Published in CHI '95 Proceedings, Conference on Human Factors in Computing Systems, Denver, CO, May 7-11, 1995.

Designing SpeechActs: Issues in Speech User Interfaces

Nicole Yankelovich, Gina-Anne Levow*, Matt Marx*

Sun Microsystems Laboratories
Two Elizabeth Drive
Chelmsford, MA, USA 01824

*Current addresses listed below.

nicole.yankelovich@east.sun.com
gina@ai.mit.edu
groucho@media.mit.edu

Postscript Version - 8 Pages

ABSTRACT

SpeechActs is an experimental conversational speech system. Experience with redesigning the system based on user feedback indicates the importance of adhering to conversational conventions when designing speech interfaces, particularly in the face of speech recognition errors. Study results also suggest that speech-only interfaces should be designed from scratch rather than directly translated from their graphical counterparts. This paper examines a set of challenging issues facing speech interface designers and describes approaches to address some of these challenges.

Keywords: Speech interface design, speech recognition, auditory I/O, discourse, conversational interaction.

INTRODUCTION

Mobile access to on-line information is crucial for traveling professionals who often feel out of touch when separated from their computer. Missed messages can cause serious inconvenience or even spell disaster when decisions are delayed or plans change.

A portable computer can empower the nomad to some degree, yet connecting to the network (by modem, for example) can often range from impractical to impossible. The ubiquitous telephone, on the other hand, is necessarily networked. Telephone access to on-line data using touch- tone interfaces is already common. These interfaces, however, are often characterized by a labyrinth of invisible and tedious hierarchies which result when menu options outnumber telephone keys or when choices overload users' short-term memory.

Conversational speech offers an attractive alternative to keypad input for telephone-based interaction. It is familiar, requires minimal physical effort for the user, and leaves hands and eyes free. And since physical space presents no constraint for a speech system, the number of commands is virtually unlimited.

Implementing a usable conversational interface, however, involves overcoming substantial obstacles. Error-prone speech recognizers require the system to emphasize feedback and verification, yet the time it takes to identify and repair errors can be tiring. Further, conversational interfaces are young, and transferring design principles from other media such as graphics can lead to unusable systems. Despite these problems, we, along with others [6, 9, 10, 11], believe the technology good enough and the promise exciting enough to make experimentation worthwhile.

In the SpeechActs project, we seek to identify principles and challenges of conversational interface design and to pinpoint limitations of current technology. In so doing, we hope to define useful avenues for research and suggest strategies for addressing the difficult problems facing speech user interface designers.

In this paper, we first describe the functionality of the SpeechActs system. We then explain our methodology, including usability testing and iterative redesign. We conclude with speech user interface design challenges and strategies for meeting those challenges in a speech-only environment.

THE SpeechActs SYSTEM

SpeechActs is a research prototype that integrates third-party speech recognition and synthesis with telephony, natural language processing capabilities, and other tools for creating speech applications. For an overview of the SpeechActs architecture, see [13]. To date, the system includes speech-only interfaces to a number of applications including electronic mail, calendar, weather, and stock quotes.

With the **mail** application, which uses Sun's Mail Tool backend, users can hear their messages, skip forward or backward from one header to the next, fax a message, reply to a message, or initiate a new message to a person on their *short list* of names known to the speech recognizer. To fax messages, users can say the name of a predefined location (i.e., work, home) or can specify a fax number by using telephone keypad input. When sending a message, the user has the option of including the current message and/or including a recorded audio attachment. Following is an example of a typical interaction with the mail application:

```
SpeechActs:  You have 14 new messages and 47 old
             messages. The first new message is from
             Eric Baatz regarding "report completed."

User:        Let me hear it.
```

```
SpeechActs:  "The first draft is ready for your comments.
             Eric."

User:        Reply and include the current message.

SpeechActs:  Preparing message to Eric Baatz. Begin
             recording after the tone. When finished,
             pause for several seconds.

User:        Eric, I'll get those to you this afternoon.

SpeechActs:  Say cancel, send, or review.

User:        Send.

SpeechActs:  Message sent. What now?

User:        Next message.

SpeechActs:  New message two from Stuart Adams,
             subject "Can we meet at 3:00 today?"

User:        Switch to calendar...
```

The SpeechActs **calendar** interface, based on Sun's Calendar Manager application, allows users to browse their own calendar as well as the calendars of other users on their short list. When the user requests information, the application reads them all the events on a selected day. Typical calendar queries include:

```
What do I have tomorrow?
What about Bob?
What did he have last Wednesday?
And next Thursday?
What was Paul doing three days after Labor Day?
```

The **weather** application provides an interface to the University of Michigan's on-line Weather Underground forecasts. Users can call up and ask for weather for states and for major cities around the country. For example, the user can say:

```
What's the weather in Seattle?
How about Texas?
I'd like the extended forecast for Boston.
```

Like the weather application, the **stock quotes** application provides a speech interface to a dynamic data feed. The user is able to ask for the prices of selected stocks, ask about their highs, lows, and volume, or ask for the prices of stocks in their portfolio (a stored list of stocks). Sample queries include:

```
What's the price of Sun?
What was the volume?
Tell me about IBM.
How's my portfolio doing?
```

As with multiple graphical applications running in the same environment, SpeechActs supports a standard set of functions that are always available in any application. For example, the user may always switch to a different application, ask for help, or end a session by saying "good bye."

USER STUDY / ITERATIVE DESIGN

Before the SpeechActs software was written, we conducted a survey and a field study [12] which served as the basis for the preliminary speech user interface (SUI) design. Once we had a working prototype, we conducted a usability study in which we adhered to Jakob Nielsen's formative evaluation philosophy of changing and retesting the interface as soon as usability problems are uncovered [8]. As a result, the formative evaluation study involved small groups of users and a substantial amount of iterative redesign.

Formative Evaluation Study Design

Fourteen users participated in the study. The first two participants were pilot subjects. After the first pilot, we redesigned the study, solved major usability problems, and fixed software bugs. After the pilots, nine users, all from our target population of traveling professionals, were divided into three groups of three. Each group had two males and one female. An additional three participants were, unconventionally, members of the software development team. They served as a control group. As expert SpeechActs users, the developers provided a means of factoring *out* the interface in order to evaluate the performance of the speech recognizer.

After testing each group of target users, we altered the interface and used the next group to validate our changes. Some major design changes were postponed until the end of the study. These will be tested in the next phase of the project when we plan to conduct a longer-term field study to measure the usefulness of SpeechActs as users adapt to it over time.

Tasks

During the study, each participant was led into a room fashioned like a hotel room and seated at a table with a telephone. They were asked to complete a set of 22 tasks, taking approximately 20 minutes, and then participate in a follow-up interview.

The tasks were designed to help evaluate each of the four SpeechActs applications, as well as their interoperation, in a real-life situation. To complete the tasks, participants had to read and reply to electronic mail, check calendar entries for themselves and others, look up a stock quote, and retrieve a weather forecast.

Instead of giving explicit directions, we embedded the tasks in the mail messages. Thus the single, simple directive "answer all new messages that require a response" led to the participants executing most of the tasks desired. For example, one of the messages read as follows: "I understand you have access to weather information around the country. If it's not too much trouble, could you tell me how warm it is going to be in Pittsburgh tomorrow?" The participant had to switch from the mail application to the weather application, retrieve the forecast, return to the mail application, and prepare a reply.

Although the instructions for completing the task were brief, participants were provided with a "quick reference card" with sample commands. For example, under the heading "Mail" were phrases such as "read me the first message," "let me hear it," "next message," "skip that one," "scan the headers," and "go to message seven." In addition, keypad commands were listed for stopping speech synthesizer output and turning the recognizer on and off.

Summary of Results

After testing the first group of users, we were able to identify the main problems in the interface. Each of our users bemoaned the slow pace of the interaction, most of them thought the computer gave too much feedback, and almost everyone insisted that they be able to interrupt the speech output with their voice. Most egregious was our inappropriate translation of the Sun Mail Tool message organization into speech. A technique that worked well in the graphical interface turned out to be confusing and disorienting in the speech interface. Details about this problem with message organization along with other design-related study results are woven into the discussion on design challenges in the following section.

In the study, our main aim was not to collect quantitative data; however, the data we gathered did suggest several trends. As hoped, we noticed a marked, consistent decrease in both the number of utterances and the amount of time required to complete the tasks from one design cycle to the next, suggesting that the redesigns had some effect. On average, the first group of users took 74 utterances and 18.5 minutes to complete the tasks compared to the third group which took only 62 utterances and 15 minutes (Table 1).

Participants	Utterances	Time (minutes)
Group 1	74	18.67
Group 2	63	16.33
Group 3	62	15.00
Developers	43	12.33

Table 1. Average number of utterances and time to complete tasks.

At the start of the SpeechActs project, we were aware that the state of the art in speech recognition technology was not adequate for the conversational applications we were building. One of our research questions was to determine if certain types of interface design strategies might increase users' success with the recognizer. Unfortunately, none of our redesigns seemed to have an impact on recognition rates - the number of utterances that resulted in the system performing the correct action. They remained consistent among the groups, with the developers showing about a 10% better rate than the first-time users. More significant than the design was the individual; for instance, female participants, on average, had only 52% of their utterances interpreted correctly compared to 68.5% for males. Even with these low recognition rates, the participants were able to complete most of the 22 tasks. Males averaged 20 completed tasks compared to 17 for females (Table 2).

Participants	Recog. Rates	Tasks Completed
Female	52%	17
Male	68.5%	20
Developers	75.3%	22

Table 2. Average recognition rates and number of tasks completed.

Paradoxically, we found that recognition rates were a poor indicator of satisfaction. Some of the participants with the highest error rates gave the most glowing reviews during the follow-up interview. It is our conclusion that error rates correlate only loosely with satisfaction. Users bring many and varying expectations to a conversation, and their satisfaction will depend on how well the system fulfills those

expectations.

Moreover, expectations other than recognition performance colored users' opinions. Some participants were expert at using Sun's voice mail system with its touch-tone sequences that can be rapidly issued. These users were quick to point out the slow pace of SpeechActs; almost without exception they pointed out that a short sequence of key presses could execute a command that took several seconds or longer with SpeechActs.

Overall, participants liked the concept behind SpeechActs and eagerly awaited improvements. Barriers still remain, however, before a system like SpeechActs can be made widely available. The next section provides a more in-depth discussion of the challenges inherent in speech interfaces as well as solutions to some of these suggested by our users' experience with SpeechActs.

DESIGN CHALLENGES

In analyzing the data from our user studies, we have identified four substantial user interface design challenges for speech-only applications. Below is a description of each challenge along with our approach to addressing the challenge.

Challenge: Simulating Conversation

Herb Clark says that "speaking and listening are two parts of a collective activity" [1]. A major design challenge in creating speech applications, therefore, is to simulate the role of speaker/listener convincingly enough to produce successful communication with the human collaborator. In designing our dialogs, we attempt to establish and maintain what Clark calls a *common ground* or shared context.

To make the interaction feel conversational, we avoid explicitly prompting the user for input whenever possible. This means that there are numerous junctures in the conversational flow where the user must take the initiative. For example, after a mail header is read, users hear a prompt tone. Almost all users comfortably take the lead and say something appropriate such as "read the message," or "skip it." In these cases, we adequately establish a common ground and therefore are rewarded with a conversation that flows naturally without the use of explicit prompts.

When we engaged users in a subdialog, however, study participants had trouble knowing what to say, or even if it was their turn to speak, when the subdialog concluded. The completion of a subdialog corresponds to a *discourse segment pop* in the discourse structure terminology described by Grosz & Sidner [3]. When the subdialog is closed, the context returns to that preceding the subdialog. For example, the user might read a string of messages and then come across one that requires a response. In the reply subdialog, the user has to decide whether or not to include the current message, has to record the new message, and, perhaps, has to review the recording. When finished, the user is back to a point where he or she can continue reading messages. In the Mail Tool graphical user interface (GUI), the reply sequence takes place in a pop-up window which disappears when the user sends the message, and their previous context is revealed. We found that we needed an analogous signal.

Our first attempt to provide a discourse pop cue - a prompt tone at the end of the subdialog - failed. We considered the use of an intonational cue, which is one technique used by human speakers. Since our synthesizer could not produce a clear enough intonational cue, we included an explicit *cue phrase* - "What now?" - to signal the discourse pop. Surprisingly, this small prompt did, in fact, act to signal the

subdialog's completion and return the user to the main interactional context.

Prosody. Prosody, or intonation, is an important element in conversations, yet many of the synthesizers available today do a poor job reproducing human-sounding intonational contours. This means that many types of utterances used by humans cannot be employed in the speech interface design. For example, as an alternative to the phrase "What did you say?", we tried to use "hmm?" and "huh?", but could not reproduce the sounds convincingly.

Despite the lack of good prosodics, most of our study participants said the speech output was understandable. On the other hand, many complained that the voice sounded "tinny," "electronic," or "choppy."

Pacing. Another important aspect of conversation involves pacing. Due to a variety of reasons, the pacing in SpeechActs applications does not match normal conversational pacing. The pauses in the conversation resulting from recognition delays, while not excessively long by graphical interaction standards, are just long enough to be perceived as unnatural. One user commented: "I had to get adjusted to it in the beginning...I had to slow down my reactions."

In addition, the synthesizer is difficult to interrupt due to cross-talk in the telephone lines which prevents the speech recognizer from listening while the synthesizer is speaking. In the implementation used by study participants, users had to use keypad input to stop the synthesizer from speaking. Unfortunately, as Stifelman also found [11], users had a strong preference for using their voice to interrupt the synthesizer. A user said: "I kept finding myself talking before the computer was finished. The pacing was off."

We have identified several strategies to improve pacing. First, we are experimenting with a *barge-in* technique that will allow users to interrupt the speech synthesizer using their voice. Second, we would like users to be able to speed up and slow down the synthesized speech. This way they could listen to familiar prompts and unimportant messages quickly, but slow the speaking down for important information. We are also considering adding keypad short-cuts for functions common to all applications (e.g., next, previous, skip, delete, help, etc.). This will allow advanced users to move more quickly through the information, skipping prompts when appropriate. Another potential aid for advanced users, which Stifelman recommends [11], is replacing some of the spoken prompts with auditory icons or sounds that evoke the meaning of the prompt.

Challenge: Transforming GUIs into SUIs

Since one of the goals of the SpeechActs project is to enable speech access to existing desktop applications, our initial SUI designs were influenced by the existing graphical interfaces. Our user studies, however, made it apparent that GUI conventions would not transfer successfully to a speech-only environment. The evolution of our SUI design shows a clear trend towards interpersonal conversational style and away from graphical techniques.

Vocabulary. An important aspect of conversation is vocabulary. We discovered early on that the vocabulary used in the GUI does not transfer well to the SUI. As much as they may use a piece of software, users are not in the habit of using the vocabulary from the graphical interface in their work-related conversations. Here is one of many examples from our pre-design field study where we analyzed human- human conversations relating to calendars: On the telephone, a manager who is a heavy user of Sun's calendar GUI, asked his assistant to look up information on a colleague's calendar:

```
Manager:     Next Monday - Can you get into John's
             calendar?
```

To access another user's calendar in the GUI, the assistant had to select an item (johnb@lab2) from the Browse menu. In his request, the manager never mentioned the word "browse," and certainly did not specify the colleague's user ID and machine name. Also note his use of a relative date specification. The graphical calendar has no concept of "next Monday" or other relative dates such as "a week from tomorrow" or "the day after Labor Day." These are not necessary with a graphical view, yet they are almost essential when a physical calendar is not present.

It turned out that the assistant could not, in fact, access John's calendar. She received the error message: "Unable to access johnb@lab2" Her spoken reply was:

```
Assistant:   Gosh, I don't think I can get into his
             calendar.
```

In designing each of the SpeechActs applications, we tried to support vocabulary and sentence structures in keeping with users' conversational conventions rather than with the words and phrases used in the corresponding graphical interface. The field study as well as the formative study both indicate that it is unlikely users will have success interacting with a system that uses graphical items as *speech buttons* or spoken commands.

Information Organization. In addition to vocabulary, the organization and presentation of information often does not transfer well from the graphical to the conversational domain. The difficulties we encountered with the numbering of electronic mail messages illustrates the translation problem. In Sun's Mail Tool GUI, messages are numbered sequentially, and new messages are marked with the letter "N." Thus, if you have 10 messages and three are new, the first new message is number 8. The advantage of this scheme is that messages retain the same number even when their status changes from new to old. The "N" is simply removed after a message is read.

We initially used the same numbering scheme in the SUI, but with poor results. Even though the start-up message told the user how many new and old messages they had, users were uniformly confused about the first new message having a number greater than one. When asked about their concept of message numbering, users generally responded that they expected the messages to be organized like Sun's internal voice mail where new messages start with number 1. No one alluded to the Mail Tool organization of messages.

We improved the situation by numbering new messages 1 to *n* and old messages 1 to *n*. Of course, this introduced a new problem. Once a message was read, did it immediately become old and receive a different number? Since we wanted users to be able to reference messages by number (e.g., "Skip back to message four."), renumbering the messages seemed unwise. Instead, we added the concept of "read messages," so if users revisited a message, they were reminded that they had already read it, but the message numbers stayed constant until the end of the session. Following the changes, users consistently stated that they knew where they were in the system, and specifically mentioned the helpfulness of the reminder messages.

Information Flow. Just as one way of organizing information can be clear on the screen and confusing when spoken, so it is with information flow. A frequently used flow-of-control convention in GUI design is the pop-up dialog box. These are often used to elicit confirmation from the user. A typical

example is a Yes/No or OK/Cancel dialog box that acts as a barrier to further action until the user makes a selection. The pop-up is visually salient, and thus captures the user's attention. The closing of the dialog box also serves as important feedback to the user.

We attempted to create speech dialog boxes. For example, we wanted a confirmation from the user before sending a new message (e.g., "Your message is being sent to Matt Marx. Okay?"). The only acceptable answers to this question were "yes," "okay," "no" and some synonyms. Users were highly non-compliant! Some seemed confused by the question; others simply ignored it. Some of the confusion was understandable. Occasionally, users had said something other than "send." If this happened, users often repeated or rephrased their command (e.g., "review") instead of answering the question with a "no." Even without recognition problems, only a few users answered the yes/no question directly. Instead, many simply proceeded with their planned task (e.g., "Read the next message."). Sometimes they added "yes" or "no" to the beginning of their phrase to acknowledge the prompt. This phenomenon was also observed by researchers at NTT [5].

When considered in the context of spoken dialog, this behavior is actually quite natural. As the classic example "Do you have the time?" illustrates, yes/no questions rarely *require* yes/no answers. The listener frequently has to infer yes or no, or pick it out from the context of a larger utterance.

Not being able to count on reliable answers to yes/no questions can be problematic from a design standpoint since designing for errors is a necessity in the speech arena. We handled this problem in a number of different ways. First, we removed as many of these spoken dialog boxes as possible. Where we felt confirmation was necessary, we allowed users to preface commands with yes or no. If they did not, we treated a valid command as an implicit request to "do the right thing." For example, in the case of the exit dialog, "Did you say to hang up?", we treated any valid input as an implicit "no." In the few rare cases where we wanted to be absolutely sure we were able to understand the user's input, we used what Candace Kamm calls *directive prompts* [4] instead of using a more conversational style. For instance, after the user has recorded a new mail message, we prompt them to "Say cancel, send, or review."

Challenge: Recognition Errors

Ironically, the bane of speech-driven interfaces is the very tool which makes them possible: the speech recognizer. One can never be completely sure that the recognizer has understood correctly. Interacting with a recognizer over the telephone is not unlike conversing with a beginning student of your native language: since it is easy for your conversational counterpart to misunderstand, you must continually check and verify, often repeating or rephrasing until you are understood.

Not only are the recognition errors frustrating, but so are the recognizer's inconsistent responses. It is common for the user to say something once and have it recognized, then say it again and have it misrecognized. This lack of predictability is insidious. It not only makes the recognizer seem less cooperative than a non-native speaker, but, more importantly, the unpredictability makes it difficult for the user to construct and maintain a useful *conceptual model* of the applications' behaviors. When the user says something and the computer performs the correct action, the user makes many assumptions about cause and effect. When the user says the same thing again and some random action occurs due to a misrecognition, all the valuable assumptions are now called into question. Not only are users frustrated by the recognition errors, but they are frustrated by their inability to figure out how the applications work.

A variety of phenomena result in recognition errors. If the user speaks before the system is ready to listen, only part of the speech is captured and thus almost surely misunderstood. An accent, a cold, or an exaggerated tone can result in speech which does not match the voice model of the recognizer. Background noise, especially words spoken by passersby, can be mistaken for the user's voice. Finally, out- of-vocabulary utterances - i.e., the user says something not covered by the grammar or the dictionary - necessarily result in errors.

Recognition errors can be divided into three categories: rejection, substitution, and insertion [10]. A *rejection error* is said to occur when the recognizer has no hypothesis about what the user said. A *substitution error* involves the recognizer mistaking the user's utterance for a different legal utterance, as when "send a message" is interpreted as "seventh message." With an *insertion error*, the recognizer interprets noise as a legal utterance - perhaps others in the room were talking, or the user inadvertently tapped the telephone.

Rejection Errors. In handling rejection errors, we want to avoid the "brick wall" effect - that every rejection is met with the same "I didn't understand" response. Based on user complaints as well as our observation of how quickly frustration levels increased when faced with repetitive errors, we eliminated the repetition. In its place, we give *progressive assistance*: we give a short error message the first couple of times, and if errors persist, we offer more assistance. For example, here is one progression of error messages that a user might encounter:

```
What did you say?
Sorry?
Sorry. Please rephrase.
I didn't understand. Speak clearly, but don't overemphasize.
Still no luck. Wait for the prompt tone before speaking.
```

As background noise and early starts are common causes of misrecognition, simply repeating the command often solves the problem. Persistent errors are often a sign of out-of-vocabulary utterances, so we escalate to asking the user to try rephrasing the request. Another common problem is that users respond to repeated rejection errors by exaggerating; thus they must be reminded to speak normally and clearly.

Progressive assistance does more than bring the error to the user's attention; the user is guided towards speaking a legal utterance by successively more informative error messages which consider the probable context of the misunderstanding. Repetitiveness and frustration are reduced. One study participant praised our progressive assistance strategy: "When you've made your request three times, it's actually nice that you don't have the exact same response. It gave me the perception that it's trying to understand what I'm saying."

Substitution Errors. Where rejection errors are frustrating, substitution errors can be damaging. If the user asks the weather application for "Kuai" but the recognizer hears "Good-bye" and then hangs up, the interaction could be completely terminated. Hence, in some situations, one wants to explicitly verify that the user's utterance was correctly understood.

Verifying every utterance, however, is much too tedious. Where commands consist of short queries, as in asking about calendar entries, verification can take longer than presentation. For example, if a user asks "What do I have today?", responding with "Did you say 'what do I have today'?", adds too much to the interaction. We verify the utterance implicitly by echoing back part of the command in the answer:

"Today, at 10:00, you have a meeting with..."

As Kamm suggests [4], we want verification commensurate with the cost of the action which would be effected by the recognized utterance. Reading the wrong stock quote or calendar entry will make the user wait a few seconds, but sending a confidential message to the wrong person by mistake could have serious consequences.

The following split describes our verification scheme: commands which involve the presentation of data to the user are verified implicitly, and commands which will destroy data or set in motion future events are verified explicitly. If a user asks about the weather in Duluth, the system will indicate that it is the report for Duluth before reading the contents. The user is then free to regain control of the interaction by interrupting the synthesizer (unfortunately using a touch-tone command in our current implementation). If, on the other hand, the user wants to fax a 500 page mail message, the system will check to make sure that's what was really meant.

Although not its primary purpose, the SpeechActs natural language component, called Swiftus, helps to compensate for minor substitution errors [7]. It does so by allowing the application developer to convert phrases meaning the same thing into a canonical form. For example, the following calendar queries will all be interpreted the same way:

```
What does Nicole have May sixth?
What do Nicole have on May six?
What is on Nicole's schedule May sixth?
```

This means that some substitution errors (e.g., "Switch to weather," misrecognized as "Please weather") will still result in the correct action.

Insertion Errors. Spurious recognition typically occurs due to background noise. The illusory utterance will either be rejected or mistaken for an actual command; in either case, the previous methods can be applied. The real challenge is to prevent insertion errors. Users can press a keypad command to turn off the speech recognizer in order to talk to someone, sneeze, or simply gather their thoughts. Another keypad command restarts the recognizer and prompts the user with "What now?" to indicate that it is listening again.

Challenge: The Nature of Speech

Current speech technologies certainly pose substantial design challenges, but the very nature of speech itself is also problematic. For users to succeed with a SUI, they must rely on a different set of mental abilities than is necessary for successful GUI interactions. For example, short- term memory, the ability to maintain a mental model of the system's state, and the capacity for visualizing the organization of information are all more important cognitive skills for SUI interactions than for GUI interactions.

Lack of Visual Feedback. The inherent lack of visual feedback in a speech-only interface can lead users to feel less in control. In a graphical interface, a new user can explore the interface at leisure, taking time to think, ponder, and explore. With a speech interface, the user must either answer questions, initiate a dialog, or be faced with silence. Long pauses in conversations are often perceived as embarrassing or uncomfortable, so users feel a need to respond quickly. This lack of think time, coupled with nothing to look at, can cause users to add false starts or "ums" and "ahs" to the beginning of their sentences, increasing the likelihood of recognition errors.

Lack of visuals also means much less information can be transmitted to the user at one time. Given a large set of new mail messages or a month's worth of calendar appointments, there is no quick way to glance at the information. One user said: "Not being able to view it - I was surprised at the level of frustration it caused."

To partially compensate for the lack of visual cues, we plan to use both scanning and filtering techniques. For example, during the iterative redesign we added the ability to scan mail headers. We also plan to add functionality so that users can have their mail filtered by topic or by user, and their calendar entries summarized by week and by month. This way, important messages and appointments will be called out to the user first, eliminating some of the need to glance at the information.

Speed and Persistence. Although speech is easy for humans to produce, it is much harder for us to consume [10]. The slowness of the speech output, whether it be synthesized or recorded, is one contributing factor. Almost everyone can absorb written information more quickly than verbal information. Lack of persistence is another factor. This makes speech both easy to miss and easy to forget.

To compensate for these various problems, we attempted to follow some of the maxims H.P. Grice states as part of his *cooperative principle* of conversation [2]. Grice counsels that contributions should be informative, but no more so than is required. They should also be relevant, brief, and orderly.

Because speech is an inherently slow output medium, much of our dialog redesign effort focused on being brief. We eliminated entire prompts whenever possible and interleaved feedback with the next conversational move so as not to waste time.

We also eliminated extraneous words whenever possible. By using a technique which we call *tapered presentation*, we were able to shorten output considerably in cases where we had a list of similar items. This technique basically involves not repeating words that can be implied. In the stock quotes application, for example, when a user asks for his or her portfolio status, the response is something like:

```
Currently, Sun is trading at 32, up 1/2 since yesterday.
           SGI is         at 23, down 1/4.
           IBM is         at 69, up 1/8.
```

With the first stock, we establish the pattern of how the data is going to be presented. With successive stocks, we streamline the presentation by eliminating repetitive words.

Also in the pursuit of brevity and in an attempt not to stress user's short-term memory, we avoid the use of lists or menus. Instead, we use conversational conventions to give users an idea of what to say next. In the calendar application, for example, we always start with "Today, you have..." By initiating the conversation and providing some common ground, it seems natural for users to respond by saying, "What do I have tomorrow?" or "What does Paul have today?"

Ambiguous Silence. Another speech-related problem, also observed by Stifelman [11], is the difficulty users have in interpreting silence. Sometimes silence means that the speech recognizer is working on what they said, but other times, it means that the recognizer simply did not hear the user's input.

This last problem is perhaps the easiest to overcome. Clearly, the user needs immediate feedback even if

the recognizer is a bit slow. We plan to add an audio cue that will serve the same purpose as a graphical watch cursor. This will let users know if the computer is working on their request, leaving silence to mean that the system is waiting for input.

CONCLUSIONS

Based on our experience designing SpeechActs, we have concluded that adhering to the principles of conversation does, in fact, make for a more usable speech-only interface. Just as in human-human dialog, grounding the conversation, avoiding repetition, and handling interruptions are all factors that lead to successful communication.

Due to the nature of speech itself, the computer's portion of the dialog must be both as brief and as informative as possible. This can be achieved by streamlining the design, using tapered presentation techniques, providing short-cuts that make use of another medium (such as touch-tones), and making verification commensurate with the cost of the action.

As with all other interface design efforts, immediate and informative feedback is essential. In the speech domain, users must know when the system has heard them speak, and then know that their speech was recognized correctly.

Finally, we have strong evidence to suggest that translating a graphical interface into speech is not likely to produce an effective interface. The design of the SUI must be a separate effort that involves studying human-human conversations in the application domain. If users are expected to alternate between modalities, care must be taken to ensure that the SUI design is consistent with the corresponding graphical interface. This involves consistency of concepts and not a direct translation of graphical elements, language, and interaction techniques.

While interface challenges abound, we hope that working with speech technology at this stage in its development will provide speech vendors with the impetus to make the improvements necessary for the creation of truly fluent speech interfaces.

ACKNOWLEDGEMENTS

The SpeechActs project is a collaborative effort. Eric Baatz and Stuart Adams have implemented major portions of the framework while Paul Martin and Andy Kehler are responsible for the natural language components. Special thanks to Bob Sproull for his contributions to the architectural design of the system.

REFERENCES

1. Clark, Herbert H. **Arenas of Language Use**. University of Chicago Press, Chicago, IL, 1992.

2. Grice, H. P. "Logic and Conversation," *Syntax and Semantics: Speech Acts*, Cole & Morgan, editors,

Volume 3, Academic Press, 1975.

3. Grosz, Barbara, and Candy Sidner. "Attention, Intentions, and the Structure of Discourse," *Computational Linguistics*, Volume 12, No. 3, 1986.

4. Kamm, Candace. "User Interfaces for Voice Applications," *Voice Communication Between Humans and Machines*, National Academy Press, Washington, DC, 1994.

5. Kitai, Mikia, A. Imamura, and Y. Suzuki. "Voice Activated Interaction System Based on HMM-based Speaker-Independent Word Spotting," *Proceedings of the Voice I/O Systems Applications Conference*, Atlanta, GA, September 1991.

6. Ly, Eric, and Chris Schmandt. "Chatter: A Conversational Learning Speech Interface," *AAAI Spring Symposium on Intelligent Multi-Media Multi-Modal Systems*, Stanford, CA, March 1994.

7. Martin, Paul and Andrew Kehler. "SpeechActs: A Testbed for Continuous Speech Applications" (7 PostScript pages), *AAAI- 94 Workshop on the Integration of Natural Language and Speech Processing*, 12th National Conference on AI, Seattle, WA, July 31-August 1, 1994.

8. Nielsen, Jakob. "The Usability Engineering Life Cycle," IEEE *Computer*, March 1992.

9. Roe, David, and Jay Wilpon, editors. **Voice Communication Between Humans and Machines**, National Academy Press, Washington, DC, 1994.

10. Schmandt, Chris. **Voice Communication with Computers: Conversational Systems**, Van Nostrand Reinhold, New York, 1994.

11. Stifelman, Lisa, Barry Arons, Chris Schmandt, and Eric Hulteen, "VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker, ACM *INTERCHI '93 Conference Proceedings*, Amsterdam, The Netherlands, April 24-29, 1993.

12. Yankelovich, Nicole. "Talking vs. Taking: Speech Access to Remote Computers" (2 PostScript pages). ACM *CHI '94 Conference Companion*, Boston, MA, April 24-28, 1994.

13. Yankelovich, Nicole and Eric Baatz. "SpeechActs: A Framework for Building Speech Applications" (9 PostScript pages). *AVIOS '94 Conference Proceedings*, San Jose, CA, September 20-23, 1994.

Current Addresses:

Gina-Anne Levow
MIT AI Laboratory
545 Technology Square, Room 810
Cambridge, MA 02139
gina@ai.mit.edu

Matt Marx
MIT Media Laboratory

Weisner Building, #E 15
20 Ames Street
Cambridge, MA 02139
groucho@media.mit.edu
