a Nonvisual Environment

Matt Marx and Chris Schmandt
groucho@media.mit.edu, geek@media.mit.edu
MIT Media Lab
E15-252, 20 Ames St.
Cambridge, MA 02139
+1 617-253-5156

ABSTRACT
MailCall is a telephone-based messaging system using speech recognition and synthesis. Its nonvisual interaction approaches the usability of visual systems through a combination of message categorization, presentation, and random access. MailCall monitors conversational context in order to improve feedback, error-correction, and help. Studies suggest that its nonvisual approach to handling messages is especially effective when the user has a large number of messages.

KEYWORDS: Auditory I/O, interaction design, mobile computing, speech recognition, speech interface design.

INTRODUCTION
Several telephone-based messaging systems, research and commercial alike [3, 5, 6, 8], demonstrate the utility of speech for mobile messaging. A common obstacle for such systems, however, is the audio-only modality of the telephone. The slow, serial, and transient nature of speech [7] make stepping through long lists of messages to find relevant ones not only frustrating but downright expensive if calling long-distance or from a cellular phone.

An effective messaging system must support rapid access by giving an overview of the information space and supporting random access to individual messages. Since these interactional demands exceed the capabilities of touch-tone input, speech recognition must be incorporated. Hence, the system must wrestle with additional issues regarding helping users learn the system vocabulary as well as handling recognition errors.

This paper describes the design of MailCall, a telephone-based messaging system which strives to support in a nonvisual environment the kinds of browsing and selection available in visual messaging systems. MailCall categorizes incoming messages by importance, provides summaries of categories, and allows random access. It also strives to meet user expectations of conversational interaction by offering context-sensitive help, tracking recognition errors, and customizing feedback according to conversational context.

RELATED WORK
Although research on telephone-based messaging systems dates back more than ten years [5], several of the above issues are well-motivated by the Phoneshell system [6]. Phoneshell offers telephone-based access via touch-tones to much of the information available on the desktop, including electronic mail, voice mail, rolodex, calendar, news, weather, and traffic. Since many users received several dozen messages a day, Phoneshell supported rule-based filtering based in order to group messages into categories such as “important” and “mass mailings.” The limitations of touch-tone input restricted the user principally to sequential navigation—reading the next message or the previous one.

Chatter [3] attempted to render the messaging functionality of Phoneshell using speech recognition. It allowed the user to retrieve email messages, send voice messages, look up information in a personal rolodex, place outgoing calls, and ask the location of other Chatter users. It presented messages in order of relevance based on the user’s past behavior using memory-based reasoning. Still, users were limited to sequential navigation. Chatter used a sophisticated discourse model to track the conversation, although it did not handle recognition errors.

The SpeechActs project [8] combined the conversational style of Chatter with the broad functionality of Phoneshell, offering a speech interface to several applications: mail, calendar, stock quotes, weather, and scheduling reminders. SpeechActs offered rule-based filtering similar to that of Phoneshell and improved upon sequential navigation by allowing the user to access messages by number (e.g., “read message 17”). Recognition errors were addressed in SpeechActs, which explicitly verified requests which were irreversible (e.g., “delete message”) and offered progressively more detailed assistance when the user experienced difficulty in formulating a command which the speech recognizer could understand.

The Wildfire Assistant is a commercial system which attempts to replace a secretary: it serves as an interactive

voice mail system, allows the user to retrieve voice (but not text) messages, schedules reminder notes, and provides information about system users' whereabouts. Incoming telephone calls can be screened or forwarded. Wildfire does not provide a summary of incoming messages but does allow the user to ask if there are messages from system users.

Phoneshell, Chatter, SpeechActs, and Wildfire all provide remote access to messages, but none of them offers interaction comparable to what users enjoy with a GUI mail reader. Phoneshell and Chatter prioritize messages but neither summarize them nor allow random access. SpeechActs scans message headers and allows the user to pick out a message by number, but remembering the number of a message adds to the user's cognitive load. Wildfire takes a small step in the right direction, allowing the user to ask if there are messages from other system users, but does not summarize incoming messages and allow users to pick from among them. To improve upon previous work, MailCall must summarize the information space and support random access to individual messages without imposing great cognitive demands on the user.

MAILCALL

MailCall is a telephone-based messaging system which employs speech recognition for input and speech synthesis for output. It was developed on a Sun Sparcstation 20 under both SunOS 4.1.3 and Solaris, using the DAGGER speech recognizer from Texas Instruments and DECtalk for text-to-speech synthesis. Call control is facilitated by XTL, ISDN software from Sun Microsystems.

Unified voice/text message retrieval. MailCall retrieves incoming messages and places them in categories depending on their importance. The user can ask the sender, subject, arrival time, or recipients of any message. Audio attachments are processed and played as sound files, and email notification sent by a homegrown voice mail system acts as a pointer to the original voice message. Messaging is "unified" in that there is no differentiation by media; the user might have two email messages and one voice message from the same person, and they would be grouped together.

Sending messages. The user can send a voice message in reply to any message, or to anyone in the rolodex. If the recipient is a local voice mail subscriber, it will be placed in the appropriate mailbox; if not, then it is encoded—available formats include Sun, NextMail, MIME, and uuencode—and sent as electronic mail. (Dictating replies to be sent as text is not feasible with current speech recognition.)

Voice Dialing. Instead of sending a voice message, the user may elect to place a call instead. If the person's phone number is available in the rolodex, MailCall uses it—and if there is both a home and work number, MailCall prompts the user to choose one or the other. If someone's phone number cannot be found, the user is prompted to enter it.

Usage Scenarios

MailCall is most useful when the user is beset with a long list of incoming messages or when time is short. In the first example, Pat is in a dreadful hurry trying to meet a deadline, waiting for Lynn to send a critical piece of information. MailCall enables Pat to check for it quickly.

MailCall: You have seven personal messages.
Pat: Is there anything from Lynn Smith?
MailCall: Sorry, no messages from Lynn Smith.
Pat: Good-bye.
MailCall: Did you say to hang up?
Pat: Yep.
MailCall: Pat, thanks for calling. Good-bye!

In a second example, Pat calls up after a marathon afternoon meeting to find over 100 new messages. MailCall's summarization and random access capabilities enable Pat to hear the most important ones first.

MailCall: Welcome to MailCall. Who's there?
Pat: It's Pat Peterson again.
MailCall: Pat Peterson, correct?
Pat: Correct.
MailCall: Hello, Pat. Please say your six-digit password.
Pat: 5-6-5-5-1-9.
MailCall: Hold on while I get your messages...Just a second...Okay! You have 16 timely messages, 33 personal ones, and 150 others.
Pat: Is there anything from Lynn Smith?
MailCall: Personal messages from Lynn Smith. Message 25 from Lynn Smith about "here's the info."
Pat: When did that arrive?
MailCall: It arrived about three hours ago.
Pat: Send a reply.
MailCall: Record your message for Lynn Smith. Pause for several seconds when finished.
Pat: *Hey Lynn, thanks for finally sending the info, but it was a bit too late. Let's try to coordinate a little better next time.*
MailCall: Got it. Review the message, send it, or cancel?
Pat: Send it.
MailCall: Did you say "send it"?
Pat: Yes.
MailCall: Please wait while I deliver your message....What now?
Pat: Go to my timely messages.
MailCall: Timely messages. I can read you the summary, or you can start with the first one.
Pat: Read me the summary.
MailCall: You have a bunch from Chris Barry, a few from Mark Matthews, one each from Deb Cohen and Dominique Farnsworth, and several others.
Pat: Read me the messages from Mark Matthews.

MailCall: Messages from Mark Matthews. Message 5 from Mark Matthews about "can we meet?"
Pat: Let me hear it.
MailCall: (playing voice message) *Hey Pat, I've got a great new pyramid scheme I'd like to get you involved in. We'll make millions!*
Pat: Delete it.
MailCall: Did you say to delete the message?
Pat: That's correct.
MailCall: Deleting message 5.
Pat: Go to my other messages.
MailCall: Other messages. Read the summary, or read the first header?
Pat: Read me the summary.
MailCall: You have an incredible number from Mail Delivery Subsystem, a few from Fran Tess, several from Joyce Ball, one from listserv@mit.edu, and many others.
Pat: Read the one from listserv at MIT dot e d u.
MailCall: Message 121 from listserv@mit.edu about "your request for privacywatch." You have been added to the privacywatch mailing list. To unsubscribe send a message.....(etc)
Pat: That's all for now.
MailCall: Did you say to hang up?
Pat: Yep..
MailCall: Pat, thanks for calling. Good-bye!

As the above examples demonstrate, MailCall enables the user to find important messages quickly. Its *organization* of incoming messages into categories makes it feasible to provide efficient *presentation*, which in turn affords effective *navigation* of the information space. Further, its attention to conversational context helps establish its credibility as a cooperative conversant. The next two sections describe MailCall's approach to nonvisual information management and its strategies for crafting conversation.

DESIGN CHALLENGE: NONVISUAL NAVIGATION

Retrieving messages over the phone is more cumbersome than with a GUI-based mail reader. With a visual interface, the user can immediately see what messages are available and access the desired one directly via point and click. In a nonvisual environment, however, a system must list the messages serially, and since speech is serial and slow, care must be taken not to overburden the user with long lists of choices. Organizing the information space by breaking down a long list of messages into several shorter lists is a first step. Once these smaller, more manageable lists are formed, the system must quickly present them so that the user can choose what to read first. And once the user is informed of available options, the system must provide simple, natural methods of picking a particular message out of the list.

Information organization

A first step towards effective message management in a nonvisual environment is prioritizing and categorizing messages. Like many other mail readers, MailCall filters incoming messages based on a user profile which consists of a set of rules for placing messages into categories. Although rule-based filtering is powerful, writing rules to keep up with dynamic users interests can require significant effort on the part of the user. We considered memory-based reasoning (adopted by Chatter) to lighten the burden on the user, but decided against it since the associated learning curve can be too slow to capture dynamic interests.

Capturing dynamic user interests either by requiring the user to write filtering rules or attempting to infer priorities from past behavior ignores a wealth of information in the user's work environment. The user's calendar, for instance, keeps track of timely appointments, and a record of outgoing email suggests people who might be important. MailCall exploits these various information sources via a background process called CLUES, which scans various databases and automatically generate rules to be used for filtering. (For details on the mechanics of the search see [4]). The messages which are prioritized by CLUES are placed in a "timely" category —thus the user can benefit from both static rules in the filtering profile and dynamic ones generated automatically by CLUES. Below are described several information sources and how CLUES uses them to prioritize messages. The elements which match entries in the information sources are printed in bold.

MONDAY 10am Motorola visit 5pm leave for airport	TUESDAY visiting Sun all day fax: 415-555-5555
ROLODEX CARD 32 OF 89 Name: Kim Silverstone Address: 25 Harshwood Way, Palo Alto CA 94306 Phone: (415) 555-5555 Email: ksilvers@eng.sun.com	
OUTGOING EMAIL: geek@media.mit.edu *latest draft of paper* chi96@sigchi.acm.org *please please give me an extension* mickelsen@leland.stanford.edu *What's up?*	
OUTGOING PHONE CALLS: 215-555-5555 (Charles Hemphill) 8-5956	

Figure 1: various information sources available in the user's computing environment, including calendar, rolodex, and records of outgoing email and phone calls.

Calendar. Assuming an "interest window" of approximately two weeks ahead of the current date and a few days back, CLUES extracts salient items from calendar entries. Thus the following message is marked "timely" even though it was not addressed directly to the user.

From lad@media.mit.edu
To: demo-staff@media.mit.edu
Subject: visitors from **Motorola** delayed until 3:30

Email replies. CLUES scans the user's record of outgoing messages to see who might be sending a reply and what they might be replying about. For instance, CLUES would automatically mark the following message as "timely."

From **chi96@sigchi.acm.org**
Subject: Re: **please please give me an extension**

Returned phone calls. Similarly, CLUES can detect when someone returns a call by correlating the user's record of outgoing phone calls—created when the user dials using one of a number of desktop dialing utilities—with the Caller ID number of voice mail. Our voice mail system sends the user email with the Caller ID of the incoming message. CLUES would prioritize the following voice message:

From Operator <root@media.mit.edu>
Subject: Voice message from **8-5956**

Email replies to phone calls. Often, someone does not reply using the same medium; one may call in response to an email message or send email in reply to a voice message. Finding the email address the person has called by looking up the phone number in the rolodex, CLUES can mark an email reply to an outgoing phone call as "timely." In this example, CLUES marks the email message from Charles Hemphill timely since I recently tried to call him.

From hemphill@csc.ti.com (**Charles Hemphill**)
Subject: I just got your voice mail; here's what I think

Geographic filtering. Business travelers often leave behind a phone number where they can be reached. CLUES correlates a phone number found in the calendar with the user's rolodex to produce a list of people who live in that area. (CLUES also keeps a small table of co-located area codes such as 408 and 415.) A message from any of those people is then marked "timely" under the assumption that the user might be trying to coordinate with them while in town. Hence, messages from those the user is visiting suddenly become important, as when my calendar has a fax number in an area code near where Kim Silverstone lives:

From **ksilvers@eng.sun.com**
Subject: let's do lunch while you're in town!

Domain-based filtering. CLUES also extracts the domain names of the email addresses for those people, presuming that messages from people at that same site may too be relevant. Since CLUES identified Kim Silverman as someone whom the user might want to visit on Tuesday's trip, messages from anyone at the same domain are marked timely.

From smith@**eng.sun.com**
Subject: Kim Silverstone told me you're in town...

Isolating the important part of a domain name is nontrivial. Some people may receive email at one sub-domain (groucho@media-lab.media.mit.edu) but send from another (groucho@timpanogos.media.mit.edu). Some corporations are geographically distributed; eng.sun.com is in California, but east.sun.com is in Massachusetts. Others like aol.com are non-location specific, so they are useless in determining timeliness. CLUES prunes domain names as appropriate.

MailCall's categorization breaks up a long list of messages into several smaller, related lists, one of those being the messages identified as important by CLUES. Once the messages have been sorted into various categories, the user needs a way to navigate among categories. Although messages may be filtered in order of interest, categories can nonetheless serve as navigational landmarks which assist in keeping context and returning to already-covered ground. The MailCall user can jump from category to category in nonlinear fashion, saying "go to my personal messages" or "go back to my important messages."

Nonvisual presentation

Categorization of messages helps to segment the information space, but when there are many messages within a single category, the user once again is faced with the challenge of finding important messages in a long list. Creating more and more categories merely shifts the burden from navigating among messages to navigating among categories; rather, the user must have an effective method of navigating within a category—or, more generally, of finding one's way through a large number of messages. Efficiently summarizing the information space is the second step toward effective nonvisual messaging.

With a GUI-based mail reader, the user is treated to a visual summary of messages and may point and click on items of interest. This works because a list of the message headers quickly summarizes the set and affords rapid selection of individual messages. These are difficult to achieve aurally, however, due to the slow, non-persistent nature of speech. Whereas the eyes can visually scan a list of several dozen messages in a matter of seconds, the ear may take several minutes to do the same; further, the caller must rely on short-term memory in order to recall the items listed whereas the screen serves as a persistent reminder of one's choices.

Since speech is slow, summaries must be streamlined, avoiding extraneous information or repetition. The approach adopted by SpeechActs is to read the headers one by one:

SpeechActs: Your next six messages are from MIT.
User: Scan the headers.
SpeechActs: Message 1 from Gina-Anne Levow about "final draft changes." Message 2 from Chris Schmandt about "visit Friday." Message 3 from Chris Schmandt about "visit Friday—correction." Message 4 from Matt Marx about "videotape."(etc).....Message 10 from Matt Marx about "final draft changes." Message 7 from Matt Marx about "hello."

MailCall's summary is more concise, though at the expense of some detail.

MailCall: Timely messages. Read the summary, or

read the first header?
User: Read me the summary.
MailCall: You have several from Matt Marx, a couple from Chris Schmandt, and one from Gina-Anne Levow.

Although the latter summary does not list the subject of each message, it is more quickly conveyed and easier to remember. By grouping messages from a single sender, it avoids mentioning each message individually, instead providing a summary of what is available.

In addition, MailCall attempts not to overburden the user with information. When reading the list, for instance, it does not say the exact number of messages but rather a "fuzzy quantification" of the number: e.g., "several from Matt Marx" instead of "six from Matt Marx." And if there are messages from many different people in the same category, MailCall will mention only the first four or five and add "...among others."

Nonvisual navigation

Now that the user can hear a summary of available messages, it is practical to support random access to individual messages. Random access refers to the act of nonlinear information access—i.e., something other than the neighboring items in a list. The chart delineates four general modes of random access.

	location-based	**content-based**
relative	"skip ahead five messages"	"read the next one about 'meeting"
absolute	"read me message thirty-five"	"read the message from John Linn"

Figure 2: Four types of random access.

By *location-based* random access we mean that the navigator is picking out a certain item by virtue of its position or placement in a list—i.e., "Read message 10." Location-based random access may either be *absolute* (as in the preceding example), when the user has a specific message in mind, or *relative*, when one moves by a certain offset: e.g., "skip ahead five messages." (It may be noted that sequential navigation is a form of relative location-based navigation where the increment is one.) SpeechActs implements both absolute and relative location-based random access, and suggests their use by reading the number of each message (for future reference, should the user need to return) and by scanning the headers in a message category. Location-based random access does impose an additional cognitive burden on the user, who must remember the numbering of a certain message in order to access it. Indeed, participants in the SpeechActs usability study were often observed jotting down the numbering of messages, though doing so would be most difficult in situations where speech is maximally beneficial—driving, for instance.

With *content-based* random access the user may reference an item by one of its inherent attributes, be it the sender, subject, date, etc. For instance, the user may say "Read me the message from John Linn." Thus the user need not recall the numbering scheme. Like location-based navigation, both relative and absolute modes exist. Relative content-based access associated with following "threads," multiple messages on the same subject. Phoneshell, for instance, allows the user to drop into "scan mode" which reads the next message on the current topic or from the current sender; this is feasible with touch-tones since the user need not specify the desired topic explicitly but signify that the current topic is to be followed. *Absolute content-based navigation* is the contribution of MailCall, allowing the user to pick the interesting message(s) from an efficient summary without having to remember details of position.

MailCall: You have several messages from Lisa Stifelman, a few from Mike Phillips, and one each from Jill Kliger and Steve Lee.
User: Read me the ones from Mike Phillips.

It is practical to support absolute content-based navigation thanks to recent advances in speech recognition. Normally a speech recognizer has a static, precompiled vocabulary which cannot be changed at runtime. This makes it impractical for the speech recognizer to know about new messages which arrive constantly. (This explains why neither Chatter nor SpeechActs could offer absolute content-based random access). Recently, however, a dynamic vocabulary updating feature added to the Dagger speech recognizer [2] enables us to add the names at runtime. When the user enters a category, MailCall adds the names of the email senders in that category to the recognizer's vocabulary. Thus the user may ask for a message from among those listed in a summary. One may also ask if there are messages from anyone listed in the rolodex, or from whom one has recently sent a message or called (as determined by CLUES). Supporting absolute content-based random access in MailCall with Dagger dynamic vocabulary updating is a positive example of technology influencing design. Absolute content-based random access brings MailCall closer in line with the experience one expects from a graphical mail reader.

DESIGN CHALLENGE: CRAFTING CONVERSATION

The rich nature of MailCall's interaction exceeds the capabilities of touch-tones and thus requires the use of speech recognition. The use of speech for both input and output raises user expectations, since the interaction implicitly resembles a conversation. Since speech recognizers are far less adept than humans, additional obstacles exist in designing nonvisual messaging systems. Knowing what to say is a stumbling block for new users, yet taking the time to explain one's options can be tiresome for experienced users. Further, speech recognition errors slow down the interaction. This section describes strategies for avoiding conversational breakdown.

Communicating capabilities with help

A major responsibility of the speech interface designer is to communicate system capabilities to the user. People may

have very different expectations for a "conversational" system: some may speak as if chatting with a good friend, expecting the machine to perform human-level speech recognition as well as topic-independent language understanding. Experience [8], however, demonstrates that many people in fact have extremely low expectations of spoken language systems. They sit mute in front of the telephone or speak slowly with exaggerated enunciation. In either case, it is essential to guide users to phrase requests which the system understands.

Like SpeechActs, MailCall eschews explicitly listing options in the form of a menu, instead opting for a more conversational style which is both more familiar and faster for experts who know what to say. Still, novice users need to know what their options are. The SpeechActs strategy involved giving each user a printed card with a list of sample commands. In the usability study, this proved successful; most users speak only those commands listed on the card. For MailCall, our goal is an "out-of-box" experience, meaning that we want users to be able to learn the system without reading manuals or carrying instruction pamphlets. MailCall users are informed at the beginning of a session that they can ask "what can I say now?" or press the 0 key for assistance.

The first step in providing help is *reestablishing context*—reminding users where they are and how they got there, usually by revisiting the last action taken by the system and explaining what the system thought the user said. Next, a list of *currently available options* is given with an explanation of each command. Finally, the user is reminded of the *global reset* command ("start over from the beginning") so that the user can begin the session anew instead of hanging up in disgust.

Handling recognition errors

Like many speech systems [7, 8], MailCall invests significant effort in detecting and correcting speech recognition errors. It handles rejection errors by apologizing and giving *progressive assistance*, and also verifies potential substitution errors, allowing the user to correct them quickly.

Its contribution to error-handling is ErrorLock, a general interface algorithm for handling recognition errors. Instead of dealing with errors on a case-by-case basis, each incoming utterance is passed through ErrorLock for evaluation and confirmation. Aside from insuring uniform error-handling among varied input, it also keeps track of past recognition errors and deals more intelligently with consecutive errors. If, for instance, the recognizer consistently picks an hypothesis but its confidence is just under the cutoff threshold, ErrorLock temporarily relaxes the threshold and asks if that is what the user meant. Error-tracking also helps to keep MailCall from sounding oblivious to context. In the following example, MailCall makes the same mistake twice but reflects that knowledge in its feedback.

MailCall: Welcome to MailCall. Who's this?
User: Matt Marx.
MailCall: Nat Parker, correct?
User: No, Matt Marx.
MailCall: I thought I heard "Nat Parker" again, but you just said that. Is that right?

If the speech recognizer were able to return more than one hypothesis (called an N-best list), ErrorLock would be able to automatically throw away hypotheses which have already been rejected and check the second-most-likely choice.

In addition to facilitating better interaction, ErrorLock is a convenience for the developer. Centralizing error-handling removes the need to copy error-handling code all over the application, and since ErrorLock is domain-independent, it can be reused for other applications, changing prompts as necessary.

Contextually customized feedback

Grice's cooperative principle of conversation [1] as applied to spoken language systems might read as follows: "give as little feedback as possible, but no less." Giving too much feedback is a vice , but failing to give enough feedback is a greater sin if the user becomes confused for lack of explanation. Both the user's level of competence and the conversational context of the current session are factors in deciding how to give feedback.

Varying the length of feedback. Like other speech systems [6, 7], MailCall offers various levels of feedback, with longer prompts for novices than experts. MailCall also adapts its feedback based on the conversational context, usually manifesting itself through the use of *ellipsis*. Ellipsis refers to the leaving out of words or phrases which have already been established earlier in the conversation, assuming that the intended meaning can be reconstructed from a shorter prompt and the existing context. In the following example, the user is asked twice whether to review, send, or cancel a message. Items omitted by ellipsis appear in strikeout text.

User: Send a message to Jordan Slott.
MailCall: Record your message for Jordan Slott. Pause for several seconds when finished.
User: *records a voice message to be sent*
MailCall: Got it. Review the message, send it, or cancel?
User: Send.
MailCall: Sending your message. Please wait.
User: Send a message to Steve Lee.
MailCall: Record your message for Steve Lee. ~~Pause for several seconds when finished.~~
User: *records a voice message to be sent*
MailCall: Got it. Review ~~the message~~, send ~~it~~, or cancel?
User: Send.
MailCall: Sending your message. Please wait.

As the above example demonstrates, MailCall uses ellipsis on individual words, phrases, and even sentences. Ellipsis helps to streamline the interaction and may help users to

perceive the interface as non-repetitive.

Varying the speed of feedback. Like other speech systems [7], MailCall allows the user to set a default speech output rate or change it during a session. It also recognizes that certain items are more familiar than others, having been established earlier in the conversation. Prompts, for instance, become familiar with use. Items which are new to the conversation, however, may be harder to understand. Thus MailCall temporarily slows down its speech—relative to the current output rate—when presenting the sender or subject of a message. It does so in the following example, with the words spoken more slowly rendered in expanded text.

User: Start with the first message.
MailCall: Message 1 is from S t u a r t A d a m s about a response to "n e x t w e e k."

Temporarily slowing down for new information can allow the user to set a high default speaking rate, speeding the presentation of prompts. Ellipsis and automatic speed adaptation help to reduce the repetitiveness of repeated prompts while helping to insure that new information can be comprehended.

FIELD STUDY

To evaluate the effectiveness of MailCall, a user study was conducted. The goal was not only to determine how usable the system was for a novice, but also how useful it would prove as a tool for mobile messaging.

Method

Since our goal was not only to evaluate ease of learning but likelihood of continued use, we conducted a long-term user study. The five-week study involved four novice (yet technically savvy) users with varying experience using speech recognition. In order to gauge the learning curve, minimal instruction was given except upon request. Sessions were not recorded or monitored due to privacy concerns surrounding personal messages, so the results described below are based chiefly on user reports. The experiences of the two system designers using MailCall over a period of three months were also considered.

Results

Feedback from novices centered mainly on the process of learning the system, though as users became more familiar with the system, they also commented on the utility of MailCall's nonvisual presentation. Seasoned users offered more comments on navigation as well as the limits of MailCall in various acoustic contexts.

Bootstrapping. As described above, our approach was to provide a conversational interface supported by a help system. All novice users experienced difficulty with recognition errors, but those who used the help facility found they could sustain a conversation in many cases. A participant very familiar with speech systems found the combination of error-handling and help especially useful:

> I have never heard such a robust system before. I like all the help it gives. I said something and it didn't understand, so it gave suggestions on what to say. I really liked this.

Other participants were less enthusiastic, though nearly all reported that their MailCall sessions became more successful with experience.

Recognition robustness. MailCall has a very large vocabulary, pushing the limits of DAGGER. The experience of the system designers was used to measure the usability of MailCall in various contexts (assuming that they knew the vocabulary). Two factors were identified in the success of the system: vocabulary size and acoustic context. Since the user can ask if there are messages from anyone in the rolodex, send them messages, or call them at any time, MailCall's vocabulary size varies directly with the number of people in the user's rolodex. A user with well over 100 names in his rolodex found recognition to be unacceptable when talking over a noisy cellular connection.

Specifying Names. A difficulty common to all users was getting MailCall to understand names, both in asking for messages from certain people and simply identifying one's self. Unlike Chatter, which used first names only, MailCall requires the user to say full names. Everyone responded to the prompt "Welcome to MailCall. Who's this?" with "It's Matt!" or something similar. Furthermore, users were disappointed when they had to refer to someone the same way that MailCall did; instead of saying "read the one from groucho at media at MIT dot E D U," they wanted to say "read the one from groucho."

Navigation. Users cited absolute content-based navigation as a highlight of MailCall. One beginning user said "I like being able to check if there are messages from people in my rolodex [just by asking]." And one of the system designers, a diehard Phoneshell user, admits that he uses MailCall instead of Phoneshell when facing an unusually large number of messages because he can ask for a summary of a category and then pick the ones he wants to hear first.

For sequential navigation, however, speech was more a bane than a boon. The time necessary to say "next" and then wait for the recognizer to respond can be far greater than just pushing a touch-tone, especially when the recognizer may misunderstand. Indeed, several used touch-tone equivalents for "next" and "previous." And since some participants in the study received few messages, they were content to step through them one by one.

These results suggest that MailCall is most useful to people with high message traffic, whereas those with a low volume of messages may be content to simply step through the list with touch-tones, avoiding recognition errors.

Implications for redesign

The results of our user study suggested several areas where MailCall could improve, particularly for novice users. Some changes have already been made, though others will

require more significant redesign of the system.

More explanation for beginners. Supporting conversational prompts with help appears to be a useful method of communicating system capabilities to novices. Our experience with four novice users, however, suggests that our prompts and help were not explicit enough. As a step in iterative design, we lengthened several prompts including those at the beginning of a session and raised the level of detail given during help; a fifth novice user who joined the study after these changes had been made was able to log on, navigate, and send messages on his very first try without major difficulties. This suggests that prompts for beginners should err on the side of lengthy exposition.

More flexible specification of names. Specifying names continues to be an elusive problem. MailCall should allow the user to refer to someone using as few items as necessary to uniquely specify them. Doing so would involve two additions to MailCall: first, a "nickname generator" which creates a list of acceptable alternatives for a given name; second, an interface algorithm for disambiguating names with multiple referents as in [8].

Moded vs. Modeless interaction. If MailCall is to be usable in weak acoustic contexts (like the cellular phone) for people with a large rolodex, its interaction may need to become more modal. We intentionally designed MailCall to be modeless so that users would not have to switch back and forth among applications, but as the number of people in the rolodex grows, it may become necessary to define a new "rolodex" application.

CONCLUSIONS

Telephone-based messaging systems can approach their visual counterparts in usability and usefulness if users can quickly access the messages they want. Through a combination of message organization, presentation, and navigation, MailCall offers interaction more similar to that of a visual messaging system than previously available. Consideration of context helps to meet user expectations of error-handling and feedback, though beginning users may require more assistance than was anticipated. Results suggest, however, that a large-vocabulary conversational system like MailCall can be both usable and useful for mobile messaging.

ACKNOWLEDGMENTS

Jordan Slott implemented the ISDN software, and several research associates of the MIT Media Lab Speech Group have contributed to infrastructure. Raja Rajasekharan and Charles Hemphill made it possible for us to use DAGGER. Nicole Yankelovich reviewed a draft of this paper. This work was supported by Sun Microsystems and Motorola.

REFERENCES

[1] H. Grice. "Logic and Conversation," *Syntax and Semantics: Speech Acts*, Cole & Morgan, editors, Volume 3, Academic Press, 1975.

[2] C. Hemphill & P. Thrift. "Surfing the Web by Voice." *To appear in ACM Multimedia '*95, San Francisco, CA, Nov. 5–9, 1995.

[3] E. Ly. "Chatter: A Conversational Telephone Agent" MIT Master's Thesis, Program in Media Arts and Sciences, 1993.

[4] M. Marx. "Toward Effective Conversational Messaging." MIT Master's Thesis, Program in Media Arts and Sciences, 1995.

[5] C. Schmandt. "Speech Synthesis Gives Voiced Access to an Electronic Mail System." *Speech Technology*, Aug/Sept 1984, pp. 66–68.

[6] C. Schmandt. "Phoneshell: the Telephone as Computer Terminal" *Proceedings of ACM Multimedia Conference*, August 1993.

[7] L. Stifelman, B. Arons, C. Schmandt, and E. Hulteen. "VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker, *ACM INTERCHI '93 Conference Proceedings*, Amsterdam, The Netherlands, April 24–29, 1993.

[8] N. Yankelovich, G. Levow, and M. Marx. "Designing SpeechActs: Issues in Speech Interfaces." *Proceedings of CHI '95*, Denver, CO, May 8–11, 1995.