

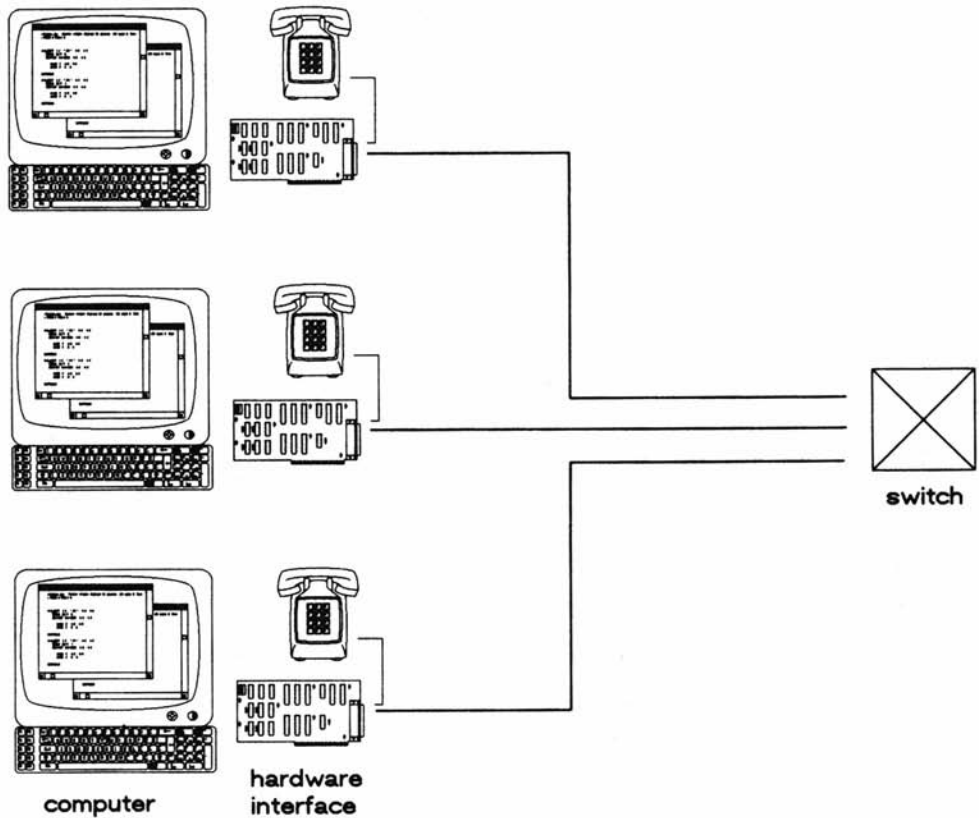


Figure 11.10. A distributed architecture with a telephone peripheral on each workstation.

local area network, sending requests and receiving events (such as notification of an incoming call) from the server. In this centralized architecture, application requests are translated into messages for the server and sent over the computer network; the server, in turn, translates these into commands to its local hardware and thence onto the telephone network. The centralized approach requires that one computer interact with many telephone lines so that a single server can act on behalf of a number of clients. This can be accomplished in several ways: each line can be connected to the server or the server can communicate directly with the telephone switch.

With analog telephony, each line has its own independent circuit for signaling and voice. A multiline analog telephone or key set has a pair of line wires for each line and selects between them with buttons. Similarly, interfacing a computer to multiple analog lines requires the use of one telephone line interface board per line. A single board can be made to handle multiple lines by replicating the interface circuits, which reduces the per-line expense; consequently, such boards are often used in voice mail systems, serving four to sixteen lines per board.

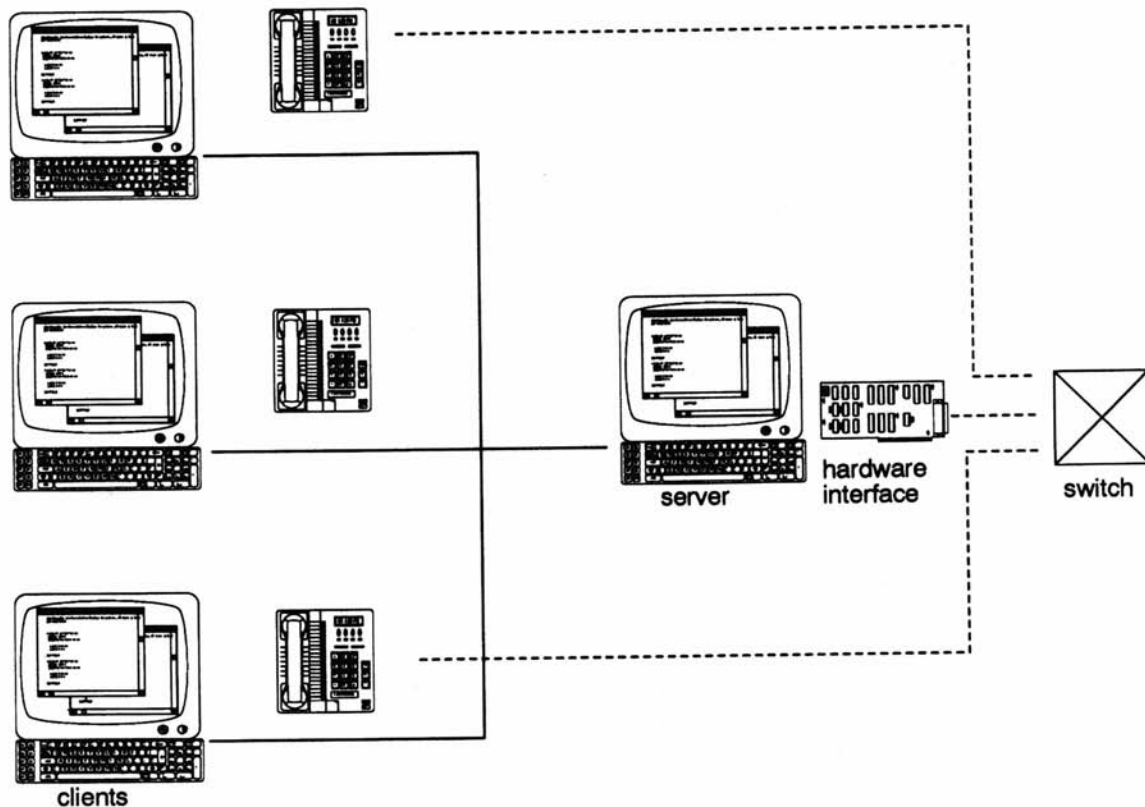


Figure 11.11. A server allows centralization of some resources with a network interface to each workstation.

An alternative arrangement is for the server to communicate with the switch directly through a custom interface. Some experimental systems, such as MICE, use small PBXs designed to be controlled by a host computer. Larger PBXs are less flexible but may still provide a simple computer interface. For example, some switches support a message set called SMDI (Simple Message Desk Interface), which carries selected signaling information from the switch over a serial line. Interfaces such as SMDI are often used by voice mail systems in the following fashion: When a call gets transferred to voice mail, the switch identifies the originally dialed number and a trunk line switches onto that which the call has been transferred so that the voice mail system can record a message into the proper mailbox. The signaling interface is bidirectional so the voice mail system can control the telephone's message waiting light via the switch. Such a PBX interface can extend call control to a single workstation, which communicates with the PBX and acts as a server to other workstations. SCAI (Switch-Computer Application Interface) is a proposed standard for external switch control. It is currently under study by the T1S1 committee of ANSI (the American National Standards Institute).

For digital telephone systems, particularly ISDN, multiple telephone numbers can appear on a single phone line if the switch is configured to support key set style telephones.⁵ The sole D channel for the ISDN line carries signaling information in both directions for all numbers appearing on the line. So if in addition to each user's personal telephone line another line is configured with appearances of each user's number, it can be used by a server to control calls on those numbers. The D channel on the server's line can be used to allow a server to answer, transfer, or place calls by proxy at the request of client applications running on other workstations not equipped with ISDN interfaces. If the server ceases to function, the user's office telephones are unaffected.

Comparison of Architectures

The ISI Phoneserver described earlier is an example of the server approach with centralized hardware. The choice of architecture was dictated by the PBX interface board; this board was expensive and incompatible with the users' workstations in their offices. Both MICE and Berbell used centralized telephone switching because the switching hardware supported a single serial interface. MICE stored call control tables in the server; instead of routing each call, clients programmed the call control tables and the server routed calls on its own. By contrast, Berbell allowed clients to dynamically route calls. MICE and Berbell also used centralized speech resources such as codecs and speech synthesizers. Although this provided adequate support for services to interact with remote callers, a user in an office also had to pick up the telephone to access these devices, e.g., to hear a voice message.

An ISDN-based telephone server similar to Berbell yet lacking built-in voice storage has been built at the MIT Media Lab. This server, used by applications described later in this chapter, allows access to a cluster of phone numbers using normal D channel signaling, obviating special hardware access to the telephone switch. In this environment only call control is centralized; audio services are provided locally at each workstation. Several servers provide specialized services such as a voice mail server to answer calls using the telephone server's B channel interface.

Both PX and Etherphone used fully-distributed architectures that took advantage of additional centralized services. In PX the telephone switch broadcasted information about every call, and the appropriate workstation requested that the switch complete the voice circuit. Although the switch was a central resource, it was designed to support fully-distributed call management, and most importantly the digitized voice was routed directly to the workstation. Similarly, in the Etherphone project, a central conversation manager connected digital audio streams from separate workstations for internal calls, whereas each workstation had complete local control to place outgoing calls and receive incoming calls from

⁵Unfortunately, these features are not published as any ISDN standard, so for now they remain vendor specific.

the public telephone network. PX stored sound locally on the workstation, while Etherphone digitized it and then stored it via a centralized voice storage server utilizing the voice ropes database described in Chapter 4. PX workstations shared a centralized speaker verification service, and Etherphone allowed workstations to have access to a central text-to-speech synthesis service.

From the point of view of hardware and system maintenance, the centralized and distributed architectures differ radically. These hardware differences can be made largely transparent to applications by means of a software library that shelters an application from details of communicating with the remote server. If implemented in such a network-transparent manner, the user is not concerned with whether the server process is on a local or remote workstation.

The server approach is attractive for device sharing and resource arbitration even if hardware devices and application processes are located on the same workstation. Device sharing promotes the development of environments with multiple small applications rather than a single monolithic application: a server allows each application to operate without knowledge of other applications that may require sporadic access to limited resources. Multiple applications may receive simultaneous notification of an incoming call; one application might alert the user using speech synthesis to name the caller or displaying the caller's name in a window on the workstation. Concurrently, an answering machine (voice mail) application waits a specified amount of time after the call is announced and picks up if the called party does not answer. Another application may keep a log of all calls, incoming or outgoing. A server architecture promotes such sharing of hardware resources, whether local or remote.

The differences between centralized and distributed architectures is more significant for voice storage, transmission, and presentation to the user in an office. If the telephone line terminates in the workstation, then when the workstation answers a call the voice circuit terminates locally. Under this architecture, an application might allow an incoming voice message to be monitored on the local workstation speaker for call screening and also permit the called party to pick up the phone in the middle of taking a message. Locally stored voice messages⁶ can be played through the speaker without needing to re-establish an audio link to the server.

When the telephone line terminates in a remote location, monitoring message recording is more difficult and call pickup may not be possible. Losing the ability to screen calls is a common complaint among users when voice mail replaces answering machines. If voice is stored on a central resource, then it can be transmitted over the computer network (as was done in Etherphone) in order to be played. MICE used the existing telephone wiring to playback voice messages by first placing a local call from the central storage manager to the telephone in the office. This required the user to answer the phone to hear the message, making

⁶The actual disk drive on which the voice is stored may be in another location remotely mounted over the local area network. But this is transparent to the application and the sound appears as a local file system.

message playback much less spontaneous. Although we are used to the telephone as the means to retrieve voice mail, when stored voice permeates many desktop applications such as described in Chapter 12, audio playback through local speakers is much more desirable at least for use in a private office.

CASE STUDIES

This section describes several sample projects from the MIT Media Lab. These case studies are intended as examples of the services and user interfaces that can result from the integration of telephony applications with computer workstations. Some of the case study examples contain similarities to the work on integrating computers and telephones discussed earlier in this chapter as many of the ideas have been explored across multiple research labs.

Phone Slave

Phone Slave was an early (1983) integrated telephone management system developed by the MIT Architecture Machine Group (predecessor to the Media Lab) [Schmandt and Arons 1985, Schmandt and Arons 1984]. Phone Slave provided a variety of telephony functions including message taking, personal outgoing message delivery, a telephone number and address database, speed dialing by name, and remote access to both voice and text mail. With the breadth of functionality it supported, Phone Slave attempted to provide a complete interface to telephony functions from a general-purpose computer.

Phone Slave included both graphical and telephone-based user interfaces. It did not run in concert with any window system as window systems were not in general use at that time. Phone Slave was a monolithic process; in subsequent work at the Media Lab, many of Phone Slave's functions were implemented as separate applications (two examples, Xphone and Xrolo, are described later in this chapter). In the absence of a window system, Phone Slave shared the single graphics screen between three different display views: one for message retrieval, one for dialing on a telephone keypad, and the last for the name, address, and number database.

Phone Slave adopted a conversational approach to taking a telephone message. It asked a series of questions and recorded each response in a separate file. During recording, an adaptive pause detection algorithm determined when the caller was finished; the algorithm was adaptive with respect to the background noise level as well as the speech rate of the caller. The sequence of Phone Slave's questions was as follows.

- "Hello, Chris's telephone, who's calling please?"
- "What's this in reference to?"
- "I'm sorry, he's not available now, but he left this message . . ." followed by the owner's outgoing message and then "At what number can he reach you?"
- "When will you be there?"

- “Can I take a longer message for you?”
- “Thanks, I’ll give him your message, goodbye.”

Some naive subjects did not realize that they were speaking to a computer (note the wording of the initial greeting). Of course, there were instances when the conversational model would break down, which most commonly occurred when the caller responded simply with “yes” to the last question. The caller proceeded to wait for an anticipated prompt to leave a message and was surprised by the closing sign-off announcement. Each recorded comment with the exception of the last had a maximum anticipated response duration; once this was exceeded, Phone Slave would interrupt, saying “Excuse me, I’m just an answering machine, and it would be helpful if you could answer my questions directly.”

Phone Slave’s conversational format served several purposes. The first was to make it easier for the caller to leave a complete message. At the time when Phone Slave was implemented, answering machines were uncommon and seldom found in business situations so not all callers were accustomed to them. The second purpose was to provide segmentation of the messages for the sake of the recipient, Phone Slave’s owner. Finally, during the caller’s response to the first question (“*Who’s calling please?*”), the telephone line audio output was routed to a speaker-dependent speech recognizer that attempted to identify the caller. The caller had to speak a consistent greeting to be recognized as the recognizer merely detected the familiar pattern of words.

A caller might be identified by Phone Slave, either by speech recognition or by entry of a touchtone password at any point in the conversation because speech recognition over long-distance telephone lines tended to be unreliable. If recognized, the caller was greeted by name, informed if Phone Slave’s owner had listened to any recent messages from the caller, and possibly received a personalized outgoing message. After this, Phone Slave would say “If you’d like to leave another message, I’ll record it now. Otherwise, just hang up and I’ll tell him you called again.”

When the owner called in and logged in, he could take advantage of a number of Phone Slave options including hearing messages, deleting messages, changing the outgoing message, and recording personal messages for known callers. Commands (e.g., Play, Delete) were entered either by speech recognition or touch tones. Because the incoming messages were segmented by the conversational recording method, several novel review options were available. For example, messages could be scanned quickly by asking “Who left messages?”; the system responded by playing back the initial recorded segment for each message in which the callers would have identified themselves.

Unlike conventional voice mail, Phone Slave integrated voice and text (electronic mail) messages. Text messages were presented by speech synthesis using the user interface described in detail in Chapter 6. Synthesized and recorded messages were intermixed, as Phone Slave sorted messages according to the sender (rather than the conventional time-sequential order) to facilitate complete responses.

In addition to taking messages, delivering personal messages, and providing mixed-media message retrieval over the telephone, Phone Slave provided screen-

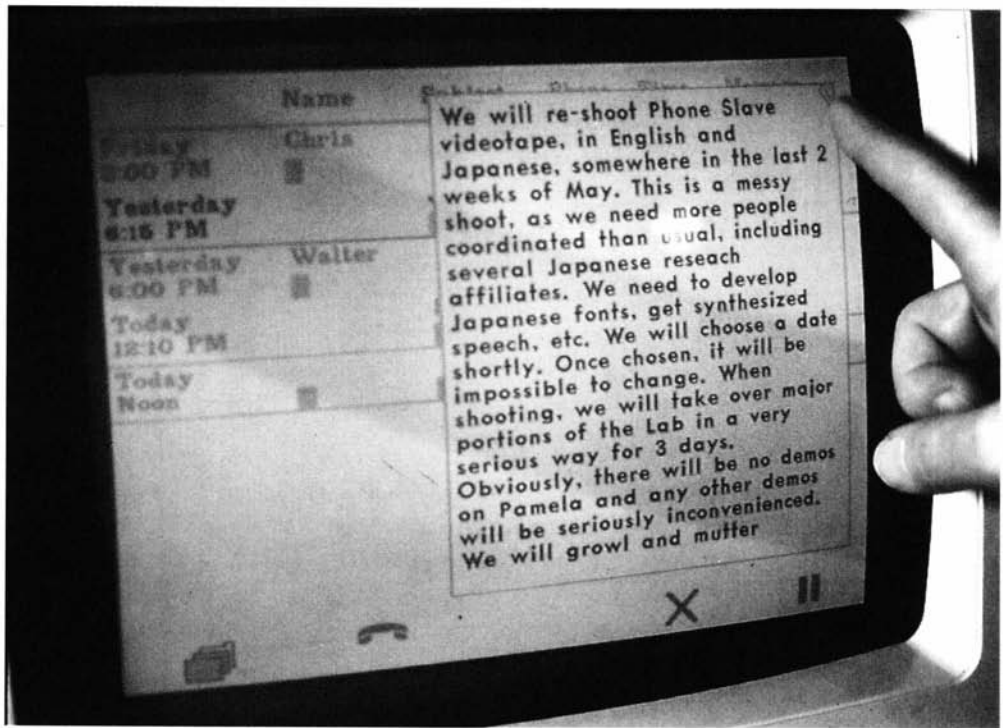


Figure 11.12. Phone Slave used a pop-up window to display a text message. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the 1984 *SID International Symposium Digest of Technical Papers*, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146-149.

based message access as well as other telephony functions using a color display and a touch-sensitive screen. Phone Slave implemented a limited set of window system functions, sharing the screen between a number of "views" and using pop-up windows for email display (see Figure 11.12). The message access view, for example, displayed each message with text fields to indicate the caller (if known), date and time, and subject field (for a text message). Each recorded sound segment appeared as a bar (see Figure 11.13), whose length indicated the duration of sound. Each message was displayed as a row of sounds; touching the left edge of the row would play each message segment in sequence. The bar changed color during playback from left to right in synchronization.⁷ Touching a text message would pop up a text display window that could page through the message (see Figure 11.12).

⁷This sound bar was a direct predecessor of the SoundViewer widget for the X window system described in Chapter 12.

	Name	Subject	Phone	Time	Message
Friday 2:00 PM	Chris				
Yesterday 6:15 PM		Japanese Video Tape			
Yesterday 6:00 PM	Walter				
Today 12:10 PM		Shadow Masks			
Today Noon					






Figure 11.13. Phone Slave's message access view. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the *1984 SID International Symposium Digest of Technical Papers*, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146-149.

Phone Slave could place calls in several ways. The simplest method involved bringing up the telephone dialing view by pressing the telephone icon at the bottom of any other view (see Figure 11.14). The dialing view was modeled after a telephone keypad with an area for dialed-digit display. As with Phonetool discussed earlier in this chapter and Xphone introduced in the next section, Phone Slave accumulated the appropriate set of digits for a complete telephone number before commencing to dial, which allowed the user to delete digits without needing to hang up and restart from the beginning. While a call was in progress, the handset in the dialing view was raised to indicate the off-hook state; to hang up, the user would touch the handset, which would drop it back in place.

Phone Slave also included a simple name, address, and telephone directory, which was displayed in a view that listed each entry as an alphabetized card. This database was created in a text editor; Phone Slave allowed read-only access to it. In addition to the expected fields on the card, such as address and telephone number, any personalized outgoing messages were indicated by the presence of a

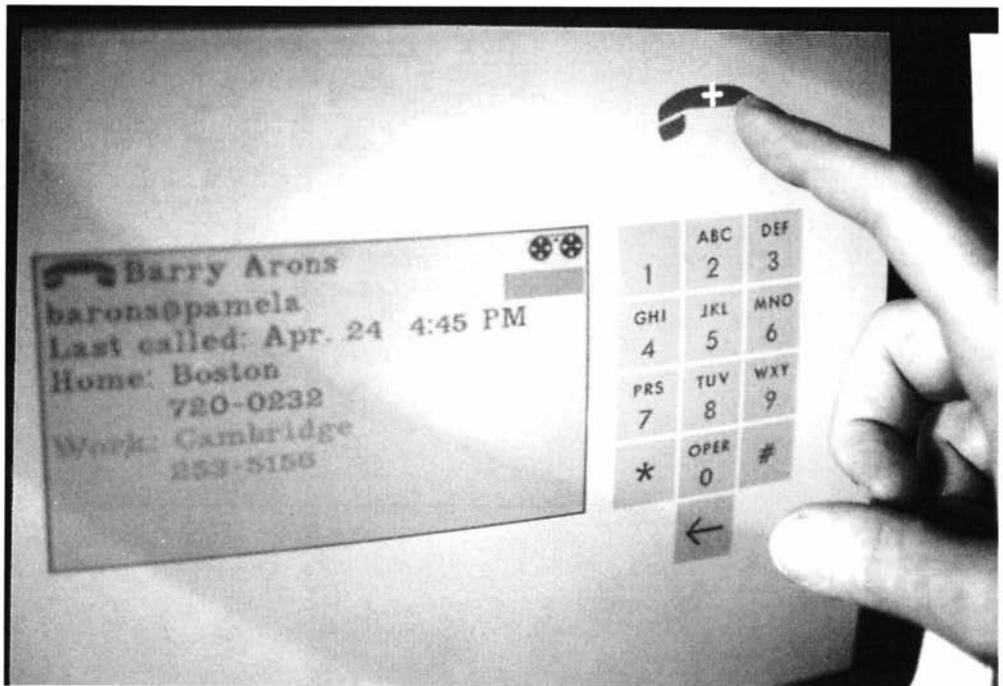


Figure 11.14. Phone Slave's telephone dialing view. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the *1984 SID International Symposium Digest of Technical Papers*, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146-149.

sound bar on the card. The owner could review the outgoing message by touching the bar or create new messages by touching on the "reel of audio tape" icon. Depending on the time of day, either the home or work number would be highlighted in red. Touching the telephone icon on the card would switch to the telephone keypad view and automatically dial the number.

A special card was dedicated to choosing the generic outgoing message for an unknown caller (see Figure 11.16). The owner could select from a number of previously recorded messages suitable to particular occasions (out to lunch, out of town, etc.) or record a new temporary message. This option was included as part of the overall project goal of maximizing the information content of a transaction with an answering machine; as much information as possible was provided to both known and unknown callers.

Phone Slave was a direct predecessor of several related projects that followed in the next seven years. The answering machine portion was included in the Conversational Desktop project described in Chapters 9 and 12. Screen-based voice mail has recently been resurrected in a multiprocess version running under the X Window system. Separate processes manage message taking, the graphical

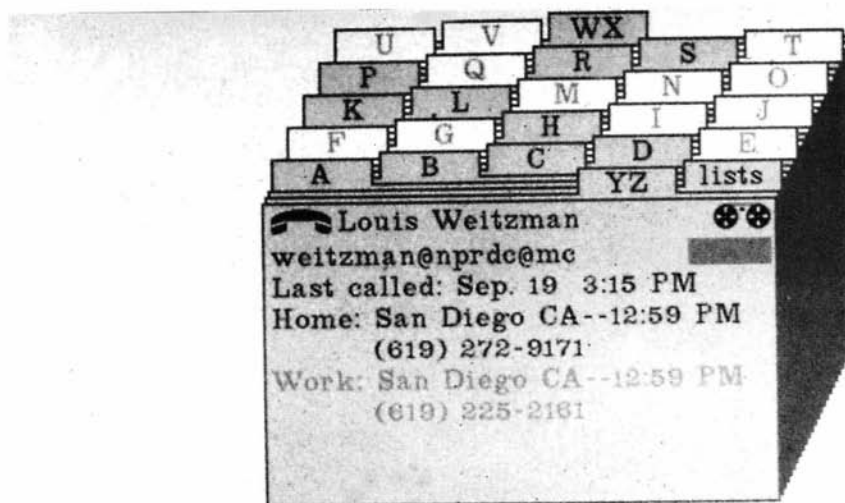


Figure 11.15. Phone Slave's name, address, and telephone number directory view. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the 1984 *SID International Symposium Digest of Technical Papers*, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146-149.

user interface, and remote telephone access to messages; this is described in more detail in Chapter 12. Also, Xphone, a telephone dialer, and Xrolo, a telephone directory application, provide the functionality of two of Phone Slave's views.

Xphone and Xrolo

Xphone is a telephone dialing tool based in large part on the Phonetool dialing tool developed by Stephen Casner and described earlier in this chapter. Xphone runs under the X Window system and offers increased functionality over the original Phonetool [Schmandt and Casner 1989]. The basic idea remains the same: a powerful telephone dialer that occupies minimal screen real estate.

Xphone is displayed as a small window showing the most recently dialed telephone number (see Figure 11.17). There are a number of different methods of

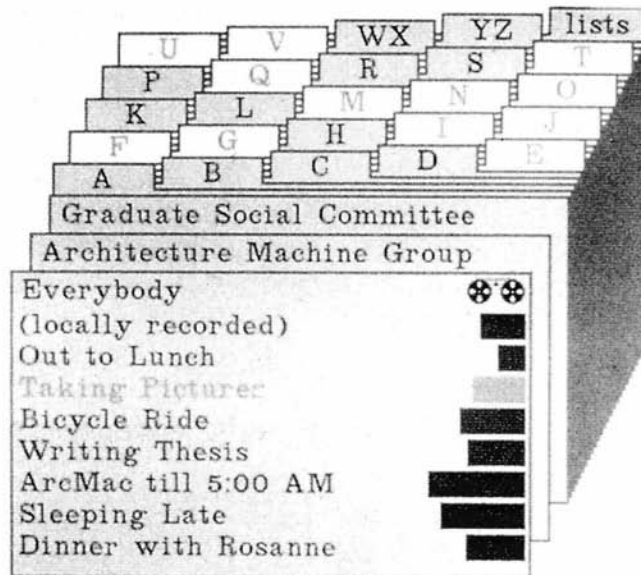


Figure 11.16. The outgoing message was selected using a special card. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the 1984 *SID International Symposium Digest of Technical Papers*, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146-149.

dialing a number with Xphone. The most recently dialed number can be redialed by clicking the left mouse button on the window. A number can be selected with the mouse from a different text window; when it is pasted into Xphone, the number is dialed. Alternatively, the user can type a number into the tool using either the number keys or numeric keypad on the computer keyboard. The keyboard-entry method is not as comfortable for dialing as the telephone keypad because the keys are arranged in a different order, but once the number has been entered, it can be dialed again much more quickly either via redial or the phone log described shortly.

Holding down the right mouse button pops up the speed-dial menu (see Figure 11.18). This menu displays the names found in a simple text file. A submenu reveals a log of recently dialed numbers (see Figure 11.19). Note that an entry in the log may appear as either a name or a number depending on how it was dialed.



Figure 11.17. The Xphone window displays the most recently dialed telephone number.

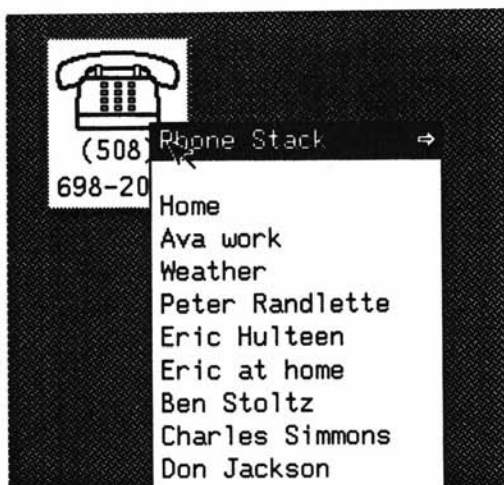


Figure 11.18. The Xphone speed-dial pop up menu. Selecting an entry places a call to the associated number.

An associated name, address, and phone number database tool, Xrolo, provides several other means of dialing. As with Xphone, Xrolo occupies limited screen space until activated, which is done either by typing a name into the small window or by holding down a mouse button to pop up a menu of names (see Figure 11.20). Once chosen, a card appears with relevant information in various fields, each of which can be edited (see Figure 11.21). The search may be based on personal name, computer login, or company. Related applications can generate facsimile cover sheets, long and short address lists, and mailing labels. There is also a text-based version of the tool for use over dialup lines.

Although the lines of text on an exposed card are independent objects and may each be edited individually, the application understands that there is a relationship among them. For example, an address usually consists of a name, a company

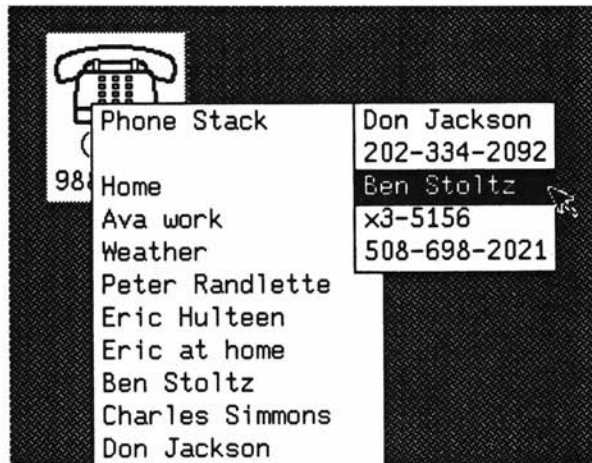


Figure 11.19. The Xphone call-log pop up menu allows recently called numbers to be redialed.

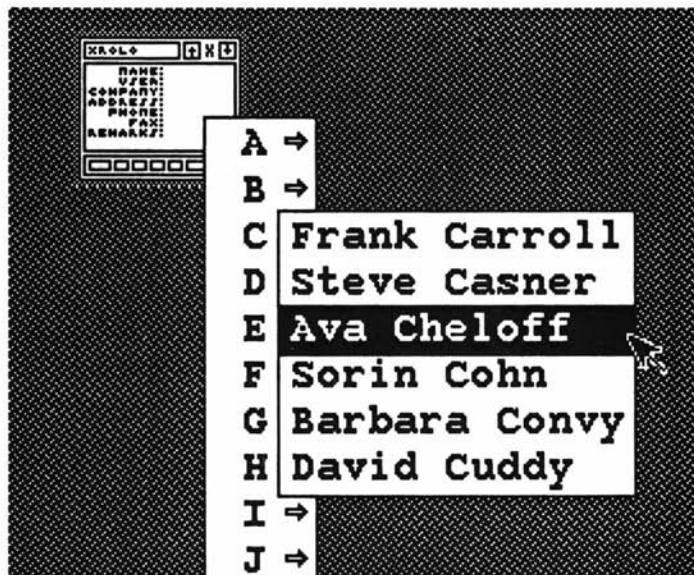


Figure 11.20. Pull down menu of names in Xrolo.

name (if it is a work address), street location, city, state, and postal code. A mouse click on the *work address* button selects all the relevant lines and highlights them appropriately for feedback to the user (see Figure 11.22). This selection may then be deposited by mouse click into another application.

Xphone and Xrolo interact with each other in several ways; the simplest and most common interaction is generating a phone call from the number on a per-

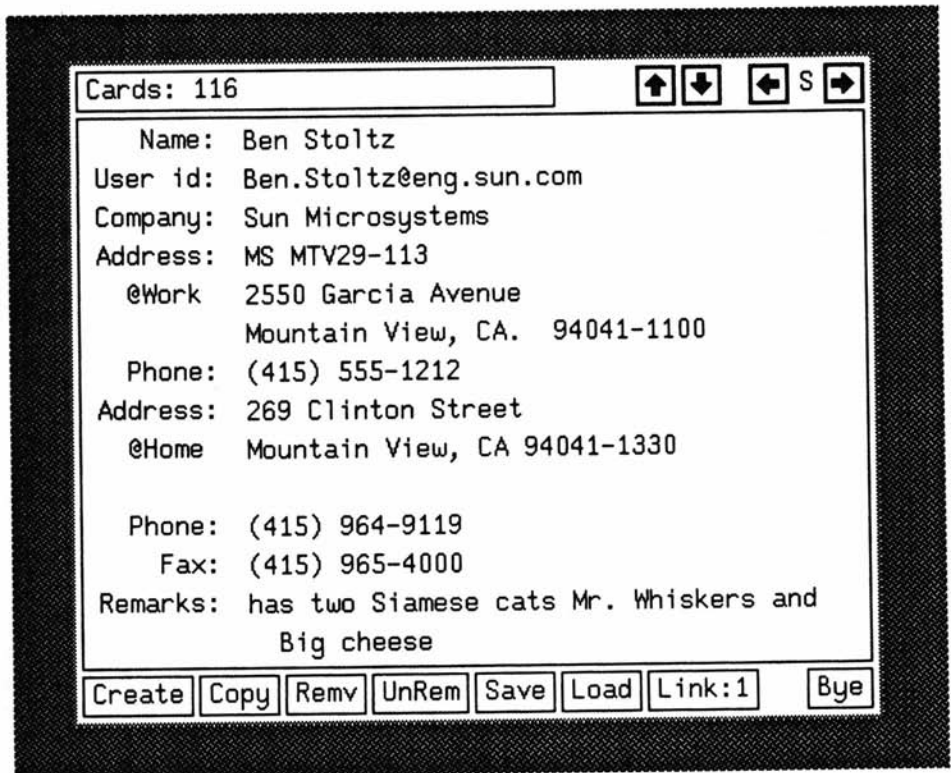


Figure 11.21. A name and address “card” from Xrolo.

son’s Xrolo card.⁸ Clicking the middle mouse button on a phone number sends a message to Xphone and dials the requested number. A number dialed in this manner would appear as a name in Xphone’s phone log menu. A person’s name or email address can be stuffed into Xphone just as easily as a phone number, which presents a more involved method of dialing by name. When Xphone parses the string containing a name, it fails to find a valid number and consequently sends a request to Xrolo to search for an entry that matches the string. If a match is found, Xphone obtains the associated telephone number and dials it and the name of the called party is noted in the call log.

Flexible Call Routing

This case study describes a series of interrelated, small-scale Media Lab projects that have a common theme of flexible management of incoming calls. At

⁸Xphone and Xrolo communicate via the X selection mechanism, which provides a rendezvous point and data exchange through the X server.

Name:	Ben Stoltz
User id:	Ben.Stoltz@eng.sun.com
Company:	Sun Microsystems
Address:	MS MTV29-113
@Work	2550 Garcia Avenue
	Mountain View, CA. 94041-1100
Phone:	(415) 555-1212
Address:	269 Clinton Street
@Home	Mountain View, CA 94041-1330
Phone:	(415) 964-9119
Fax:	(415) 965-4000
Remarks:	has two Siamese cats Mr. Whiskers and Big cheese

Figure 11.22. Highlighted text of the fields associated with *work address*.

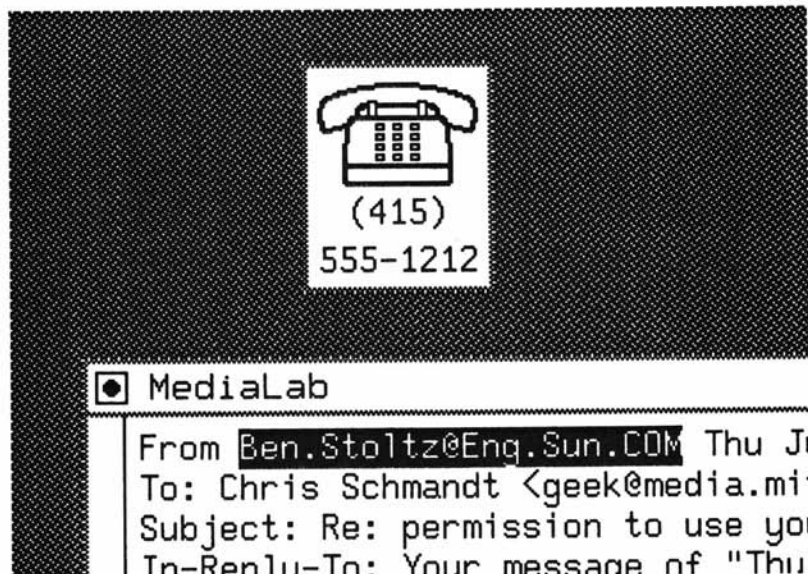


Figure 11.23. Placing a phone call by name.

issue is the potential for computers through their participation in call setup to more effectively connect parties who need to speak to each other. At the same time it is essential to minimize the degree to which users are interrupted by unwanted calls if subscribers are to fully embrace the emerging mobile telephone technologies.

There is little doubt that although nearly all of us view the telephone as an essential part of our lives, we all also experience its annoying aspects as well. We resent unsolicited telemarketing calls at home during dinner or misdialed wrong-number calls in the middle of the night. We are annoyed when a face-to-face conversation in an office is interrupted for a call and are frustrated to wait in line in a hotel lobby while the reception clerk at the front desk takes a message for a guest. When we work alone in our offices, we may resent the interruption of the phone "ringing off the hook" all day; at home we often use answering machines to screen calls (a feature which disappears with centralized voice mail). Yet we rarely disconnect our phones, zealously play phone tag with those we wish to contact, and are disappointed with our few friends who still do not have answering machines.

The popularity of pocket cellular telephones⁹ lends credence to the telephone operating companies' dreams of personal communication networks (PCNs) in which we each carry a phone and are always accessible via a universal personal number. Although some intense telephone users are ready for such services today, most of us shy away from the concept of always being reachable. If computers can become involved in call setup, how can they personalize and filter calls to make such services more attractive? This case study discusses some techniques that try to answer these questions based on alternate means of announcing incoming calls, user interfaces allowing manual intervention by either the caller or called party to route calls more effectively, and finally, automatic call filtering and routing by means of a personalized user agent.

The ring of the telephone is pervasive and penetrating, yet conveys only the minimal information that someone dialed a particular telephone number. With the gradual acceptance of calling party identification, consumer devices have become available that display the number of each caller, and many business PBX telephones similarly indicate this information on a small display. Unfortunately, the display built into the telephone is usually difficult to read, and it requires us to remember the phone numbers of all the people we wish to speak with. By the time our gaze is diverted to the telephone set, we have interrupted whatever task we were performing when the call came in. At MIT, telephones may ring differently for on-campus and off-campus calls, but this too is only marginally useful (contrast this with Etherphone's caller-specific ring tunes, for example.) Computer-based visual or audible alerting may minimize the interruption.

⁹In early 1993 the number of cellular telephone subscribers in the United States passed 10 million. There have been suggestions that poorly wired regions of Eastern Europe should bypass the local loop and go directly to universal cellular service.

Alternate alerting at the Media Lab is performed by a variety of workstation programs that act as clients of a centralized telephony server. This server uses a single ISDN interface to send notifications to clients (e.g., incoming call attempts) and can route calls at their request by performing call transfer operations. On-campus calls include calling number information, which the server translates to a name from a database.

One client performs visual alerting, and several others announce calls using synthesized speech. Visual alerting pops up a window on the user's display; as shown in Figure 11.24, this window indicates the caller's name (or "unknown") and disappears when the call is answered or the caller hangs up. Audible alerting in the office consists of a speech synthesizer announcing the caller's name. Additional audible alerting is provided in a large common lab work space where the synthesizer announces both the called and calling parties over a public address system; a similar service was provided under BerBell. Someone with an office near the lab space who is paged may be able to reach the telephone in time to take the call. But a workstation user can also take advantage of a variant on the visual call alert service by sending the call to the nearest telephone. This form of alert window shown in Figure 11.25 includes buttons that will route the call if the user selects one with the mouse.

Finally, some students and staff wear "Active Badges" that include several buttons. The Active Badges from Olivetti Research Laboratory use infrared signals to periodically transmit their identity to a network of sensors in strategic positions around the building [Want and Hopper 1992]. By pressing one of the buttons, a person is also able to route the call to the nearest telephone; this service has the advantage of being accessible even if the called party is not logged on to a workstation. Badges can also receive messages from the sensors to sound a tone or illuminate an LED (this provides yet another means of alerting a mobile

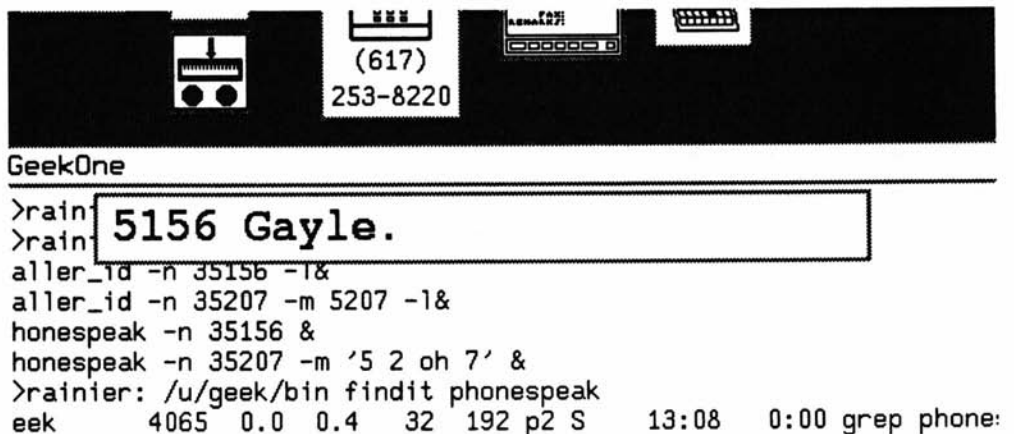


Figure 11.24. Window-based visual call alerting displays the name of the calling party.

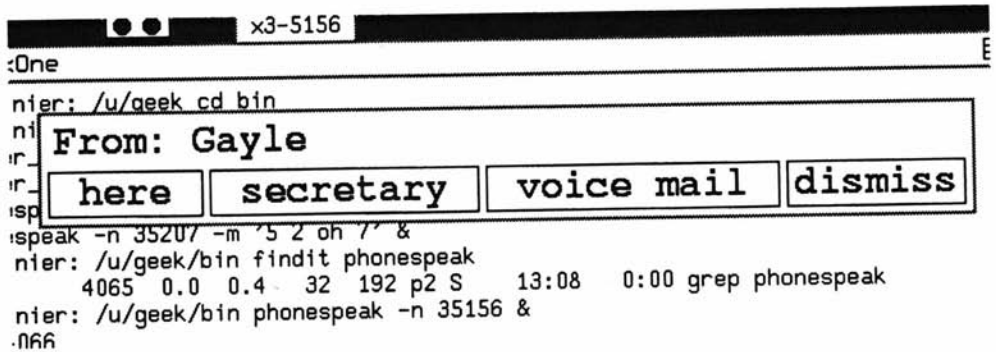


Figure 11.25. When routing buttons are added to the visual alert window, the called party can dynamically forward calls when working in another office.

user to an incoming call). At the Media Lab, an Activity Server combines badge-sighting reports with information about which users are logged in to which workstations on the local network (the “finger” command) to track users’ activities and locations even when they are not wearing badges. The Activity Server also receives events from the telephone server and is notified when any lab telephone is in use.

Knowing colleagues’ locations and activities helps local lab callers judge whether to interrupt someone with a phone call and know that person’s location if they do proceed to call. A graphical user interface, *Watcher* (shown in Figure 11.26) displays users’ physical locations and states, e.g., “alone in own office,” “on the phone,” or “in a meeting” (with other badge wearers) along with the number of the nearest phone. Clicking on the *phone* button sends a message to Xphone causing it to call the displayed number.

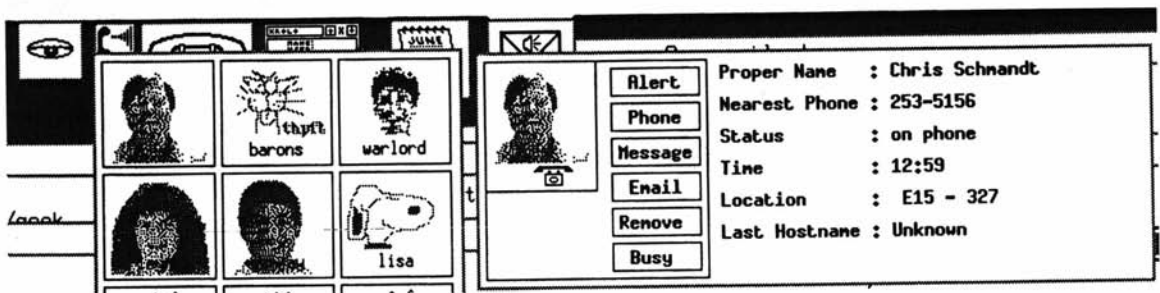


Figure 11.26. *Watcher* shows the location and activity of members of a workgroup. Buttons in the detail windows choose different means of sending messages to the selected person.

An interface such as Watcher helps the caller decide whether and where to phone, but it is available only to someone who is calling from a workstation in the lab.¹⁰ The flexible alerting and call forwarding mechanisms just discussed enable the called party to manually intervene in call routing in response to the stimulus of an incoming call but depend on the availability of additional hardware such as a nearby workstation screen or a badge. A similar style of hardware-based remote intervention using pager technology is described in [Kramer *et al.* 1993]. Ideally one's personal workstation would make routing decisions automatically based on knowledge of the identity of the caller, the time of day, the location and activity of the called party, and whether a secretary or receptionist was available to take the call if it otherwise went unanswered. A person's location and activity may be derived from the dynamic information gathered by an entity such as the Activity Server, but personal calendars also give clues as to travel plans and meetings scheduled in advance.

A Media Lab thesis project [Wong 1991] explored the feasibility of call forwarding based on user-authored rules conditioned on the routing factors just mentioned. The graphical user interface shown in Figure 11.27 was used to create rules. A rule was defined by clicking with the mouse on icons representing four classes of conditions shown on the left of the figure; at this point, text-based dialog boxes appeared that allowed the user to complete the condition, e.g., specifying the time of day or name of the calling party. Each rule routed calls to destinations depicted on the right; these could be either a predefined phone number (e.g., a secretary's desk), voice mail, or a more dynamic location (e.g., the nearest available telephone). A knowledge-based "electronic receptionist" (from Bellcore) without a graphical interface is described in [Gifford and Turock 1992], which presents usage data supporting the position that automatic routing of some telephone calls may provide a valuable service.

Although initially promising, this method of defining call routing actions was problematic for several reasons. The graphical user interface was designed to shield users from the syntax and other details of the text-based rule language used to specify call routing. Although it was fairly effective for creating rules, the interface turned out to be unwieldy for modifying or deleting rules; it was difficult to graphically display a rule, especially one with multiple conditions for activation. The graphical interface did not provide capability for testing and debugging rule sets.

It may also be difficult for users to match their mental models of how the phone should behave with the operation of such an explicitly rule-based approach. Rules may conflict. For example, if I have one rule that says that after 5 P.M. calls forward to voice mail and another which says that calls from my daughter always forward to the nearest phone, then what should happen when my daughter calls

¹⁰Actually through the Phoneshell service described in Chapter 12, a knowledgeable subscriber can phone in, discover where other users are located, and then transfer the call to that location.

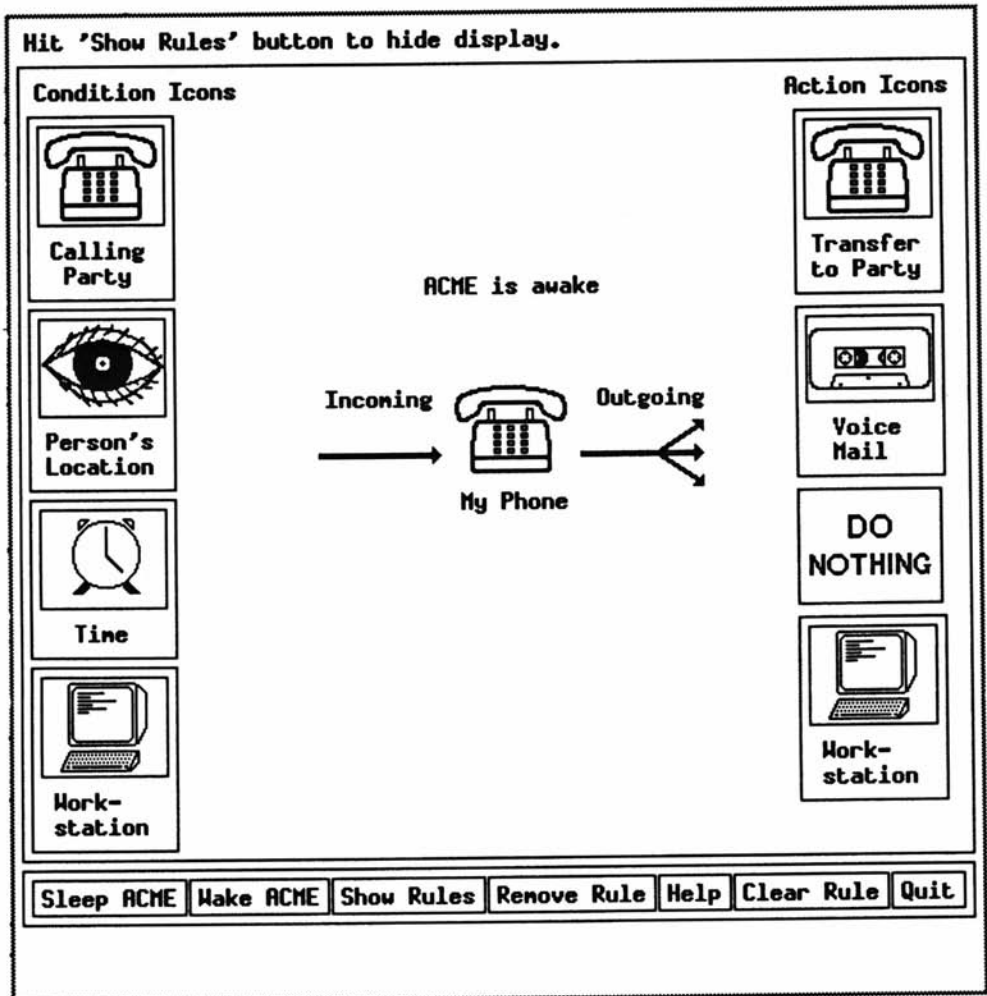


Figure 11.27. The graphical user interface to a rule-based call router. The user selected conditions on the left and associated them with routing destinations on the right.

at 5:30? This might be resolved by saying that rules involving the identity of the calling party take precedence over rules based on other conditions as telephone calls are more highly associated with the person calling than the time of day, for example. But what should happen if I also have another rule that says that I am never to be disturbed when in the conference room and when my daughter calls at 5:30 that is my location? Flexible call routing may be better performed by software that learns my preferences by observing my behavior over time, and this requires a feedback mechanism to retrain such software whenever it makes an incorrect decision.

SUMMARY

This chapter was about computers performing telephone management tasks. Computers can place calls on behalf of a user, answer the telephone, or decide how to route an incoming call. This chapter began by making the case for computer-mediated telephony on the basis of its providing superior user interfaces than telephone sets, facilitating a more integrated overall electronic communications perspective, and extending the variety of call management services to which users may subscribe.

To place, receive, and route telephone calls, computers require hardware interfaces capable of communicating with the telephone switch. In some circumstances it may be appropriate to place such hardware on each computer in every office; this is a fully-distributed architecture. If telephone interface hardware is excessively costly or not compatible with the workstation in each office, a centralized approach may be preferable. With a centralized architecture, a single server computer with appropriate hardware manages telephones for many client computers. In either hardware scheme, centralized or distributed, a software-based server approach allows multiple client applications to participate in telephone management and effectively insulates application software from the details of the physical arrangement of hardware.

The concept of computer management of telephone calls was illustrated by a number of example projects discussed in different levels of detail. Etherphone and PX utilized distributed architectures to control flexible call switching entities, while MICE and BerBell explored the potential for client-based call control using centralized hardware similar to a telephone company's central office switch. Phone Slave presented a unified call management user interface including sophisticated answering machine capabilities. Xphone and Xrolo are examples of more modern software architectures and illustrate how multiple applications may coordinate with each other to provide enhanced access to multiple communication functions. Finally, a variety of small applications were discussed together under the theme of computer routing of incoming calls, either manually or automatically, and with or without the cooperation of the calling party.

The variety of functionality computers may bring to telephone management motivates and underscores the research projects discussed in this chapter. Some examples included enhanced message-taking features as well; once messages have been recorded, computer users can benefit from improved user interfaces to voice mail and may use these messages in other applications as well. This theme is continued in the next chapter, which discusses uses and interfaces for managing stored voice on the desktop.

Desktop Audio

Because of limitations in speech technology, its successful deployment has been limited to certain situations in which voice offers unique advantages over visual interfaces or in which no other interface modality was practical. This bias has been reflected in the case studies as well, with their emphasis on hands-and-eyes busy environments or tasks oriented around managing voice communication with other people. This chapter instead focuses on applications of voice in an ordinary desktop computing environment used for day-to-day office activities. To continue the theme from the end of the last chapter, once computers manage our telephones and take voice messages for us, what else can we do with this stored speech?

Although nearly every office is equipped with a computer as well as a telephone, desktop voice processing is only beginning to become a reality. This gap is due in large part to the lack of a convincing argument for desktop audio; much office work is handled effectively with the use of keyboard and mouse to access text-based applications. To appreciate the potential of voice applications in the office we must first understand how speech can benefit office work as well as the synergy of using voice across a range of applications instead of in an isolated niche.

For decades computers were used to perform one task at a time; now window systems and multiprocessing operating systems give the user the ability to interact with many applications simultaneously. What is true with text-based software will be even truer with voice: no single application will dominate computer usage or provide sufficient benefit to warrant the purchase of voice processing capabilities. Rather in the future we can expect families of voice-capable applica-

tions running on the desktop sharing data and interaction techniques. Equally important to the need to share data among voice applications are the means to interchange information between voice and text databases and utilize presentation methods that allow telephone access to the desktop and other portable devices.

This chapter describes the concept of **desktop audio**, an environment that supports multiple voice applications on a single desktop computer and remote voice access to office databases. These ideas have already been touched upon in previous chapters: Xspeak (see Chapter 8) used speech recognition at the desktop to coordinate window-based interaction with multiple applications, Phone Slave (see Chapter 11) included a variety of telephone-related functions in a single application, and Voiced Mail (see Chapter 6) provided an integrated remote interface to voice and text messages.

This chapter considers several aspects of desktop audio. It first explores strategies that can smooth the transition from text-only to multimedia computing in the office and then presents a variety of graphical user interfaces to stored voice, which can help overcome its slow and serial nature in the desktop environment. The text advocates a client-server based software architecture to support multiple voice applications simultaneously. The notion of voice as a means of capturing much of our normal office conversations for later retrieval is discussed. In closing, the chapter presents a number of case studies to illustrate these desktop audio concepts.

EFFECTIVE DEPLOYMENT OF DESKTOP AUDIO

Most new workstations¹ are now equipped with a speaker and a microphone and increasingly fast processors allow software-based implementations of speech recognition, text-to-speech synthesis, audio compression algorithms, and time-scale modification to intelligibly play voice back in less time than originally spoken. Yet deployment of voice applications has proceeded slowly and lags behind the underlying technology. This delay is due in part to the inability of software developers to target applications for which voice offers a benefit to the user, in part due to the difficulty of developing user interfaces that overcome the slow and serial nature of voice as data, and in part due to software architecture constraints interfering with the development of applications to be portable across a variety of hardware platforms.

From the desktop audio perspective, much of the difficulty lies in the lack of any single ideal voice application. To date the most prevalent speech application is voice mail, but current stand-alone implementations of voice mail are deployed and efficiently managed as part of the telephone system, not a networked com-

¹The term "workstation" is used in this chapter to refer to any desktop computer. With ever-increasing microprocessor speeds, the distinction between "workstation" and "personal computer" is becoming increasingly blurred.

puter resource. The delay in proliferation of voice applications is exacerbated by the lack of integration between new voice-capable software and existing applications. If each speech application uses its own distinct user interface, it is harder still on the user to access multiple applications. The absence of a single "killer" desktop voice application does not mean speech will not be useful but it does suggest that it is essential to consider how speech applications will work with each other, how voice will enhance existing workstation applications, and what unique new capabilities voice will enable.

To succeed voice must offer supplemental value to the existing productivity tools on workstations rather than replace those tools. Users have not been willing to give up functionality or applications that they are already using for the sake of adding voice; voice capability must instead augment these applications as well as introduce additional new ways in which to use the computer. Augmenting instead of replacing functionality can be accomplished by maintaining compatibility with current text-based databases and applications wherever possible. For example, the multimedia mail system developed as part of Carnegie Mellon University's Andrew project [Morris *et al.* 1986] added the ability to include nontextual enclosures such as animations within Email messages. Users of a non-Andrew system could view the text portion of the message simply as conventional email without the animation, enabling Andrew users to continue to exchange messages with the non-Andrew Email users. But Andrew multimedia messaging never caught on in the larger non-Andrew world because it was so tightly embedded with the rest of the Andrew environment and hence not very portable. Recently, the MIME message format [Rose 1993] has been proposed for use in the Internet community to allow a broader exchange of multimedia messages in a heterogeneous operating environment. MIME is based on the desire to make multimedia capabilities available across the Internet without any modification of mail transport protocols or software and includes software to add minimal multimedia support to many mail reader programs across many operating systems. Rather than the all-or-nothing approach of Andrew (and similar multimedia mail systems), MIME attempts to add new media ability while minimizing disruption to traditional text-based mail. Desktop voice applications must do the same.

But even if voice can be added to the desktop with minimal disruption, what new capabilities will it provide? Chapter 4 already discussed the expressive richness of voice and its useful role as a document type in and of itself. With an appropriate graphical representation, voice can be mixed with media such as text and image as well and even moved between applications. Ordinary office conversations and meetings can be recorded for future use allowing this important record of decision-making activity to be archived and shared. Perhaps most powerful, speech user interfaces will allow computers to be accessed in new, nontraditional work environments.

Integrating telephone access with a unified voice application environment can result in a synergistic increase in the utility and need of voice as data. Speech synthesis can translate text into voice, enabling access to many text databases over voice telephone connections. This functionality will prove most valuable for retrieving timely personal information while traveling or working away from the

office; databases may include telephone numbers and addresses, one's personal calendar, and electronic mail. Although voice access will not eliminate the use of modems, it is an attractive alternative in that it provides rapid access from any telephone obviating the need for additional hardware.

Perhaps even more significant will be the resulting changes in existing applications for dynamic support of stored voice as a data type. When users can access databases over the telephone some will wish to update them concurrently. If a caller wishes to enter data such as an entry in a calendar or the reply to an email message, recording a voice snippet is much easier than trying to type text with the telephone keypad. However, since we cannot yet reliably translate voice into text, the new entry must remain a voice file, and the underlying database becomes multimedia. The combination of voice and text data, in turn, has repercussions on the graphical interfaces of desktop applications accessing that database. And screen-based voice mail with audio cut and paste allows all the user's telephone messages to become data for manipulation by other applications as well.

Voice interfaces allow the telephone to be used as a computer terminal, and with the advent of highly portable cellular telephones, users can access their desktop from almost anywhere. But other portable technologies will utilize voice, and, in turn, contribute to the role of voice as a data type on the desktop. As laptop computers shrink to palmtops and smaller, the display and keyboard become limiting factors to further size reduction. Voice interfaces require little space; hand-held computing appliances may soon be used to record voice notes for later inclusion in other desktop applications [Stifelman *et al.* 1993]. For example, one may use a hand-held digital recorder to gather thoughts for a presentation while taking a walk or driving to work and later organize the recorded ideas at the office. Once uploaded into a desktop computer, graphical tools could allow the user to further edit, organize, and annotate the spontaneously spoken ideas into a coherent outline or a multimedia document.

GRAPHICAL USER INTERFACES

Although the previous section pointed out that nonvisual user interfaces could enable computer use in novel situations, most applications of voice as data also benefit from visual interfaces when used on the desktop. Graphical representations of audio cue the user to the presence of voice data in an application as well as allow control of playback. Richer graphical interfaces allow direct manipulation of the stored voice by providing a means of positioning playback at random points throughout the sound as well as offering visual cues to a sound's overall length.

Simple "sound button" graphical interfaces (see Figure 12.1) use an icon such as a sketch of a speaker to alert the user to the presence of a sound. The user can click on the icon to play the sound and then click again to stop. Buttons are advantageous because they require little screen space, allowing many of them to be used by an application or to be mixed with text (see Figure 12.2). Buttons are



Figure 12.1. An assortment of “button” representations to indicate the presence of sound in a visual user interface. These representations are from (left to right) NeXT, Digital’s Xmedia, and Sun’s OpenWindows.

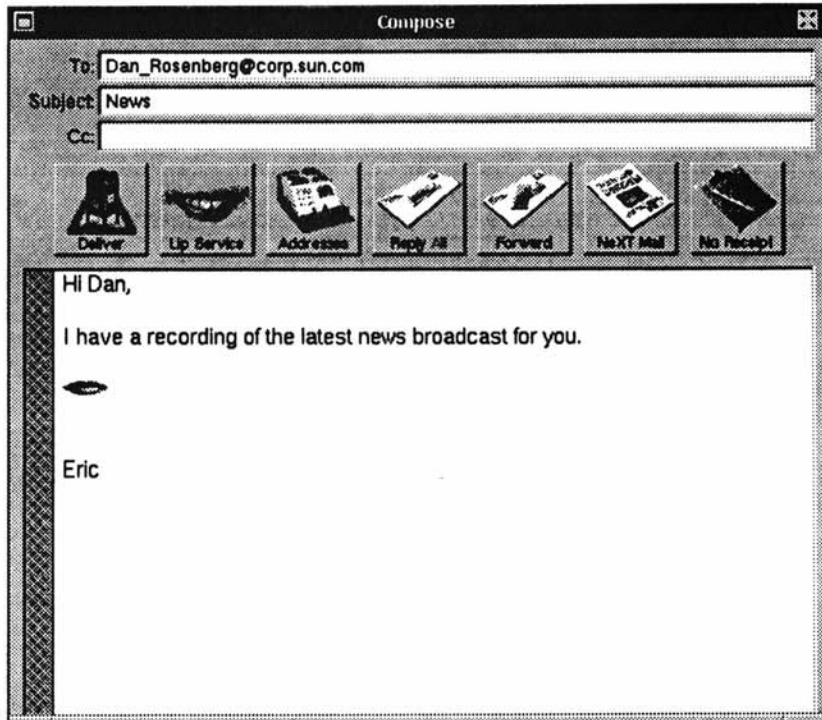


Figure 12.2. Because they are small, buttons can be easily mixed with text. This figure shows voice embedded in a text email message, on a NeXT computer.

intuitive to operate for most users familiar with graphical interfaces. The chief disadvantages of buttons are that the user has no sense of how long a sound will last once it starts and must listen to the entire sound sequentially. The lack of any sense of the duration of a sound while it plays makes it difficult to quickly search a number of sounds for desired information and shows many difficulties associated with similar graphical interfaces [Myers 1985]. The inability to skip around within an audio segment renders buttons useful only when they are used to control very short sounds.

An alternate form of graphical user interface displays additional information about the sound file and allows greater playback control; such visual representa-

tions of sound map time to the horizontal dimension, showing a bar the length of which indicates sound duration. The user plays the sound by clicking on the bar, and as playback progresses a cursor moves in synchronization to indicate temporal position. With many of these interfaces the user can change the playback position by clicking to reposition the cursor. Chapter 4 illustrated a variety of time bar representations, many showing speech and silence intervals. The Sound-Viewer is a more recent example of such an interface; it is described below as a case study.

The time bar representation has several advantages over the simple sound button by indicating the duration of the sound and providing finer control over playback. Its main disadvantage is the amount of screen space it requires. This difficulty has led to implementations in which the presence of a sound within an application is indicated by a small icon; when this icon is clicked on or dragged to another location a larger control panel opens up for playback, sound editing, and other functions (see Figures 12.3 and 12.4). Such an interface affords many of the advantages of the time bar display while saving screen space (especially useful if multiple sounds are visible), although it does require additional effort on the part of the user, and more importantly requires the user to shift gaze to the location where the control panel appears.

AUDIO SERVER ARCHITECTURES

An appropriate software architecture is required to support access to desktop audio by multiple applications. Current audio servers support digitization and playback of voice, but future servers will also incorporate software implementations of the speech processing technologies discussed throughout this book. Software-based speech recognition, speech synthesis, audio data compression, time-scaling, and other signal manipulation algorithms are feasible on today's workstations; products to perform these operations are already becoming available. When recognition and synthesis required additional external hardware devices or add-in cards, they were costly and unwieldy. When available as software, these technologies can be considerably less expensive and more widely available and hence attractive to application developers.

Audio servers allow voice resources (speaker, microphone, voice processing algorithms) to be made available to multiple applications running simultaneously on one workstation. A server-based approach allows distributed audio processing and resource management among a variety of client applications. The relationship between an audio server and its clients is similar to that found in a server-based window system; instead of directly manipulating audio devices, clients make requests to the server, which controls devices on behalf of the clients. Because clients do not manipulate hardware directly, each client can operate without knowledge of the other clients. The server, or a separate policy agent operating in concert with the server, arbitrates conflicting requests (for a window system, the policy agent is the window manager). For example, if one client requests the server to play a sound while another client is already in the

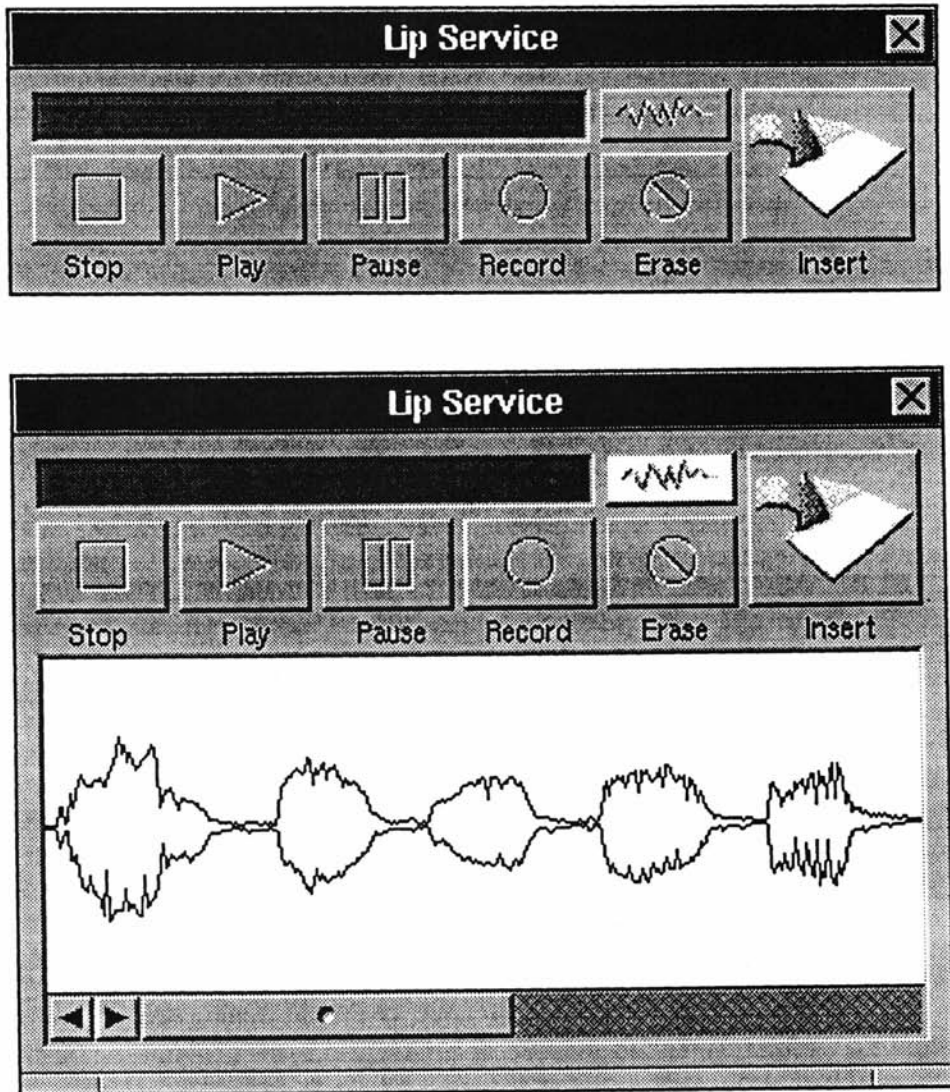


Figure 12.3. The NeXT audio control panel. In the top form sound is represented only as a bar; by clicking a button the user can invoke the waveform envelope display shown at the bottom.

midst of playing a different sound, various policies might be: make the new client wait, make the first client stop playing, or mix the sounds and play them simultaneously.

Because the server approach divides client and server into separate processes that communicate via a well-known protocol, several additional benefits are realized. An application need not be recompiled to execute with a different version of

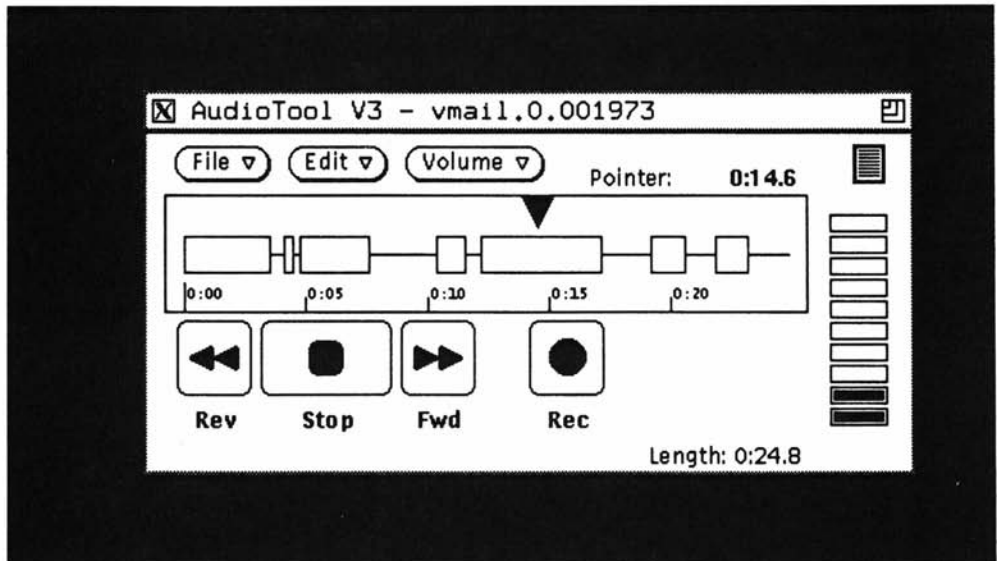


Figure 12.4. Sun's audiotool control panel displays speech and silence segments and provides editing capabilities.

the server; the same application binary file can run on different workstations with different audio hardware, simplifying software distribution. Additionally, because the server implies an interprocess communication path between client and server, a client can be made to communicate transparently with servers on its local or remote workstations across the network.

Another advantage of an audio server is that it frees the client application from the time-critical task of reading and writing data between the audio devices and disk files. The server also shelters the application from the detailed programming sequences that control the audio devices; the audio protocol is device independent. These characteristics lead to more modular and portable application software.

The client-server approach is evident in many of the projects described elsewhere in this book, such as Etherphone and MICE (see Chapter 11). An early audio server was developed at the M.I.T. Media Lab [Schmandt and McKenna 1988] in 1984; as with the two examples just mentioned, this server executed on separate hardware from the client. The availability of inexpensive add-in hardware and the real-time nature of the server's DOS operating system allowed the audio server to be implemented on a personal computer that communicated with a Unix host over a serial interface. This early server was motivated largely by the difficulties of managing the real-time audio data stream on early slow Unix workstations. Once playback of a sound begins, it must continue to completion without interruption; stops and starts in the audio medium are much more jarring to the user than an interrupted window refresh on a display.

All audio data was stored on a dedicated disk within the server hardware. The server incorporated a request **queue**; a client could ask the server to prepare for a series of play and record operations (e.g., the prompts and message-taking for an answering machine) and then let the server execute them without further intervention. By queuing requests, an application allowed the server to perform some time-consuming operations in advance such as creating and opening files and ensured that back-to-back play requests could be performed with no silence in between. The client received asynchronous events notifying it of completion of each task.

This server was later extended to include speech recognition, speech synthesis, and several audio compression algorithms, but never fully supported multiple clients' access to the server simultaneously. To do so would have required the server to associate requests with clients and maintain the correct state of devices for each client. For example, one client may set the output gain high, while another sets it low; the server would need to reset the gain of the audio device to the appropriate client-dependent level.

The Olivetti VOX audio server [Arons *et al.* 1989] provided more sophisticated protocols and implemented resource management for multiple simultaneous clients. It digitized and played sounds back from files managed by the server, but VOX also included external computer-controlled analog mixing and switching equipment to route audio to several speakers located in the same or different offices. To support multiple clients, VOX allowed clients to build hierarchical *CLAUDs* (Composite Logical Audio Devices) as server-side constructs. A *CLAUD* defined all the audio resources needed by a client, much as a window hierarchy specifies all of the graphical objects belonging to a client of a window system server. Again, as with window systems, clients could request that a *CLAUD* be **mapped** (activated) or **unmapped** (deactivated). Mapping would fail if resources were unavailable because they were currently utilized (mapped) by another client. A *CLAUD* contained a *Logical Audio Device* (*LAUD*) for each physical device it included; a *LAUD* stored the desired state of the physical devices for a particular client. When a *CLAUD* became active, the physical devices were set to the state appropriate for the client as contained in its *LAUDs*.

The Digital Equipment Corporation's Xmedia audio server [Angebrannt *et al.* 1991] incorporates the server-side client resource constructs of VOX (the composite audio device hierarchy), but the mixing and routing of audio paths are completely digital. The protocol for this server is more heavily influenced by that of the X window system, including a more sophisticated protocol than VOX for allowing a "media manager" to implement resource management policy. The Xmedia server also provides audio data paths to the client; in addition to recording or playing from a file, the client can provide or access audio data directly. Its internal logic for mixing and synchronizing multiple simultaneous data streams extends to control digital video as well as audio.

The Xmedia server was originally designed in a framework including an audio toolkit to free clients from the need to directly interface to the low-level audio server protocol, but only a minimal toolkit has been developed to date. A recently

published audio server architecture from Hewlett-Packard shows a design very similar to that of the Xmedia server [Billman *et al.* 1992].

Because of the limited availability of any of the audio servers described so far, a new server has recently been developed at the M.I.T. Media Lab; it is used by many of the applications described later in this chapter [Arons 1992b]. This server does not contain any of the server-side resource management constructs from VOX or the Xmedia server but does implement a simple priority-based resource management policy. Clients can send audio data to the server or have it play from files and receive asynchronous notification of audio events. This server includes software for time-scale modification (as well as touch tone detection) making it simple for any application to vary playback speed. Other servers, e.g., for speech recognition and synthesis, are designed to work in concert with the audio server by using its ability to route copies of audio data streams. For example, a recognition server will always get copies of audio input while recognition is enabled. If some other client requests the server to record a sound, the same audio data will be copied into a file. Finally a recording-level client (e.g., to display a graphical “VU meter”) would request its own copy of audio data whenever another client activates audio recording.²

The Media Lab’s audio server is motivated primarily by the need for effective resource management because many applications share the audio device as well as audio data itself. In the absence of a policy agent, or audio manager, this server implements a simple priority scheme. Clients may declare themselves to be of high (urgent), medium (normal), or low (background) priority; higher priority requests supersede lower priority ones, whereas lower or equal priority requests are queued until the server is idle. When a play operation is superseded by another of higher priority, playing pauses and the client is notified; when the new request finishes, the interrupted one resumes. An interrupted record operation does not resume without client intervention.

One component of a software architecture missing from the servers just discussed is an **audio toolkit**. Graphical user interfaces are usually implemented in conjunction with software toolkits consisting of a set of direct manipulation objects such as scroll bars, sliders, and dialog boxes.³ Audio toolkit objects might include constructs such as a **menu**, which speaks the user’s choices and awaits a selection via touch tone, or a **form**, which speaks, prompts and records replies. Resnick’s Hyperspeech project implemented a comprehensive set of telephone interaction objects to facilitate rapid development of telephone-based community bulletin boards [Resnick 1992b, Resnick 1992a, Malone *et al.* 1987]. Another such toolkit is described in [Schmandt 1993]; it is heavily used by Phoneshell, described below.

²Actually the meter client would request audio level events whereby the server provides periodic updates as to the audio energy level and thereby minimizes the data being transmitted from server to client.

³These are the *widgets* of the X-windows community.

Arons describes another hypothetical set of “audgets”—audio widgets—in [Arons 1991a]. An audio toolkit should be distinct from a graphical toolkit as some audio applications have no graphical interfaces. Graphical audio controllers, such as the SoundViewer described later, are best implemented as a separate toolkit layer, as shown in Figure 12.5, so as to free nonvisual or nonaudio applications from the added baggage of the unused media.

UBIQUITOUS AUDIO

Most applications of desktop audio involve explicit invocation of recording activity; the user clicks the mouse or presses a touch tone key to digitize a short snippet of audio. Such recordings comprise but a small fraction of the speech we produce during a work day. We hold meetings, converse on the telephone, chat with visitors, and catch up on news at the water cooler. All of this talk has so far been beyond the reach of computer applications, but desktop audio configurations could allow recording and subsequent retrieval of much of this previously evanescent “ubiquitous audio.”

Assuming 10:1 data compression of telephone-quality speech with half of the day spent in conversation of which 80% is speech and the remainder silence, a year’s worth of office conversation would require slightly over 2 Gigabytes of storage. Although this quantity would have seemed outrageous as recently as several years ago, such storage capacities are rapidly becoming feasible. The combination

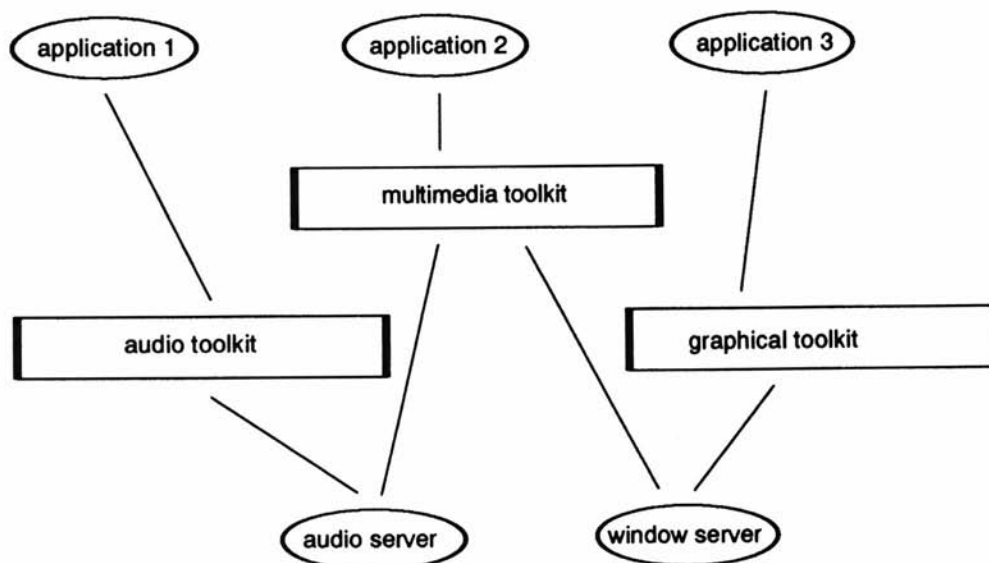


Figure 12.5. Audio and graphical toolkits shield applications from details of communication with their respective servers. Multimedia toolkits can be built on top of single-medium toolkits.

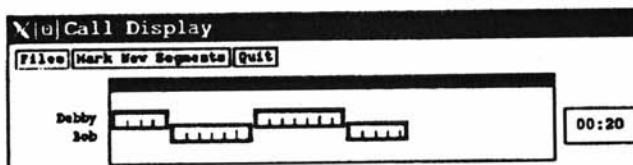
of audio-capable workstations, plentiful storage, and computer-mediated telephony hardware could allow us to record all the conversations that occur in our offices and meeting rooms. There is certainly no doubt that this audio record would contain a wealth of useful information and a valuable archive of business or engineering design decisions, but it would also include a plethora of social or otherwise irrelevant conversations of little archival value.

How could we access the important portions from such a vast quantity of stored voice, which is such a difficult medium to summarize or search? A variety of information sources come into play. Situational information specifies the circumstances in which a recording was made, but nothing about its contents; this includes dates, location, and participants in the conversation. It has been suggested that situational information may be adequate for retrieval of otherwise poorly structured multimedia records such as audio or video recordings [Lamming and Newman 1991]. If users are identified by wearing locator devices, by face recognition from a video record, or by initiating or receiving a call placed by a computer or the telephone network, the audio archive could store a list of talkers with each recording session. Applying situational constraints to form queries such as "Find all long telephone conversations with Eric last spring" can narrow the search immensely.

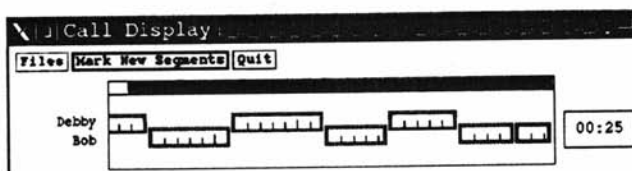
Because of the slow and serial nature of speech, playback of monolithic audio files is tedious regardless of their length. If structure can be added to the recording, it could provide a means of breaking the recording into segments, which indicate reference locations identifying conversational landmarks to assist random access techniques. Structure to aid audio retrieval can be created explicitly by user activity during recording or derived after the fact based on acoustic analysis.

For example, participants in a meeting may take notes on laptop computers; these text entries can be time-stamped and synchronized to the audio recording. Later a graphical conversation browser could tag the visual representation of the recording with the text from the notes or alternatively a text viewer could allow the user to click on a sentence and hear the associated audio. Acoustic analysis can break a conversation into segments delineated by pauses, and this can be incorporated into the visual browser as well. A more powerful method is to identify which speaker is talking at any moment, and visually differentiate the display of each speaker.

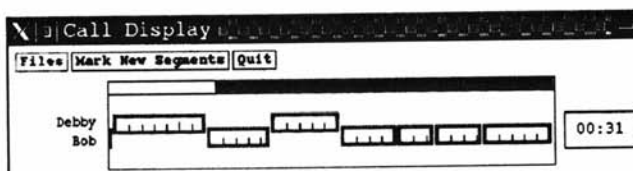
In some situations more explicit user activity could trigger recording, which would yield smaller units to archive, but this action must be unobtrusive. A Media Lab project shown in Figure 12.6 displays a graphical interface on the workstation while a user is engaged in a telephone call [Hindus 1992]. Speaker turns are detected and each scrolls across a small viewing panel showing the recent history of the conversation; each box represents the audio of a single turn. The user's hands and eyes are free during a phone call, and the mouse can be used to start or stop recording at any point. Several portions of the call might be recorded, and each is grouped into a "paragraph," or series of consecutive turn segments. Text annotation can also be added to the speech display. A similar graphical interface can be invoked at a later time for browsing completed telephone calls.



- D: Hello, this is Debby Hindus speaking.
 B: Hi Deb, it's Bob. I'm just getting out of work, I figured I'd call and see how late you're going to stay tonight.
 D: Well, I think it'll take me about another hour, hour and a half, to finish up the things I'm doing now.
 B: OK, I'm just going to head on home, I'll probably do a little shopping on the way.



- D: Well, if you think of it, maybe you could get some of that good ice cream that you got last week.
 B: OK. By the way, somebody, uh...
 B: mentioned an article you might be able to use



- B: in your tutorial. Debby: Oh really? [Debby's very short turn is ignored.]
 B: Yeah, it's by Graeme Hirst, in the June '91 Computational Linguistics.

Figure 12.6. A telephone recording tool shows the recent history of a conversation. Each box represents one turn; position differentiates speakers. Reprinted with permission from proceedings of the *ACM 1992 Conference on Computer-Supported Cooperative Work*, © 1992, ACM.

Another Media Lab application, Xcapture, records ambient sound in an office into a circular buffer or “digital tape loop,” which serves as a short-term auditory memory aid. The archetypical situation for its use is a collaborative writing session; one person suggests an alternate wording for a sentence while the other hurries to write it down, but neither can repeat the revision word-for-word and it is lost. Xcapture records in the background, displaying a small animated icon with a moving bar. When the user clicks on this icon, recording stops and a SoundViewer widget appears in a popup window (see Figure 12.7); the SoundViewer allows interactive playback of approximately the last five minutes of recording (the precise amount depends on available system memory). The user can review this recording, scan it at a faster speed, save it to a file, or cut and paste it into another application.

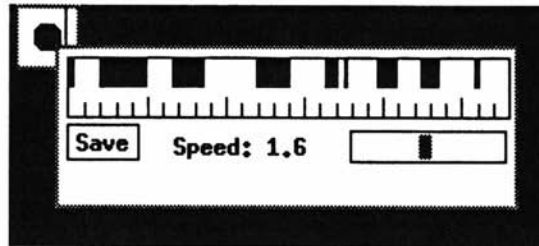


Figure 12.7. Xcapture displays a recording of recent conversation in a pop-up window.

Both these projects demonstrated potential but are limited in capability [Hindus and Schmandt 1992]. Xcapture is barely adequate for even short-term retrieval of a recent conversation because the SoundViewer lacks any cues of turns, pauses, or other structural landmarks in a conversation; even five minutes is a long duration of sound to navigate. Tools which are more sophisticated need to take advantage of multiple sources of structure and acoustic processing to enhance interactive retrieval of archived conversations. A requirement for the user is that it must take much less time to find the desired information than it would to listen to the entire recording sequentially. Playback speed is variable, using the time scale modification techniques described in Chapter 3. A recording can be scanned by playing small segments in sequence, skipping over larger intervening segments. Pauses in the conversation [O'Shaughnessy 1992] or emphasis detected using intonational cues [Chen and Withgott 1992] may suggest segments that are more likely to contain semantically significant utterances or mark the introduction of fresh topics. When coupled with an interactive user interface, such as one based on a touch pad, to scan through a recording at several levels of granularity [Arons 1993], none of these techniques need to work perfectly. Instead, they can act as an aid to the intelligent user, who may have even participated in the conversation being searched and have some memory of its overall structure.

Although recording the ubiquitous audio at work may be straightforward, retrieval from audio archives is difficult. Special capture applications unique to particular recording situations may be useful in the short term. In the long term, more research is needed into techniques to supply structure to recorded sound based on acoustic analysis. Additionally, development of interaction techniques, based in part on the acoustically derived structure, will facilitate retrieval from the audio archive. Whether such factors will impart real value to large audio archives remains to be demonstrated.

CASE STUDIES

This section presents four case studies emphasizing various aspects of desktop audio. The first case study describes the iterative design of a visual representa-

tion of stored voice. The second highlights Conversational Desktop, an eight-year-old project designed to provide functionality spanning many desktop audio application areas. The third case study is of a telephone interface to many common desktop utilities, which is in use today. The final study returns to visual interfaces by examining more recent work in screen interfaces to audio applications such as voice mail; these applications complement the telephone interface in the third case study and are also in use today.

Evolution of a Visual Interface

Visual interfaces to stored audio have been a recurring theme in this book. Chapter 4 pointed out the role of a visual user interface for an audio editor. Earlier in this chapter we returned to the utility and possible functionality of such an interface, presenting two extremes of graphical control of audio playback. A button-style interface requires very little screen space but offers minimal functionality; its small size allows it to be used repeatedly on the computer display by which-ever applications contain audio data. These buttons were contrasted with more elaborate interfaces which offer visual representations of the stored voice, provide random access through voice files, and may even support limited editing functionality (but at the cost of size and complexity). In the control-panel style interface, sounds are represented iconically and the control panel appears when a user activates an icon.

This section considers a case study of the Media Lab's SoundViewer, a graphical interface which attempts to provide on-the-spot interactivity combined with small size so that the interface can be used in place by many applications [Hindus *et al.* 1993]. The SoundViewer uses horizontal size and time marks to convey the duration of a sound before a user decides to play it. It provides for direct manipulation at the place on the display where the user's attention is already focused, instead of in a separate control panel. Finally, a SoundViewer can operate in conjunction with other SoundViewers in the same or other applications to facilitate sharing voice data. Because the SoundViewer is an X widget, it can be controlled via X window resources and any change to the widget is inherited by all applications using it (after compilation). An application is required only to set the location of the SoundViewer and specify the sound file it is to control; all other graphical and audio interactions are managed by the widget.

The initial design of the SoundViewer was based heavily on earlier M.I.T. visual interfaces, including the Intelligent Ear (Chapter 4) and Phone Slave's graphical interface (Chapter 11). As with these earlier projects, the visual object's size indicates the length of the sound and its representation changes appearance in synchrony with audio playback. The basic SoundViewer action is shown in Figure 12.8; as the sound plays, a bar of contrasting color moves along left to right. The left mouse button is used to start and stop playback, while the middle button moves the position of the bar. Moving the bar during playback skips to a new location in the voice recording. Repeated clicking on the middle button repeats a segment over and over, which is particularly useful for tasks such as transcribing a phone number from a voice mail message. The right mouse but-

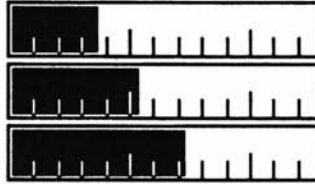


Figure 12.8. As a SoundViewer plays its associated audio file, a bar moves left to right in synchrony.

ton can be used to specify a portion of the SoundViewer, which then becomes the audio *selection* (see Figure 12.9); the selection can be moved to other applications using the standard X protocols.

One issue with visual display of audio data is the mapping between time and pixels, or width, which can provide both relative and absolute indications of sound length. The SoundViewer uses an internal algorithm (or one supplied by the application) to determine the interval at which to display tick marks; indicating time with too much temporal detail would just blur the marks. Although the heights of the tick marks are meant to differentiate them (e.g., those at every second are shorter than those at five-second intervals), no users have ever understood this without instruction. SoundViewer width is a very good indicator of relative length, however. But scale becomes a problem when short sounds are mixed with much longer sounds; the time-to-pixels ratio which just allows a 7 minute sound to fit on a screen would likely result in a 5 second sound being so short as to be rendered almost invisible. The SoundViewer therefore was made to support a mode in which sounds less than a specified threshold length are displayed at a constant scale to allow relative length comparisons, while longer sounds are “crunched” into allocated space by changing the time scale. Although this also changes the spacing of the tick marks appropriately, and the time bar moves more slowly as a result of greater time-to-width compressions, this duration cue is also poorly understood by users.

The tick marks provide useful navigational cues. For example, while listening to voice mail, users often watch to note the location of the time bar when the caller speaks a phone number; this makes it easy to return to the number at the conclusion of the message. But the ticks convey nothing of the *content* of the sound so an alternate representation was developed to display speech and silence intervals (see Figure 12.10) similar to Etherphone’s visuals. More recently, “dog



Figure 12.9. Horizontal streaks provide visual feedback when a portion of a SoundViewer’s associated sound is selected with the mouse. This SoundViewer shows less temporal detail than those in Figure 12.8.



Figure 12.10. The addition of speech and silence marking and “dog ears” to a SoundViewer.

ear” markings were added to the SoundViewer. A spot can be marked (using the keyboard) during playback, and the location of the mark, which appears as a caret symbol, is saved with the sound and appears whenever it is displayed.

After speech and silence intervals had been added, the SoundViewer was made to allow the user to jump ahead to the next speech segment or back to repeat the current or previous segments also under keyboard control. Pauses in the speech correspond to breath groups and are usually deliberate, semantically meaningful speech segments. The SoundViewer also has for some time supported audio time-scaling to play back sounds faster or slower than they were recorded (time-scaling algorithms were discussed in Chapter 3). Because this is a widget “resource,” users can set default time scales for all SoundViewers or for those of just a particular application; when users become accustomed to time-scaled speech they begin to prefer it over normal rate speech. A more recent addition provides improved random access; while moving the location bar by hand back and forth across the width of the SoundViewer, it plays small chunks of sound in synchrony. This feedback provides surprisingly useful navigational cues although the proper presentation style is the subject of further research.

The SoundViewer breaks down for very long sounds. When time is mapped to width so as to allow a 10 or 15 minute sound to fit into a reasonable portion of the screen, time is so compressed that the sound bar barely moves as it plays, and it can be positioned only very coarsely. Also, the speech and silence markings interfere with each other when most pauses are shorter than the audio duration represented by a single pixel. Although these effects are not noticed on short sound snippets characteristic of voice mail or calendar entries, they have become problematic for longer sounds such as recordings of meetings or newscasts. A new variant of the SoundViewer provides a global and a detailed view of the sound as shown in Figure 12.11. The lower representation presents a close-up view and is a window into the entire portion of the sound, which appears above. The user can start playback, set a position, or make a selection from either view; as the sound plays, both views get updated simultaneously.

The SoundViewer has been used in all the recent Media Lab desktop speech applications, some of which are described later in this chapter. It has been effective at providing a direct manipulation interface to stored voice files while consuming minimal screen space. Its visual representations have evolved to better facilitate navigation, and additional audio processing enhances the SoundViewer with time-scaling and auditory feedback during random access. Maintaining consistent visual time scales has been a problem from the beginning, and an auxiliary view seems essential for managing very long sounds.

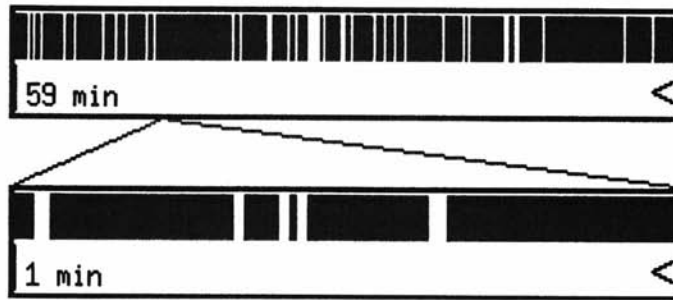


Figure 12.11. For very long sounds two representations are provided. The lower view is a close up of the entire sound shown above.

Conversational Desktop

Conversational Desktop was an early (1985) effort at the Media Lab to present a vision of the possible range of desktop audio applications [Schmandt and Arons 1986]. It emphasized integrated telecommunication management combined with extensive use of voice both as a data type and as an element in the user interface. In addition to spoken language, Conversational Desktop included a touch-sensitive graphical interface which, in anticipation of window systems, allowed the user to quickly switch between displays of the calendar or voice messages. Conversational Desktop's dialogue system and parsing techniques that cope with fragmentary and error-prone speech recognition were described in Chapter 9; this case study focuses on its functionality and integration of multiple application areas.

Conversational Desktop operated across multiple workstations. Each workstation included speech recognition, speech synthesis, audio digitization hardware (implemented on separate dedicated computers acting as servers), and a telephone interface. This project spanned telephone management, voice mail, scheduling, remote database access, and audio reminders triggered by external events. From the standpoint of this chapter, Conversational Desktop's most interesting aspect was the synergy arising from the interaction among these functions.

Conversational Desktop emphasized the role of the workstation in managing both remote data access as well as voice communication through local and wide area networks. The workstation acted as a communication agent, contacting external databases (a traffic information service and simulated access to airline reservation systems) as well as negotiating with agents of other users (scheduling meetings and call setup negotiation). Although services were implemented over analog telephone lines and Ethernet, this project was designed in anticipation of ISDN; calling another Conversational Desktop user established a data connection as well as a voice circuit so the users' agents could communicate while users were conversing.

Conversational Desktop placed telephone calls through voice dialing and incorporated the conversational answering machine approach first implemented in Phone Slave (see Chapter 11). Calls between Conversational Desktop work-

stations were set up using the local computer network before an analog telephone circuit was established; as the calling party was therefore known, the answering machine could play personalized messages and inform callers of the status of earlier messages just as Phone Slave had.⁵ Because the Conversational Desktop maintained the user's schedule, outgoing messages could be automatically selected to account for current activity (in a meeting, out to lunch, out of town, etc.). In addition, microphones in the office monitored audio levels to determine when visitors were present, and could automatically take messages from unknown callers without interrupting a conversation in progress.

Because it utilized early connected speech recognition, Conversational Desktop users were required to wear a head-mounted noise-canceling microphone. This microphone was also used during telephone conversations; its noise cancellation allowed use of full-duplex speakerphones. Additional microphones mounted by the walls were used to detect when other people were speaking; speech present at the background microphones but not at the noise-canceling microphone indicated that a visitor was speaking. When the user spoke, a comparison of audio levels in the background microphones could determine the direction he was facing; direction of speaking was used to model user attention and enable recognition. While the user was facing away from the monitor, speech was ignored, but when the user faced the workstation, audio from the microphone was switched to the recognizer. Turning towards the workstation also muted audio to the telephone, so the user could have a private conversation with the computer during a phone call.⁶

Conversational Desktop also allowed its users to record short audio reminders for later playback. When issuing the command to record a reminder, the user specified the situations or activities that would prompt playback such as "when I come in tomorrow." Playback of a single reminder could be triggered by multiple events: a "when I talk to Barry" reminder would be played when Barry called, when the user placed a call to Barry, or when the application announced that it was time for a scheduled meeting with Barry. Reminders were not played, however, when another person was detected in the office as the reminder might contain private information.

This project was an early example of applications of desktop audio. It used speech recognition and a dialogue system to address multiple application functions simultaneously and allowed each function access to stored voice. It explored issues in dynamic routing of audio paths among applications and combinations of speakers, microphones, and voice processing hardware. By incorporating a range of functions, Conversational Desktop demonstrated the synergistic interactions among disparate computer applications dealing with what might appear to be a single operation by the user, e.g., being reminded of something when placing a

⁵Where available, telephone network provided calling party identification and it could serve the same function today.

⁶An unfortunate disadvantage of this arrangement was that the user could not simultaneously read the computer screen and talk on the telephone.

phone call or automatically updating the outgoing voice mail message depending on one's schedule. Finally, Conversational Desktop demonstrated the personalization of call setup, depending on factors such as the calling party and activity in one's office.

Conversational Desktop also had a number of limitations. It was written as a monolithic application, which eliminated the need for interprocess communication but interfered with its modularity and hampered its extensibility. It also side-stepped some important considerations in managing the focus of speech recognition input across multiple applications; if the user is running several programs at once, which is being addressed at any moment? This project utilized custom-built or expensive peripherals, and a good deal of the computing resources of a laboratory group, which relegated it to the role of a demonstration system instead of one to be used in daily work. The audio switching based on direction of speech worked well in a single, sound-treated room, but may not extend well to noisy offices or open cubicles.

Despite these limitations, Conversational Desktop was valuable as an early, visionary system that later motivated much of the work described in several other projects described as case studies in this book; a remaining case study is described in this chapter. Although desktop audio required many exotic peripherals in 1985, all the functionality of Conversational Desktop can be implemented without additional hardware on workstations recently introduced by several vendors.

Phoneshell

Phoneshell [Schmandt 1993] is a family of applications that allows telephone and facsimile access to common desktop utilities including voice mail, electronic text mail, name and address database, and calendar. Phoneshell illustrates the utility of the telephone as a remote link to the desktop, and the enhanced role voice can play as a data type motivated in large part by the difficulty of entering text with a telephone. The telephone interface stimulates the use of voice as a data type and places new requirements on the screen-based applications that must provide access to the stored voice at the desk.

Phoneshell consists of a set of applications loosely linked together under a top-level menu, which allows the user to invoke applications sequentially during a session. The applications share many aspects of their user interfaces, for consistency, and depend on common lower-level utilities, but each application has been developed independently. Figure 12.12 summarizes the functions embedded in Phoneshell.

The voice mail application is similar to many commercial voice mail products with several additional features. Voice messages can originate as telephone messages, from other voice mail users sending a voice message from their workstations, or via digitized voice attachments encapsulated in email arriving over the computer network. In addition to replying to a voice message or recording a message for another user, a caller can record "memo" messages. A memo is intended for screen access only; instead of cluttering the voice mailbox during telephone



Figure 12.12. Phoneshell's menus provide access to a variety of personal information management functions.

access, a separate graphical things-to-do application absorbs the memo messages into its own database. Finally, in addition to being able to transfer a voice message to another user, a Phoneshell user can save an entire message to her calendar by specifying a date; this is the nongraphical equivalent of audio cut and paste.

The email-reading application is similar in many ways to Voiced Mail described in Chapter 6 in that it uses touch tone input to control the reading of email messages using speech synthesis. As with Voiced Mail, Phoneshell carefully pre-

Date: Tue, 30 Mar 93 21:01:55 PST
 From: Ben.Stoltz@Eng.Sun.COM (Ben Stoltz)
 Message-Id: <9303310501.AA13979@denwa.Eng.Sun.COM>
 To: geek@media.mit.edu
 Subject: Re: your schedule
 Status: RO

> From geek@media.mit.edu Tue Mar 30 15:17:50 1993
 > Delivery-Date: Tue, 30 Mar 1993 15:17:53 -0800
 > Date: Tue, 30 Mar 93 18:17:44 -0500
 > From: Chris Schmandt <geek@media.mit.edu>
 > To: stoltz@denwa.Eng.Sun.COM, stoltz@Eng
 > Subject: your schedule
 >
 > What's the possibility of getting together Tuesday?
 >

Dinner would be fun. Tuesday evening.

Figure 12.13. An email message that contains parts of other messages.

processes text such as email addresses so that they are pronounced correctly and breaks a message into sentences so the caller can skip forward and backward in the message and repeat the sentence currently being spoken. In addition, while reading a message included messages and their mail headers (see Figure 12.13) are detected; headers are summarized (“Header of message from ‘Chris Schmandt’ in reply to ‘your schedule’”) and included text can be skipped over by a “jump ahead” button. A message can be faxed as well as read; this is especially useful if reviewing mail about subjects such as an agenda or schedule from a remote telephone.

The user can also send replies using either voice or text. Voice replies are recorded and then returned to the original sender as a voice attachment to an email message using formats from a number of vendors. With voice being supported on a growing number of workstations, such attachments have become a convenient means of reaching many correspondents directly; this had been impossible a decade earlier under Voiced Mail. If the user does not know what message format to use or has no reason to believe the sender uses an audio-capable workstation, text replies can be typed with touch tones. Two keypresses specify each

character; the first indicates the group of three letters and the second selects one of the three depending on whether it is from the left, middle, or right column of keys. Punctuation and digits are also available. Whenever the user completes a word ("*" is the space key), it is echoed. When the user pauses, the current word is spelled out. The most recent word can always be deleted. The message includes an automatically generated addendum explaining that it was entered using touchtones and the author's "signature" information.

Mail reading is facilitated by the addition of **filtering**. Messages can be sorted into categories such as "urgent," "important," "personal," and "other" based on keywords found in the subject line or on the basis of the author of the message. Phoneshell users can specify arbitrary filtering categories and determine their presentation order. A user who receives large quantities of mail is likely to "read" only the more important categories using speech synthesis simply because this method is so much more time consuming than reading text on a terminal.

Some aspects of Voiced Mail were not carried over into Phoneshell. Because users now get many more email messages, they are less likely to want to hear all their messages so Phoneshell speaks the sender and subject of each message but does not recite the message itself unless requested. The repetition strategy of Voiced Mail (slow down, spell mode) was also abandoned.

The calendar application, Caltalk, lets users scan their calendars and add new entries. A date can be specified with touchtones, and users can hear calendar entries for that day item-by-item; text entries are synthesized and voice entries are played. New entries are recorded and stored as voice annotations. Portions of the calendar can also be faxed if requested.

Although reciting each entry for a day is effective in describing a particular day, in many ways using an auditory calendar interface is more difficult than a graphical interface. Earlier versions of the application provided a "week-at-a-glance" function that merely recited each entry day by day; this ineffectively conveys the overview available from scanning a graphical representation: "The first part of the week is rather free, Wednesday and Thursday have some appointments, and Friday is packed." A more recent version of Caltalk includes new week-at-a-glance and month-at-a-glance functions that attempt to better summarize the calendar and recognize keywords such as "important" in calendar entries as well as entries that span multiple days or are regularly scheduled each week. Caltalk might say, e.g., "Nothing scheduled Monday or Wednesday, important meeting with British Telecom and the usual meetings on Tuesday, you are in Palo Alto Thursday and Friday." Terse summarization is difficult to do well and illustrates some of the problems in converting a tool which usually is accessed visually to a voice-only interface.

A fourth application, Rolotalk, provides access to a personal name and address database. The caller spells a name using touch tones, one tone per letter, to select a "card" from the database; the user can also specify alternate search criteria, such as company name. Once selected, the user can request telephone numbers, postal addresses, electronic mail addresses, and additional information about the selected person. Most useful is Rolotalk's ability to communicate

with the selected person; it can place a phone call to either the home or work number or send a voice or text message. When placing a call, Rolotalk dials the destination party and creates a three-way conference call including itself, the user, and the called party. Rolotalk remains on the line for several minutes; if it hears a “#” tone, it drops the onward call and returns to normal user interaction. This allows the user to remain connected to Rolotalk after dialing a busy number or leaving a message on an answering machine. Message transmission, either voice or text, is accomplished by the same mechanisms employed to respond to electronic mail.

In addition to information stored in explicit fields, the underlying Rolotalk database allows users to specify **links** to other files. Typical links contain maps to the homes or offices of people in the database or driving directions as text. When a user calls in, Rolotalk can try to recite any text file links using speech synthesis or can fax all the links to a nearby fax machine. A user can also fax email messages; this is especially useful for long formatted messages such as a meeting agenda or conference schedule.

Phoneshell also provides several simple communication utilities. A dial-by-name directory enables a caller to spell out the name of a Media Lab staff member and transfer the call to that person's number. Phoneshell can also report the locations of Speech Group members using the activity information described in Chapter 11. The caller can find out who is currently in the lab or logged in from home, call on site users at the nearest telephone, or inquire when a specific user was most recently detected. A Phoneshell user can forward either stored fax messages or an item selected from a small archive of frequently faxed documents (e.g., a map to one's office) to another number.

Although it is currently considered to be in developmental stages as a research project, Phoneshell has been deployed for several years at two locations with as many as 20 users. Its design has benefited from iterative improvements to its user interface due to users' varying requirements and levels of expertise. It has demonstrated the effectiveness of speech as a means of remote access for the mobile user by turning any telephone into a terminal. Although some of the Phoneshell databases are more easily accessed by a laptop computer and a modem, it is often inconvenient to hook up such hardware at a pay phone, to a cellular phone, or in a host's office or home.

Key to the success of Phoneshell has been its integration of multiple communication applications. Reading email over the telephone without any ability to reply would be very frustrating; the recent surge of support for voice encapsulated in email messages facilitates on-the-spot voice replies. It is sometimes useful to send a copy of an email message to someone who appears in one's rolodex. The necessity of calling separate voice mail and text mail applications to retrieve incoming messages would be inconvenient. The ability to query one's calendar while hearing messages about upcoming meeting dates makes it possible to coordinate schedules more effectively. In short, Phoneshell presents a work environment wherein a user can easily switch between as many applications as are required to complete a particular task.

Visual User Interfaces to Desktop Audio

Earlier in this chapter visual user interfaces were extolled as the means of providing the user with random access to voice as data, thereby overcoming some of its slow and serial nature. The visual user interface also acts as a spatial representation, providing a navigational aid for applications presenting multiple snippets of audio. In the desktop audio context, some applications are intrinsically linked to stored voice, such as a graphical interface to voice mail messages. But with the increased availability of audio recording hardware on workstations plus remote telephone access through interfaces such as Phoneshell, many other applications may benefit from stored voice snippets. Both classes of applications require graphical user interfaces to stored speech.

This section describes a family of applications developed in recent years at the Media Lab which operate in concert with Phoneshell as well as some of the telephone management utilities described in Chapter 11. The intent of this case study is to illustrate this chapter's claims about visual interfaces and interapplication communication with actual examples. The SoundViewer widget described earlier in this chapter is used across all of these applications.

The first application to consider is a visual user interface to voice mail, seen in Figure 12.14. When not in use, this application is reduced to a small icon; when new voice mail arrives, the icon blinks, serving as a "message waiting" light. If the user clicks a mouse button on the icon, the message window opens, displaying messages as a column of SoundViewers with indication of the calling party's name or number if known. Vmail also accepts email messages containing voice attachments in several vendor formats; for these the sender's email address is displayed instead of a phone number. When the mouse is moved into the label identifying the caller, text in the upper-right corner of the window shows the date and time at which the message was recorded.

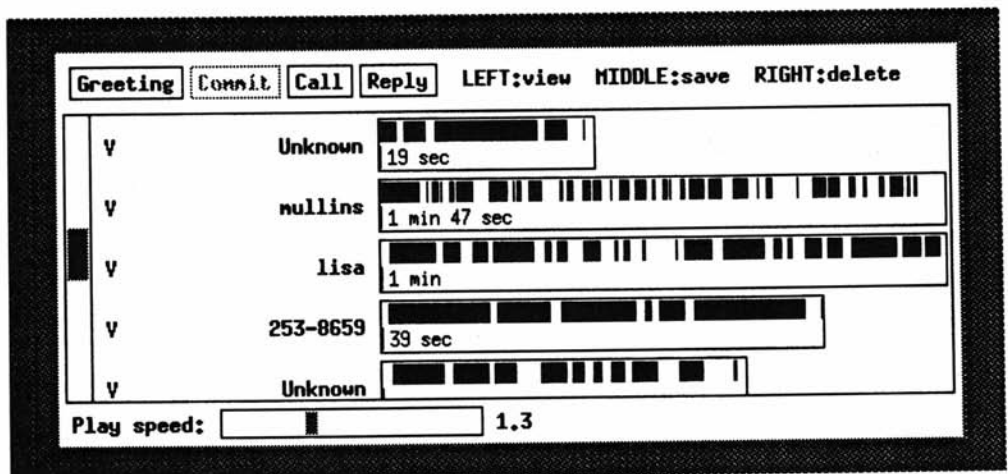


Figure 12.14. A visual interface to voice mail. Each row represents a message, and displays the caller's name or number, if known.

Clicking on a message causes it to play, with the SoundViewer bar moving left-to-right synchronously. While playing, the bar can be manipulated to move around within the message. If a second message is selected, the current message stops and the new message begins playback; this makes it easier to scan through a number of messages looking for one in particular, as messages typically can be identified by playing their first three or four seconds. Playback speed can be changed dynamically by a speed slider at the bottom of the window, and the user can specify a default playback speed for all messages.

After playback, a **V** appears to the left of the message indicating that it has been viewed; messages remain until they are explicitly deleted. Messages can also be saved to a file, in which case the user specifies a file name. If the caller's phone number is known, a return call can be placed by clicking on the "call" button; this sends a call request message to Xphone (described in Chapter 11). If the caller's name or email address is known, the "reply" button sends a request to Xmemotool (see below) to record a voice message in response.

This visual user interface to voice mail has proven very popular among the small community of users at the Media Lab. The direct manipulation SoundViewer interface makes it easier to play portions of the sound repeatedly, specifically while writing down a telephone number. The ability to increase playback speed lets users save time listening to messages, and some of them take advantage of this.⁷ Because the application allows multiple users to access the same voice mailbox, a group secretary manages four mailboxes with 20 to 30 messages a day. Although many users delete messages immediately after they have been read, some leave 10 or 20 messages in their mailbox because the visual interface makes it easy to navigate between them and quickly find a particular old message; this is cumbersome with a tone-based telephone interface.

Users can also move messages or portions of messages to other applications. Holding down a mouse button and dragging the cursor across a SoundViewer selects a portion of sound, which can then be pasted into one's calendar, for example, or into the Sedit editor (described in Chapter 4) for further annotation. After editing, the resulting sound file can be forwarded to another user by pasting it into Xmemotool.

Xmemotool (see Figure 12.15) is a window-based utility for recording voice messages. Messages can be composed for other local voice mail subscribers or sent to remote users as email messages, in which case one of several sound file formats must be selected. Xmemotool can also receive the current audio selection instead of taking a new recording; this allows a segment to be cut from a SoundViewer in any application and pasted in as a voice message to be sent.

Xcal (see Figure 12.16) is a visual user interface to a calendar database. In many respects it is similar to a variety of calendar applications with the difference being that Xcal also supports voice annotations. Voice annotations may be recorded directly into Xcal but are more likely to be recorded over the telephone via Phoneshell. Voice entries can also be made by cutting and pasting from other

⁷A playback speed of 1.4 times faster than the original seems popular.

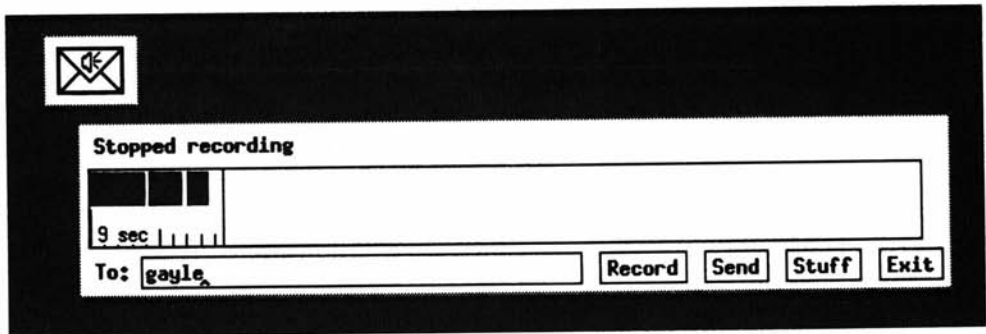


Figure 12.15. Xmemotool, an on-screen voice message taker.

Monday June 7	Tuesday June 8	Wednesday June 9
10:30 – 11:00 Knight–Ridder  1:00 – 1:45 Sun call Debby	4:00 – 5:00 photos for FRAMES TOIS paper overdue	proposal for Emil Ava and Kaya tickets lunch with Paul 3:30 – 4:00 Richard Basch 

Figure 12.16. A personal calender containing voice and text entries.

applications; e.g., a user may select a portion of a voice message from a caller who will visit in a few days and paste that into the calendar as a reminder of the visit or to suggest topics for discussion during the visit.

Another application, ToDo, shown in Figure 12.7, combines voice and text in a things-to-do list. A user can enter voice or text notes while at the workstation, but voice notes more frequently originate from telephone interactions with Phoneshell. While accessing voice mail in PhoneShell, a user records a “memo”; this sound file is then incorporated into the ToDo database.

While each of these applications is fairly basic in isolation, it is their combination on a single workstation screen that renders them most powerful. In practice, little of the audio data used in these applications is recorded at the workstation; text is usually preferred simply because it is easier to retrieve at a later date. Instead, voice annotations arrive over the telephone into the databases used by these applications, either in the user’s own voice via Phoneshell or as portions of voice messages from other people through audio cut and paste. Although none of

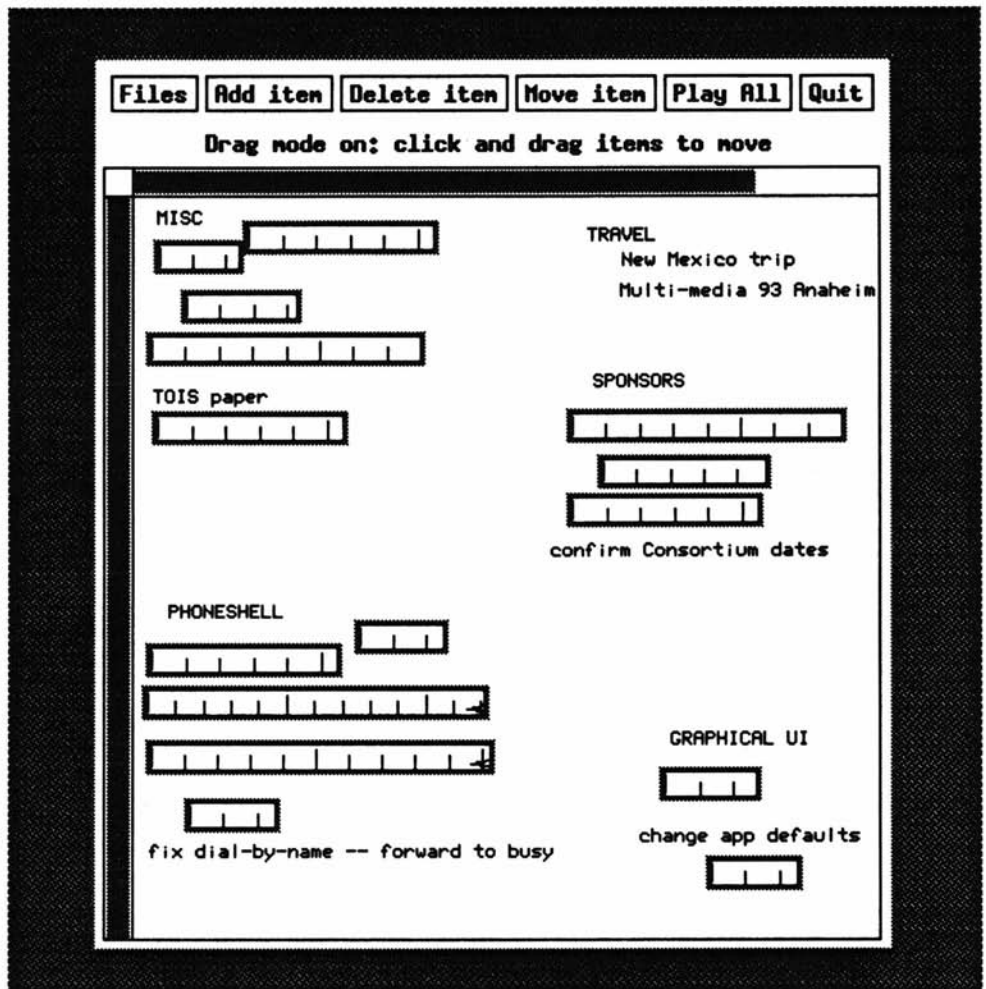


Figure 12.17. A personal project manager containing voice as well as text entries. Voice entries are most often recorded over the telephone using Phoneshell.

these applications is overwhelming in isolation, the appropriateness and ease with which they can be used in combination hints at the ultimate utility of desktop audio.

SUMMARY

This chapter builds on the previous chapter's point of view of unifying telephones and computers by presenting the concept of Desktop Audio, the integration of voice processing into screen-based desktop computing. It started by arguing that

the successful deployment of voice applications depends on a number of factors. The most important factor for success is that voice must add value to existing work practices, as it is unlikely that any single new application is going to radically change the ways in which office computers are being used. But it can enhance current productivity tools, e.g., by providing better user interfaces to current voice systems such as voice mail. Desktop voice processing can enable remote access over the telephone to desktop applications and their databases as was described in the case study on Phoneshell. The case study on Conversational Desktop was meant to illustrate this broad range of potential Desktop Audio functionality, and how it benefits from the integration of multiple voice processing technologies.

Desktop Audio requires graphical user interfaces to indicate the presence of stored voice within an application and to control its playback. Graphical interfaces can also support cut-and-paste of stored voice between applications. Common interfaces are the "button," which implements click-to-play while requiring little screen space, and more elaborate audio playback control panels which may also provide a temporal representation of the stored voice. The SoundViewer, described as a case study, attempts to provide the enhanced functionality of control panels while minimizing its screen space requirements. The SoundViewer was shown in use in an array of applications discussed in the final case study.

Desktop Audio also benefits from a client-server architecture, which enables multiple applications to make use of digital audio resources simultaneously without interfering with each other. Audio toolkits should insulate clients from details of the audio server's protocol and provide convenient user interface building blocks to enable rapid application development. A number of server architectures were discussed in detail.

Desktop Audio has become a reality only very recently, driven primarily by the increased speeds of workstations. Issues with appropriateness of speech applications as well as the techniques to employ voice processing technologies effectively all come to bear on this integrated approach. With proper integration, voice can be used seamlessly across a range of tasks we perform in our daily work lives.

Toward More Robust Communication

This book has discussed a variety of speech technologies including digital audio coding, speech recognition, text-to-speech synthesis, and the telephone. Intermingled with chapters explaining the algorithms that provide these capabilities to computers have been chapters devoted to applications and interactive techniques whereby speech may be employed effectively in a computational environment. The emphasis has been on *interaction* and *responsiveness* of voice applications; the case studies expand on these obvious surface features of conversation. But, despite the themes of higher-level discourse presented in Chapter 9, no specific application is capable of conversing at the level of a four-year-old child even when limited to a single topic.

Major constraints to conversational interactions are the limited ability of current speech recognition technology and the marginal fluency of synthetic speech. Chapters 10 and 11 temporarily overlooked these limitations by discussing the role of computers as *facilitators* of unconstrained conversation among people over the telephone. This led to Chapter 12's discussion of the role of audio as data captured on the desktop from sources such as voice mail and recordings of a meeting or telephone call. In some ways these later chapters are disjoint from prior chapters in that they describe situations in which the computer makes no pretense at conversational ability, understanding nothing about the content of the audio it is routing or recording while it assists with setting up a conversation or archiving it. But from the end user's perspective, this disjunction may not be so apparent. We converse in diverse situations and for many purposes; we introduce strangers, initiate conversations, and often listen to others more than we speak. For the user, speech is about getting something done, and the tasks we already perform

among ourselves are more important in defining how we will utilize speech than the limitations of current technology.

This final chapter returns to the issue of increased computer participation in conversation and poses several domains in which computers woefully lag our conversational abilities beyond hearing words or stringing them together into spoken sentences. One component of improved computer support for conversations is a better appreciation of the interactive techniques we employ to ensure robust communication while talking. Another is tighter coupling between the acoustic and conceptual phases of language understanding. Finally, prosody is an essential element of human speech largely overlooked in computational speech systems. However, these are not presented here as unsolved problems but rather as encouraging avenues that may lead to more productive voice interaction between people and computers.

ROBUST COMMUNICATION

In our use of language we employ a variety of techniques to confirm that the other party understands our message. When the other party is a computer communicating via technology of marginal ability, such robustness is even more essential. Most current applications of speech technology are extremely brittle; if they succeed it is mostly due to the cooperation of the user and our remarkable ability to adapt to conversations under conditions of extreme noise or very thick accents. If conversational computers incorporate our own communication protocols, they can become much more capable partners in conversation. This section describes some aspects of our conversational ability that have been largely left out of the design of dialogue systems to date, although many of these ideas were briefly introduced in Chapter 9.

In a seminal and lengthy journal article, Hayes and Reddy note a number of important elements of conversational protocols and identify some additional behaviors that a conversational system might employ [Hayes and Reddy 1983]. They identify the principle of *implicit confirmation*: if the speaker believes that the listener received the message, then the speaker also believes that the listener understood the message in the absence of any response. This places a huge burden on the listener who must indicate any difficulties understanding the message at the proper time or else the speaker may continue and the conversation will rapidly break down. Such expectations imply that speech recognition applications, which may be thought of as a listener at the end of a very noisy communication channel, must always respond in a timely manner to recognition errors.

Clearly it is a violation of the listener's side of conversational protocol to detect that recognition failed and not say anything as this constitutes an explicit acknowledgment that the message was understood. What should be said in response? Humans concisely communicate what has been understood and what information is lacking through a process of progressive clarification. This may be accomplished by asking a series of short, specific questions rather than more general requests for the speaker to repeat or rephrase the original utterance. The

nature of questions that we ask strongly influences the speaker's response, and we should guide the conversation to a successful conclusion. By echoing portions of what has been thought to be understood, the listener also invites correction by the speaker. The principle of implicit confirmation suggests that an echoed statement can be assumed to be correct unless it is contested by the original speaker.

We often describe an object partially, identifying only its salient features. Grice's maxims support this behavior; they indicate that we should speak only as much as is necessary, i.e., if a complete description is not necessary to identify an object, then we should not elaborate. Humans excel at understanding each other from fragmentary utterances, but these sentence fragments can wreck havoc on computational parsers and semantic analyzers. "Conceptual" parsers operate on the basis of keyword detection, e.g., by placing keywords into slots with appropriate roles in a frame. But simply detecting keywords may be insufficient for identifying their role even in a syntactically simple utterance such as "The lion roared at the elephant in the tiger's cage"; here we must identify the syntactic relationships between the animals if we are to infer their roles in the activity described.

A conversational speech system should have the ability to answer questions that may fall into several categories. Hypothetical questions like "If I were to go on Tuesday could I get a lower fare ticket?" are often a prelude to further transaction and establish the basis of the next request. Questions about ability like "Can you pass the salt?" are often indirect speech acts. But other times answering questions about ability is essential as speech systems are not omnipotent but perform in a limited domain. In a graphical user interface, the user can pull down menus and read options in the process of deciding how to make a choice. In a speech system, this is not possible: the application would need to continually recite all possible options because of the transient nature of speech. Instead, the user must be able to ask the system what its capabilities are, and it must be able to reply in a rational manner.

Combining clarifying protocols with the various aspects of discourse described in Chapter 9, we begin to appreciate the complexity of conversation. There may be a round of clarifying discourse to resolve ambiguities about what one party has just said. This contributes to subgoals—mutual understanding on an utterance-by-utterance basis—in the context of the discourse. But discourse is usually not about simply understanding each other; certainly in the case of a human speaking to a computer, the human wishes the computer to perform some action or service. The computer must track the discourse focus and understand the user's goals at each step while formulating its own subgoals to understand the utterance. This is very difficult, yet it provides a powerful basis for graceful conversational interaction.

SPEECH RECOGNITION AND ROBUST PARSING

One disturbing trend in speech recognition research is putting more and more linguistic knowledge into the word recognition process (this is good) but then using this knowledge to excessively constrain recognition, which leaves little

room for speaker or recognizer error. Linguistic constraints, most prominently rules of syntax or word combination probabilities, limit the perplexity of the recognition task and this results in both significantly improved accuracy and the potential for much larger recognition vocabularies. Such constraints are certainly necessary to understand fluent speech and humans use them as well. But once a language is completely specified, what happens when the human errs?

In the majority of current, experimental, large-vocabulary connected speech recognizers, omission of a single word is likely to result in no recognition result reported at all. If the recognizer is configured to hear seven or ten digit telephone numbers and the talker stops after six digits, the recognizer may report nothing or possibly only the occurrence of an error without any hint as to its nature. But consider the conversational techniques discussed earlier in this chapter as well as in Chapter 8. It is extremely unproductive for the application to remain mute or to utter unhelpfully "What was that?" A preferable alternative might be to echo the digits back to the user either in entirety or in part. Another might be to use back-channel style encouragement ("Uh-huh" spoken with a somewhat rising pitch) to indicate that the application is waiting for more digits and anticipates that the user has not yet completed his or her turn. Further misunderstandings might cause the application to explain what it thinks the user has requested and why the request is not yet complete, e.g., "I can dial the phone for you but you must specify at least seven digits for a valid number."

In the above scenario, the human made a clear mistake by speaking only six digits. But a cooperative system will not castigate the user, rather it will try to help complete the task. Perhaps the user accidentally missed a digit or cannot decipher a digit in someone else's handwriting, or paused to obtain reassurance that the system was still listening and realized that a call was being placed. Such occurrences are not even limited to situations in which the human makes such an unequivocal mistake; spoken language is ripe with false starts, ill-formed grammatical constructs, and ellipsis. When these happen, is there any hope for an application using speech recognition to recover gracefully and cooperatively?

One approach, similar to that used by Conversational Desktop and described in Chapter 9, is to define the recognizer's grammar so that all possible sentence fragments appear as valid utterances. When used during the recognition phase of discourse, such an under-constrained grammar would lead to reduced recognition accuracy due to increased perplexity. But humans are effective at conversing to attain a goal and, to the extent that we can engage in question-answering, we are capable of getting acceptable performance even in the face of recognition errors [Hunnicuttt *et al.* 1992] so letting some errors occur and then negotiating with the user may be more productive than insisting on near-perfect recognition before reporting *any* results.

Two approaches from MIT's Spoken Language Systems Group illustrate practical hybrid solutions. One approach reported by [Seneff 1992] employs conventional parsing based on syntactic rules, but when a well-formed parse is not found, the parser switches to a more semantically oriented mode based on analysis of keywords in the context of the task domain. Additionally, conventional syntactic parsing can be improved by weighting word choices with probability

measures. Under this approach, a grammar is augmented with word-sequence probabilities based on analysis of a corpus of utterances spoken by naive subjects attempting to access the chosen task [Hirschman *et al.* 1991].

But in natural speech these probabilities are dynamic and depend on the current state of the conversation as well as the listener's expectations of what the talker may be up to. In a fully integrated conversational application, pragmatic information such as identification of a partially completed plan or detection of elements of a script could tune word probabilities based on expectations of what is likely to follow. Similarly a focus model could suggest heightened probabilities of words relevant to attributes of entities recently discussed in addition to resolving anaphoric references. Domain knowledge plays a role as well, from simplistic awareness of the number of digits in telephone numbers to knowledge of the acceleration and turning capabilities of various types of aircraft in an air traffic control scenario.

Part of the difficulty with flexible parsing is the excessive degree of isolation between the application, its discourse system, and the speech recognition component. For many current systems the recognizer is given a language model in whatever form it requires; it then listens to speech and returns a string of text to the application, which must parse it *again* to know how to interpret the words meaningfully. Not only has syntactic information been needlessly lost when reporting the recognized speech as a string of words, but it also may be detrimental to strip the representation of any remaining acoustic evidence such as the recognizer's degree of certainty or possible alternate choices of recognition results. How can partial recognition results be reported to the parser? Perhaps several noun phrases were identified but the verb was not, confounding classification of the nouns into the possible roles that might be expressed in a framed-based representation.

These observations are intended to suggest that despite this book's portrayal of the various layers of language understanding and generation as distinct entities, they still must be tightly woven into a coherent whole. Isolation of the word identification portion of discourse understanding into a well bounded "speech recognizer" component cannot in the long run support sophisticated conversational systems. Knowledge must be communicated easily across components, and analysis must be flexible and based on dynamic conversation constraints.

PROSODY

Prosody refers to the spoken style of discourse independent of lexical content, and it includes several aspects of how we speak. *Intonation* is the tune of an utterance: how we modulate F0 to change the pitch of our speech. Intonation operates at a sentence or phrase level; the rising tune of a yes-or-no question immediately differentiates it from the falling tune of a declarative statement. Intonation also helps to convey the stress of words and syllables within a sentence as stressed syllables are spoken with a pitch higher or lower than normal, and specific words are emphasized—an important aspect of communicating intent—by stressing

their stressed syllables even more. *Phrasing* is the breaking of speech into groups, how many words we squeeze into an utterance before stopping for breath, or how we may speak a few words much more slowly for emphasis. *Meter* is carried largely by the duration of individual syllables, stressed syllables being generally longer than unstressed ones. Among stressed syllables, some syllables are more stressed than others resulting in a meter scheme across an entire phrase. Syllabic stress is also reinforced by intonation, and intonation helps convey phrasing as well, so all these elements intermingle during speech production.

In a nutshell, prosody encompasses the majority of information lost when comparing an utterance to its transcription, and much of the richness of speech is conveyed by exactly this nonlexical means of expression.¹ Although a back alley in the field, intonation has been explored extensively by linguists attempting to categorize it and understand how it is systematically and predictably used in spoken language [Ladd 1978, Bolinger 1982]. Pierrehumbert and Hirschberg suggest a grammar relating pitch accents to aspects of meaning [Pierrehumbert and Hirschberg 1990]. In their interpretation, intonation indicates factors such as the salience of an utterance, the speaker's degree of involvement or belief in the facts being proposed, and paths of inference the speaker wishes to emphasize. Although prosody is what differentiates spoken from written language, few conversational speech systems have attempted to exploit its expressiveness.

We notice the prosody of synthetic speech mostly by its absence or occasional misplaced syllabic stress. In Chapter 5 intonation was discussed in two contexts: differentiating alternate syntactic forms of words spelled identically but stressed distinctly and differentiating declarative and interrogative sentences by overall pitch contours. This analysis did not extend beyond individual sentences. The difficulty with applying prosodic cues to human-authored text lies chiefly with the fact that intonation conveys so many levels of meaning. Each sentence would need to be parsed to identify the syntactic role of some of the confusing forms, e.g., "live," "elaborate," and "conduct." But without understanding the semantic structure of a sentence, it is difficult to identify which word should be most stressed. And a sentence must be considered in a discourse context to correctly use prosody to convey the difference between given and new information (new is more heavily stressed).

Better use of prosodic cues for speech synthesis can be made by applications employing synthesis-from-concept techniques, i.e., generating utterances in a discourse based on internal models of the discourse and an associated task domain. Witten's Telephone Enquiry Service allowed application programmers to explicitly program intonation into text to be synthesized by marking it with special codes [Witten and Madams 1977]. Davis and Hirschberg used intonational cues to improve the expressiveness of the Direction Assistance program described in Chapter 6 [Davis and Hirschberg 1988]. They used intonational cues to convey given and new information and to cue listeners to shift the focus of discourse by

¹Other information lost by the transcription are the speaker's identity (carried in part by the timbre or spectral characteristics of one's speech), accent, and emotional state.

increasing pitch range. Cahn explored the use of prosody and other acoustical cues to convey affect using synthesized speech; such prosodic cues included pitch range and speech rate as well as variations in how phonemes are realized and acoustical parameters of the synthesizer's vocal tract model [Cahn 1990]. Similar work was also reported by [Murray et al. 1988].

Attempts to employ prosodic cues for language understanding have likewise been limited. Lea proposed a wide ranging framework to take advantage of prosody during speech recognition, including, in part, the observation that stressed syllables are more phonetically invariant than unstressed ones (unstressed syllables are subject to reduction, e.g., to a schwa) [Lea 1980]. More recently, Waibel investigated the role of prosody in speech recognition suggesting that it could be used as a cue to word boundaries [Waibel 1988].

Grunt, described in Chapter 9, detected monosyllabic questions and responded to them according to its discourse model. Daly and Zue analyzed longer utterances' pitch contours in an attempt to differentiate questions of the sort which expect yes-or-no answers from Wh- questions [Daly and Zue 1990]. They achieved significant (though imperfect) results largely by looking at the final boundary tone or pitch excursion at the end of an utterance.

These are but small steps into the realm of intonation. The role of intonation in language is far from completely understood, and detection of prosody may be acoustically difficult especially in the absence of clear lexical analysis of an utterance. Nonetheless, prosody is an essential and powerful component of both speaking and listening and key for more natural dialog systems.

WHAT NEXT?

This brief chapter has suggested some ways in which we have only begun to tap into the richness and robustness of conversation as a potential means of interacting with computer systems. But this is meant to be an optimistic note not a pessimistic one. The very richness of the human voice and its pervasiveness across so much of our expression mean that any ability to exploit it has potential for rewards.

Although the speech technologies described in this book are feeble when compared with human capabilities, the case studies demonstrate that with careful matching of technology to task and careful crafting of interaction techniques successful voice applications are already a reality. Speech is so powerful that even applications of very limited ability can be extremely effective in specific situations.

Speech technologies are improving rapidly, assisted by ever-increasing computer power. Raw technologies are an enabling factor and guarantee success only with careful consideration of how and when to apply them. That modern technologies impinge on only the most basic aspects of our attempts to converse clearly shows the power that conversational computing systems are one day destined to achieve.

Bibliography

- Ades, S. and D. C. Swinehart. "Voice Annotation and Editing in a Workstation Environment." In *Proceedings of the 1986 Conference*, pages 13–28, San Jose, CA: The American Voice I/O Society, September 1986.
- Allen, J. *Natural Language Understanding*. Benjamin/Cummings Series in Computer Science. Reading, MA: The Benjamin/Cummings Publishing Co., Inc., 1987.
- Allen, J., M. S. Hunnicutt, and D. Klatt. *From Text to Speech: The MITalk system*. Cambridge University Press, 1987.
- Allen, J. "Overview of Text-to-Speech Systems." In S. Furui and M. M. Sondhi, editors, *Advances in Speech Signal Processing*, pages 741–790. New York: Marcel Dekker, Inc., 1992.
- Allen, R. B. "Composition and Editing of Spoken Letters." *International Journal of Man/Machine Studies*, 19:181–193, 1983.
- Allport, D. A., B. Antonis, and P. Reynolds. "On the Division of Attention: a Disproof of the Single Channel Hypothesis." *Quarterly Journal of Experimental Psychology*, 24:225–235, 1972.
- Angebrannt, S. R. L. Hyde, D. H. Luong, N. Siravara, and C. Schmandt. "Integrating Audio and Telephony in a Distributed Workstation Environment." In *Proceedings of the Summer 1991 USENIX Conference*, pages 419–435, Berkeley, CA: USENIX Association, June 1991.
- Arons, B., C. Binding, C. Schmandt, and K. Lantz. "The VOX Audio Server." In *Proceedings of the 2nd IEEE Comsoc International Multimedia Communications Workshop*, New York: IEEE, April 1989.
- Arons, B. "The Design of Audio Servers and Toolkits for Supporting Speech in the User Interface." *Journal of The American Voice I/O Society*, 9:27–41, March 1991.
- Arons, B. Hyperspeech. "Navigating in Speech-Only Hypermedia." In *Proceedings of Hypertext '91*, pages 133–146. New York: ACM, December 1991.

- Arons, B. "Techniques, Perception, and Applications of Time-Compressed Speech." In *Proceedings of the 1992 Conference*, pages 169–177, San Jose, CA: The American Voice I/O Society, September 1992.
- Arons, B. "Tools for Building Asynchronous Servers to Support Speech and Audio Applications." In *Proceedings of the ACM Symposium on User Interface Software and Technology*. New York: ACM, November 1992.
- Arons, B. "Speech Skimmer: Interactively Skimming Recorded Speech." In *Proceeding of the ACM Symposium on User Interface Software and Technology*. New York: ACM, November 1993.
- Austin, J. L. *How to do Things with Words*. Oxford, England: Clarendon Press, 1962.
- Beasley, D. S. and J. E. Maki. "Time- and Frequency-Altered Speech." In N.J. Lass, editor, *Contemporary Issues in Experimental Phonetics*, pages 419–458. Academic Press, 1976.
- Biermann, A. W., R. D. Rodman, D. C. Rubin, and F. F. Heidlage. "Natural Language With Discrete Speech as a Mode for Human-to-Machine Communication." *Communications of the ACM*, 28(6):628–636, 1985.
- Billman, R., R. Hammond, P. McElbatton, E. N. Brandt, F. Sung, and N. Yost. "Workstation Audio in the X Environment." *The X Resource*, 1(4):137–158, October 1992.
- Bly, S. A. and J. K. Rosenberg. "A Comparison of Tiled and Overlapping Windows." In *Proceedings of the Conference on Computer Human Interface*, pages 101–106, New York: ACM, April 1986.
- Bolinger, D. "Intonation and Its Parts." *Language*, 58(3):505–533, 1982.
- Bolt, R. A. "'Put-That-There': Voice and Gesture at the Graphics Interface." *Computer Graphics*, 14(3):262–270, 1980.
- Bowles, I., R. Kamel, et al. "PX: Integrating Voice Communications With Desktop Computing." *Journal of The American Voice I/O Society*, 9:1–19, March 1991.
- Briley, B. E. *Introduction to Telephone Switching*. Reading, MA: Addison-Wesley Publishing Co., Inc., 1983.
- Cahn, J. E. "The generation of affect in Synthesized Speech." In *Journal of the American Voice I/O Society*, pages 1–19. San Jose, CA: American Voice I/O Society, July 1990.
- Chalfonte, B. L., R. S. Fish, and R. E. Kraut. "Expressive Richness: A Comparison of Speech and Text as Media for Revision." In *Proceedings of the Conference on Computer Human Interaction*, pages 21–26. New York: ACM, April 1991.
- Chen, F. R. and M. Withgott. "The Use of Emphasis to Automatically Summarize Spoken Discourse." In *Proceedings of the 1992 International Conference on Acoustics, Speech and Signal Processing*, pages 229–233. New York: IEEE, 1992.
- Chorafas, D. N. *Telephony: Today and Tomorrow*. Prentice-Hall Series in Data Processing Management. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1984.
- Clark, H. H. and C. R. Marshall. "Definite Reference and Mutual Knowledge." In Webber Joshi and Sag, editors, *Elements of Discourse Understanding*, 10:63. Cambridge University Press, 1981.
- Clark, H. H. and D. Wilkes-Gibbs. "Referring as a Collaborative Process." *Cognition*, 22:1–39, 1986.
- Clark, H. H. and E. F. Schaefer. "Contributing to Discourse." *Cognitive Science*, 13:259–294, 1989.
- Clark, H. H. and S. E. Brennan. "Grounding in Communication." In L. B. Resnick, J. Levine, and S. D. Behrend, editors, *Socially Shared Cognition*. American Psychological Association, 1990.
- Cohen, P. R., J. Morgan, and M. E. Pollack, editors. *Intentions in Communication*. Cambridge, MA: MIT Press, 1990.
- Crowley, T. "Voice in a Multimedia Document System." *Journal of The American Voice I/O Society*, 9:21–26, March 1991.

- Daly, N. A. and V. W. Zue. "Acoustic, Perceptual, and Linguistic Analyses of Intonation Contours in Human/Machine Dialogues." In *Proceedings of the International Conference on Spoken Language Processing*, November 1990.
- Davis, J. R. *Back Seat Driver: Voice assisted automobile navigation*. Ph.D. thesis, MIT Media Arts and Sciences Section, September 1989.
- Davis, J. R. and T. F. Trobaugh. Direction Assistance. Technical Report 1, MIT Media Laboratory Speech Group, December 1987.
- Davis, J. R. and J. Hirschberg. "Assigning Intonational Features in Synthesized Spoken Directions." In *Proceedings of the Association for Computational Linguistics*, pages 187–193, 1988.
- Davis, J. R. and C. Schmandt. "The Back Seat Driver: Real Time Spoken Driving Instructions." In *Vehicle Navigation and Information Systems*, pages 146–150, 1989.
- Davis, J. R. and C. Schmandt. "Discourse Strategies for Conversations in Time." In *Proceedings of the AVIOS 1990 Conference*, pages 21–26, San Jose, CA: The American Voice I/O Society, September 1990.
- Davis, J. R. "Let Your Fingers Do the Spelling: Implicit Disambiguation of Words Spelled With the Telephone Keypad." *Journal of The American Voice I/O Society*, 9:57–66, March 1991.
- Denes, P. B. and E. N. Pinson. *The Speech Chain: The physics and biology of spoken language*, 2nd edition. New York: W.H. Freeman, 1993.
- Dixon, N. R. and T. B. Martin, editors. *Automatic Speech & Speaker Recognition*. IEEE Press Selected Reprint Series. New York: IEEE Press, 1979.
- Duncan, S., Jr. "Some Signals and Rules for Taking Speaking Turns in Conversations." *Journal of Personality and Social Psychology*, 23:283–292, 1972.
- Duncan, S., Jr. "On the Structure of Speaker-Auditor Interaction During Speaking Turns." *Language in Society*, 2:161–180, 1974.
- Egan, J. P. "Articulation Testing Methods." *Laryngoscope*, 58:955–991, 1948.
- Englebeck, G. and T. L. Roberts. "The Effect of Several Voice-Menu Characteristics on Menu-Selection Performance." Technical report, U.S. West Advanced Technologies, Boulder, CO, 1989.
- Fairbanks, G. "Test of Phonemic Differentiation: The Rhyme Test." *Journal of the Acoustical Society of America*, 30:596–600, 1958.
- Fish, R. S., R. E. Kraut, M. D. P. Leland and M. Cohen. "Quilt: a Collaborative Tool for Cooperative Writing." In *Proceedings of the Conference on Office Information Systems*, pages 30–37, New York: ACM 1988.
- Flanagan, J. L. "Analog Measurements of Sound Radiation From the Mouth." *Journal of the Acoustical Society of America*, 32(12), 1960.
- Flanagan, J. L. *Speech Analysis Synthesis and Perception*. Springer Verlag, 1972.
- Flanagan, J. L., M. R. Schroeder, B. S. Atal, R. E. Crochiere; N. S. Jayant, and J. M. Tribolet. "Speech Coding." *IEEE Transactions on Communications*, 27(4):710–737, April 1979.
- Fletcher, H. and W. A. Munson. "Loudness, Definition, Measurements and Calculation." *Journal of the Acoustical Society of America*, 5, 1933.
- Fry, D. B. "Experiments in the Perception of Stress." *Language and Speech*, 1(2):126–152, 1958.
- Fujimura, O. and J. Lovins. "Syllables as Concatenative Phonetic Elements." In A. Bell and J. B. Hooper, editors, *Syllables and Segments*, pages 107–120, North Holland, New York, 1978.
- Furui, S. and M. M. Sondhi, editors. *Advances in Speech Signal Processing*. New York: Marcel Dekker, Inc., 1992.
- Gifford, W. S. and D. L. Turock. "The Electronic Receptionist: A Knowledge-Based Approach to Personal Communications." In *Proceedings of the IEEE Conference on Selected Topics in Wireless Communications*. New York: IEEE, June 1992.
- Goffman, E. *Forms of Talk*. University of Pennsylvania Press, 1981.

- Gould, J. D. "How Experts Dictate." *Journal of Experimental Psychology: Human Perception and Performance*, 4(4):648-661, 1978.
- Gould, J. D. and S. J. Boies. "How Authors Think About Their Writing, Dictating, and Speaking." *Human Factors*, 20(4):495-505, 1978.
- Gould, J. D. "Writing and Speaking Letters and Messages." *International Journal of Man/Machine Studies*, 16(2):147-171, 1982.
- Gould, J. D.; J. Conti, and T. Hovanyecz. "Composing Letters With a Simulated Listening Typewriter." *Communications of the ACM*, 26(4):295-308, April 1983.
- Gould, J. D., S. J. Boies, S. Levy, J. T. Richards, and J. Schoonard. "The 1984 Olympic Message System: A Test of Behavioral Principles of System Design." *Communications of the ACM*, 30(9):758-769, September 1987.
- Grice, H. P. "Logic and Conversation." In Cole and Morgan, editors, *Syntax and Semantics: Speech Acts*, volume 3, pages 41-58. Academic Press, 1975.
- Grosz, B. J., A. K. Joshi, and S. Weinstein. "Providing a Unified Account of Definite Noun Phrases in Discourse." In *Proceedings of the 21st conference of the Association for Computational Linguistics*, pages 44-50, 1983.
- Grosz, B. J. and C. L. Sidner. "Attention, Intentions, and the Structure of Discourse." *Computational Linguistics*, 12(3):175-204, 1986.
- Grosz, B. J.; K. S. Jones, and B. L. Webber, editors. *Readings in Natural Language Processing*. Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1986.
- Handel, W. S. *Listening: An Introduction to the Perception of Auditory Events*. Cambridge, MA: MIT Press, 1989.
- Hauptmann, A. G. "Speech and Gestures for Graphic Image Manipulation." In *Proceedings of the Conference on Computer Human Interface*, pages 241-245, New York: ACM, May 1989.
- Hayes, P. J. and R. Reddy. "Steps Towards Graceful Interaction in Spoken and Written Man-Machine communication." *International Journal of Man/Machine Systems*, 19:231-284, 1983.
- Hindus, D. "Semi-Structured Capture and Display of Telephone Conversations." Master's thesis, MIT, February 1992.
- Hindus D., C. Schmandt, and C. Horner. "Capturing, Structuring, and Presenting Ubiquitous Audio." *Transactions on Information Systems*, New York: ACM October 1993.
- Hindus, D. and C. Schmandt. "Ubiquitous Audio: Capturing Spontaneous Collaboration." In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 210-217 New York: ACM, 1992.
- Hirschman, L., S. Seneff, D. Goodine, and M. Phillips. "Integrating Syntax and Semantics Into Spoken Language Understanding." In *Proceedings of the Fourth DARPA Speech and Natural Language Workshop*, February 1991.
- Hodges, M. E., R. M. Sasnett, and M. S. Ackerman. "A Construction Set for Multimedia Applications." *IEEE Software*, pages 37-43, January 1989.
- Hunnicutt, S., L. Hirschman, J. Polifroni, and S. Seneff. "Analysis of the Effectiveness of System Error Messages in a Human-Machine Travel Planning Task." In *Proceedings of the Second International Conference on Spoken Language Processing*, October 1992.
- Jacob, R. J. K. "The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look at is What You Get." *ACM Transactions on Information Systems*, 9(3):152-169, April 1991.
- Jelinek, F. "The Development of an Experimental Discrete Dictation Recognizer." In *Proceedings of the IEEE*, volume 73, pages 1616-1624. New York: IEEE, November 1985.
- Kamel, R., K. Emami, and R. Eckert. "PX Supporting Voice in Workstations." *IEEE Computer*, 23(8):73-80, August 1990.
- Klatt, D. H. "Software for a Cascade/Parallel Formant Synthesizer." *Journal of the Acoustical Society of America*, 67(3):971-990, March 1980.

- Klatt, D. H. "Review of Text-to-Speech Conversion for English." *Journal of the Acoustical Society of America*, 82:737-783, September 1987.
- Klatt, D. H. "Review of Text-to-Speech Conversion for English." *Journal of the Acoustical Society of America*, 82:737-793, September 1987.
- Koons, D.B, C.J. Sparrel, and K.R. Thorisson. "Integrating Simultaneous Input from Speech, Gaze and Hand Gesture." In T.M. Maybury, editor, *Intelligent Multi-Media Interfaces*, Menlo Park: AAAI Press, 1993.
- Kramer, M., G. Ramirez, D. L. Turock, and R. S. Wolff. "Use of Two-Way Wireless Messaging for Personal Telephone Management." In *Proceedings of ICUPC '93*, October 1993.
- Kraut, R. E., S. H. Lewis, and L. W. Swezey. "Listener Responsiveness and the Coordination of Conversation." *Journal of Personality and Social Psychology*, 42(4):718-731, 1982
- Kraut, R. E. and S. H. Lewis. "Some functions of feedback in conversation." In H. E. Syphers and J. L. Applegate, editors, *Communication by Children and Adults*, Chapter 9. Sage, 1984.
- Ladd, D. R. *The Structure of Intonational Meaning*. Indiana University Press, 1978.
- Ladefoged, P. *Elements of Acoustic Phonetics*. Chicago:University of Chicago Press, 1962.
- Lamming, M. and W. Newman. "Activity-Based Information Retrieval: Technology in Support of Human Memory." Technical Report 91-03, Rank Xerox EuroPARC, February 1991.
- Lea, W. A. "Prosodic Aids to Speech Recognition." In Wayne A. Lea, editor, *Trends in Speech Recognition*, Chapter 8. Prentice Hall, 1980.
- Lee, K. *Large-Vocabulary Speaker-Independent Continuous Speech Recognition Using Hidden Markov Models*. Ph.D. thesis, Carnegie-Mellon University, 1988.
- Lee, K. and H. Hon. "Large-Vocabulary Speaker-Independent Continuous Speech Recognition Using HMM." In *International Conference on Acoustics, Speech, and Signal Processing*. New York: IEEE, April 1988.
- Lee, K. and F. Alleva. "Continuous Speech Recognition." In Sadaoki Furui and M. Mohan Sondhi, editors, *Advances in Speech Signal Processing*, pages 623-650. New York: Marcel Dekker, Inc., 1992.
- Leland, M. D. P., R. S. Fish, and R. E. Kraut. "Collaborative Document Production Using Quilt." In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 206-215, New York: ACM, 1988.
- Levinson, S. C. *Pragmatics*. Cambridge University Press, Cambridge, Great Britain, 1983.
- Lison, H. and T. Crowley. "Sight and Sound." *UNIX Review*, 7(10):76-86, October 1989.
- Luce, P. A., T. C. Feustel, and D. B. Pisoni. "Capacity Demands in Short-Term Memory for Synthetic and Natural Speech." *Human Factors*, 25(1):17-32, 1983.
- Mackay, W. E., T. W. Malone, K. Crowston, R. Rao, D. Rosenblitt, and S. K. Card. "How do Experienced Information Lens Users Use Rules?" In *Proceedings of the Conference on Computer Human Interface*, pages 211-216, New York: ACM 1989.
- Malone, T. W., K. R. Grant, K. Lai, R. Rao, and D. Rosenblitt. "Semi-Structured Messages are Surprisingly Useful For Computer-Supported Coordination." *ACM Transactions on Office Information Systems*, 5(2):115-131, 1987.
- Martin, G. L. "The Utility of Speech Input in User-Computer Interfaces." *International Journal of Man/Machine Studies*, 30:355-375, 1989.
- Maxemchuk, N. F. "An Experimental Speech Storage and Editing Facility." *The Bell System Technical Journal*, 59(8):1383-1395, October 1980.
- McPeters, D. L. and A. L. Tharp. "The Influence of Rule-Generated Stress on Computer-Synthesized Speech." *International Journal of Man/Machine Studies*, 20:215-226, 1984.
- Medina-Mora, R., T. Winograd, R. Flores, and F. Flores. "The action Workflow Approach to Workflow Management Technology." In *Proceedings of the Conference on Computer-Supported Cooperative Work*, New York: ACM, November 1992.

- Mermelstein, P. "Automatic Segmentation of Speech into Syllabic Units." *Journal of the Acoustical Society of America*, 58(4):880-883, October 1975.
- Mermelstein, P. "G.722, a new CCITT coding standard for digital transmission of wideband audio signals." *IEEE Communications Magazine*, 26(1), 1988.
- Minneman, S. L. and S. A. Bly. "Managing a Trois: a Study of a Multi-User Drawing Tool in Distributed Design Work." In *Proceedings of the Conference on Computer Human Interaction*, pages 217-224. New York: ACM, April 1991.
- Minsky, M. "A Framework for Representing Knowledge." In P. Winston, editor, *The Psychology of Computer Vision*, pages 211-277. McGraw-Hill, New York, 1975.
- Morris, J. H., et al. "Andrew: A Distributed Personal Computing Environment." *Communications of the ACM*, March 1986.
- Muller, M. J. and J. E. Daniel. "Toward a Definition of Voice Documents." In *Proceedings of the 1990 Conference on Office Information Systems*, 1990.
- Murray, I. R., J. L. Arnott, and A. F. Newell. "Hamlet—Simulating Emotion in Synthetic Speech." In *Speech '88; Proceedings of the 7th FASE Symposium*. Institute of Acoustics, Edinburgh, 1988.
- Myers, B. A. "The Importance of Percent-Done Progress Indicators for Computer-Human Interfaces." In *Proceedings of the Conference on Computer Human Interface*, pages 11-17, New York: ACM, 1985.
- Nakatani, L. H., et al. "TNT: a Talking Tutor 'n' Trainer for Teaching the Use of Interactive Computer Systems." In *Proceedings of the Conference on Computer Human Interaction*, New York: ACM, 1986.
- Nicholson, R. T. "Usage Patterns in an Integrated Voice and Data Communications System." In *ACM Transactions on Office Information Systems*, volume 3, pages 307-314. New York: ACM, July 1985.
- Nusbaum, H. C., M. J. Dedina, and D. B. Pisoni. "Perceptual Confusions of Consonants in Natural and Synthetic CV Syllables." *Speech Research Lab Tech. Note 84-02*, Indiana University, 1984.
- Nye, P. W. and J. Gaitenby. "The Intelligibility of Synthetic Monosyllabic Words in Short, Syntactically normal sentences." *Haskins Lab. Status Report on Speech Research*, 38:169-190, 1974.
- Nyquist, H. "Certain Factors Affecting Telegraph Speed." *Bell System Technical Journal*, 3:324-326, 1924.
- Ochsmann, R. B. and A. Chapanis. "The Effects of 10 Communication Modes on the Behavior of Teams During Co-Operative Problem-Solving." *International Journal of Man/Machine Studies*, 6:579-619, 1974.
- Olive, J. P. "Rule Synthesis of Speech from Diadic Units." In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 568-570. New York: IEEE, 1977.
- O'Neill, E. F. "TASI—Time Assignment Speech Interpolation." *Bell Laboratories Record*, 37:83-87, March 1959.
- Orr, D. B., H. L. Friedman, and J. C. Williams. "Trainability of Listening Comprehension of Speeded Discourse." *Journal of Educational Psychology*, 56:148-156, 1965.
- O'Shaughnessy, D. *Speech Communication*. New York: Addison-Wesley Publishing Company, 1987.
- O'Shaughnessy, D. "Recognition of Hesitations in Spontaneous Speech." In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 1521-1524. New York: IEEE, March 1992.
- Oshika, B. T., V. W. Zue, R. V. Weeks, H. Neu, and J. Aurbach. "The Role of Phonological Rules in Speech Understanding Research." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23(1):104-112, 1975.

- G. Peterson, W. Wang, and E. Sivertsen. "Segmentation Techniques in Speech Synthesis." *Journal of the Acoustical Society of America*, 30:739-742, 1958.
- Pierrehumbert, J. and J. Hirschberg. "The Meaning of Intonational Contours in the Interpretation of Discourse." In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, chapter 14, pages 271-311. Cambridge, MA: The MIT Press, 1990.
- Pisoni, D. B. and S. Hunnicutt. "Perceptual Evaluation of MITalk: The MIT Unrestricted Text-to-Speech System." In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 572-575. New York: IEEE, 1980.
- Pisoni, D. B. "Speeded Classification of Natural and Synthetic Speech in a Lexical Decision Task." *Journal of the Acoustical Society of America*, 70:S98, 1981.
- Pisoni, D. B., H. C. Nusbaum, and B. G. Greene. "Perception of Synthetic Speech Generated by Rule." *Proceedings of the IEEE*, 73(11):1665-1676, November 1985.
- Rabiner, L. R. and R. W. Schafer. *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978.
- Rabiner, L. R. and B. H. Juang. "An Introduction to Hidden Markov Models." *IEEE ASSP Magazine*, 3(1):4-16, January 1986.
- Rabiner, L. R. and B. H. Juang. "An Introduction to Hidden Markov Models." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 4-16, January 1986.
- Resnick, P. Hypervoice: "A Phone-Based CSCW Platform." In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 218-225, New York: ACM, November 1992.
- Resnick, P. *Hypervoice: Groupware by Telephone*. Ph.D. thesis, MIT, September 1992.
- Resnick, P. and R. A. Virzi. "Skip and Scan: Cleaning Up Telephone Interfaces." In *Proceedings of the Conference on Computer Human Interface*, New York: ACM, 1992.
- Roberts, T. L. and G. Engelbeck. "The Effects of Device Technology on the Usability of Advanced Telephone Functions." In *Proceedings of the Conference on Computer Human Interface*, pages 331-337, New York, May 1989. ACM.
- Root, R. W. and C. R. Koster. "Experimental Evaluation of a Mnemonic Command Syntax for Controlling Advanced Telecommunications Services." In *Proceedings of the Conference on Computer Human Interaction*, pages 809-813, 1986.
- Root, R. W. and C. Chow. "Multimode Interaction in a Telecommunications Testbed: the Case of Memory Dialing." In G. Salvendy, editor, *Cognitive engineering in the design of human-computer interaction and expert systems*, pages 399-406. Elsevier Science Publishers, 1987.
- Rose, M. T. *The Internet Message*, Chapter 6. Prentice Hall Series in Innovative Technology. Englewood Cliffs, NJ: P T R Prentice Hall, 1993.
- Rosson, M. B. and A. J. Cecala. "Designing a Quality Voice: An Analysis of Listeners' Reactions to Synthetic Voices." In *Proceedings of the Conference on Computer Human Interface*, pages 192-197, New York: ACM, April 1986.
- Roucos, S. and A. M. Wilgus. "High Quality Time-Scale Modifications for Speech." In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 493-496. New York: IEEE, 1985.
- Rudnick, A. I. and A. G. Hauptmann. "Models for Evaluating Interaction Protocols in Speech Recognition." In *Proceedings of the Conference on Computer Human Interface*, pages 285-291, New York: ACM, 1991.
- Rudnick, A. I.; J. M. Lunati, and A. M. Franz. "Spoken Language Recognition in an Office Management Domain." In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 829-832. New York: IEEE, May 1991.
- Sacks, H., E. A. Schegloff, and G. Jefferson. "A Simplest Systematics for the Organization of Turn-Taking for Conversation. Language," 50(4):696-735, 1974. Reprinted in *Studies in the Organization of Conversational Interaction*, J. Schenken, ed., Academic Press 1978.

- Sakoe, H. and S. Chiba. "Dynamic Programming Algorithm Optimization for Spoken Word Recognition." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, February 1978. Reprinted in Dixon and Martin.
- Schank, R. C. and R. Abelson. *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Lawrence Erlbaum, 1977.
- Schmandt, C. "The Intelligent Ear: A Graphical Interface to Digital Audio." In *Proceedings IEEE Conference on Cybernetics and Society*, pages 393–397, New York: IEEE, October 1981.
- Schmandt, C. and E. Hulstén. "The Intelligent Voice Interactive Interface." *Human Factors in Computer Systems*, pages 363–366, 1982.
- Schmandt, C. "Speech Synthesis Gives Voiced Access to an Electronic Mail System." *Speech Technology*, 2(3):66–69, 1984.
- Schmandt, C. and B. Arons. "A Conversational Telephone Messaging System." *IEEE Trans. on Consumer Electr.*, CE-30(3):xxi–xxiv, 1984.
- Schmandt, C. and B. Arons. "Phone Slave: A Graphical Telecommunications Interface." *Proc. of the Soc. for Information Display*, 26(1):79–82, 1985.
- Schmandt, C., B. Arons, and C. Simmons. "Voice Interaction in an Integrated Office and Telecommunications Environment." In *Proceedings of the 1985 Conference*, pages 51–57. San Jose, CA: American Voice I/O Society, September 1985.
- Schmandt, C. and B. Arons. "A Robust Parser and Dialog Generator for a Conversational Office System." In *Proceedings of the 1986 Conference*, pages 355–365. San Jose, CA: American Voice I/O Society, September 1986.
- Schmandt, C. "Employing Voice Back Channels to Facilitate Audio Document Retrieval." In *Proceedings of the ACM Conference on Office Information Systems*, pages 213–218., New York: ACM, 1988.
- Schmandt, C. and M. McKenna. "An Audio and Telephone Server for Multi-Media Workstations." In *Proceedings of the 2nd IEEE Conference on Computer Workstations*, pages 150–159. New York: IEEE, March 1988.
- Schmandt, C. and S. Casner. "Phonetool: Integrating Telephones and Workstations." In *Proceedings of the IEEE Global Telecommunications Conference*, pages 970–974. New York: IEEE Communications Society, November 1989.
- Schmandt, C., D. Hindus, M. Ackerman, and S. Manandhar. "Observations on Using Speech Input for Window Navigation." In *Proceedings of the IFIP TC 13 Third International Conference on Human-Computer Interaction*, pages 787–793, August 1990.
- Schmandt, C. "Phoneshell: the Telephone as Computer Terminal." In *Proceedings ACM Multimedia 93*, pages 373–382. New York: ACM, 1993.
- Schmandt, C., M. S. Ackerman, and D. Hindus. "Augmenting a Window System With Speech Input." *IEEE Computer*, 23(8):50–56, August 1990.
- Schwab, E. C., H. C. Nusbaum, and D. B. Pisoni. "Some Effects of Training on the Perception of Synthetic Speech." *Human Factors*, 27(4):395–408, 1985.
- Searle, J. R. "The Classification of Illocutionary Acts." *Language in Society*, 5:1–24, 1976.
- Seneff, S. "A Joint Synchrony/Mean-Rate Model of Auditory Speech Processing." *Journal of Phonetics*, 16:55–76, 1988.
- Seneff, S. "Robust Parsing for Spoken Language Systems." In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. New York: IEEE, March 1992.
- Sivertsen, E. "Segment Inventories for Speech Synthesis." *Language and Speech*, 4:27–90, 1961.
- Snell, J. "The Lucasfilm Real-Time Console for Recording Studios and Performance of Computer Music." *Computer Music Journal*, 6(3):33–45, Fall 1982.
- Spiegel, M. F. "Pronouncing Surnames Automatically." In *Proceedings of the 1985 Conference*. San Jose, CA: American Voice I/O Society, September 1985.

- Spiegel, M. F. "Using a Monosyllabic Test Corpus to Evaluate the Intelligibility of Synthesized and Natural Speech." In *Proceedings of the 1988 Conference*, San Jose, CA: The American Voice I/O Society, 1988.
- Spiegel, M. F. and M. J. Macchi. "Synthesis of Names by a Demisyllable-Based Speech Synthesizer (Orator)." *Journal of The American Voice I/O Society*, 7:1-10, March 1990.
- Spiegel, M. F., M. J. Macchi, M. J. Altom, D. Kahn, N Jackson, and K. D. Gollhardt. "State-of-the-Art Name Pronunciation Accuracy: Preparing the ORATOR Synthesizer for Applications." In *Proceedings of the 1991 Conference*, pages 119-123, San Jose, CA: The American Voice I/O Society, September 1991.
- Stallings, W. *ISDN: An Introduction*. New York: Macmillan Publishing Company, 1989.
- Starker, I. and R. A. Bolt. "A Gaze-Responsive Self-Disclosing Display." In *Proceedings of the Conference on Computer Human Interface*, pages 3-9. New York: ACM, April 1990.
- Stifelman, L. J., B. Arons, C. Schmandt, and E. A. Hulteen. Voice Notes: "A Speech Interface for a Hand-Held Voice Notetaker." In *Proceedings of INTERCHI '93*, New York: ACM, April 1993.
- Streeter, L. A., D. Vitello, and S. A. Wonsiewicz. "How to Tell People Where to Go: Comparing Navigational Aids." *International Journal of Man/Machine Systems*, 22(5):549-562, May 1985.
- Talley, D. *Basic Carrier Telephony*. Rochelle Park, NJ: Hayden Book Company, Inc., second edition, 1966.
- Terry, D. B. and D. C. Swinehart. "Managing Stored Voice in the Etherphone System." *ACM Transactions on Computer Systems*, 6(1):3-27, February 1988.
- Thomas, R. H., H. C. Forsdick, T. R. Crowley, R. W. Schaaf, R. S. Tomlinson, V. M. Travers, and G. G. Robertson. "Diamond: A Multimedia Message System Built on a Distributed Architecture." *IEEE Computer*, 18(12):65-78, December 1985.
- Treisman, A. and A. Davies. "Divided Attention to Ear and Eye." In *Attention and Performance*, volume IV, pages 101-117, 1973.
- Vitale, T. "An Algorithm for High Accuracy Name Pronunciation by Parametric Speech Synthesizer." *Journal of Computational Linguistics*, 17(1):257-276, 1991.
- Waibel, A. *Prosody and Speech Recognition*. Morgan Kaufmann, 1988.
- Waibel, A. and K. Lee, editors. *Readings in Speech Recognition*. San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1990.
- Want, R. and A. Hopper. "Active Badges and Personal Interactive Computing Objects." *IEEE Transactions on Consumer Electronics*, 38(1):10-20, Feb 1992.
- Weimer, D. and S. K. Ganapathy. "A Synthetic Visual Environment With Hand Gesturing and Voice Input." In *Proceedings of the Conference on Computer Human Interface*, pages 235-240, New York, ACM, May 1989.
- Weimer, D. and S. K. Ganapathy. "Interaction Techniques Using Hand Tracking and Speech Recognition." In M.M. Blattner and R.B. Dannenberg, editors, *Multimedia Interface Design*, ACM Press frontier series, pages 109-126. New York: Addison-Wesley, 1992.
- Wickens, C. D., S. J. Mountford, and W. Schreiner. "Multiple Resources, Task-Hemispheric Integrity, and Individual Differences in Time-Sharing." *Human Factors*, 23:211-230, 1981.
- Wilpon, J. G., L. R. Rabiner, C. Lee, and E. R. Goldman. "Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(11), November 1990.
- Winograd, T. *Language as a Cognitive Process - Syntax*. New York: Addison-Wesley, 1983.
- Winograd, T. "A Language/Action Perspective on the Design of Cooperative Work." *Human Computer Interaction*, 3(1):3-30, 1988.
- Witten, L. and P. Madams. "The Telephone Inquiry Service: A Man-Machine System Using Synthetic Speech." *International Journal of Man/Machine Systems*, 9:449-464, 1977.
- Witten, I. H. *Principles of Computer Speech*. Academic Press, 1982.

- Wong, C. C. "Personal Communications". Master's thesis, MIT, June 1991.
- Yost, W. and D. Nielsen. *Fundamentals of Hearing*. Holt, Rinehart, and Winston, 1977.
- Yngve, V. H. "On Getting a Word in Edgewise." In *Papers from the Sixth Regional Meeting*, pages 567–578. Chicago Linguistic Society, 1970.
- Zellweger, P. T. "Scripted Documents: a Hypermedia Path Mechanism." In *Proceedings of Hypertext '89*, pages 1–14. New York: ACM, 1989.
- Zue, V., J. Glass, M. Phillips, and S. Seneff. "The MIT SUMMIT Speech Recognition System: A Progress Report." In *First DARPA Speech and Natural Language Workshop*, pages 166–178, February 1989.
- Zue, V., N. Daly, J. Glass, *et al.* "The Collection and Preliminary Analysis of a Spontaneous Speech Database." In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 126–134, October 1989.

Index

- μ-law, 45, 220, 224
- acoustics, 9, 11, 28, 146, 164
- Active Badges, 263, 291
- Adaptive Delta Pulse Code Modulation (ADPCM), 49–51, 54–55, 224
- adjacency pair, 195
- aid for the disabled, 97
- air pressure, 11, 12, 20, 29, 37, 102
- aliasing, 38–41
- allophone, 15, 24–25, 91
- anaphora, 197
- Andrew (mail system), 270
- Anglicization, 95
- annotation of text by voice, 65
- answering machines, 61, 230
- Arabic script, 84
- Arpabet, 16
- articulation, 9, 10, 19
- articulatory model of speech, 25
- aspiration, 15, 27
- AT&T Bell Laboratories, 94
- attentional structure, 198
- audio-tex, 62
- auditory system, 28–34
- augmented transition network (ATN), 184, 186
- Automatic Gain Control (AGC), 63
- Back Seat Driver, 121–123
- back-channel, 6, 196–197, 200, 300
- backward-looking center, 198
- bandwidth, 36, 93
 - of speech signal, 42
 - of the telephone, 42, 155, 162, 219
- Bell Northern Research, 239
- Bellcore, 66–67, 94, 237, 239, 265
- BerBell, 239, 249, 263
- bigram grammar, 151
- blind users, 97, 105, 108
- bottom-up parser, 185–186
- branching factor, 150
- breath group, 17, 195
- broadcast nature of speech, 62
- busy signal, 215
- call forwarding, 232
- call progress, 215
- call setup, 211, 218, 240,
- call teardown, 212
- call waiting, 217
- calling number identification, 217, 245
- Carnegie Mellon University, 151, 168, 270
- case grammar, 187, 189
- cellular telephone, 129, 155, 262, 271
- central office (CO), 212–214, 219, 223

- Centrex, 226
- Cepstral Analysis, 14
- circuit switch, 212
- citation form, 83, 88
- coarticulation, 87, 88–89, 94, 139
- codec (coder/decoder), 44, 134, 224, 246, 249
- cognitive load, 96, 101
- compact audio disc, 12, 43, 44
- concatenative synthesis, 93–94
- conference calls, 220
- connected recognition, 139, 159, 204, 205
- connected speech recognizer, 139
- consonant cluster, 17, 94
- consonants, 26–28, 93
- context-free grammar, 182–184, 185–186, 205
- Continuously-Varying Slope Delta Modulation (CVSD), 48–49
- Conversational Desktop, 165, 204–208, 208, 255, 282, 285–287, 300
- conversational implicature, 193–194
- Coordinator, 193
- critical band, 34
- cue phrase, 197
- Custom Local Area Signaling Services (CLASS), 226

- decibel (dB), 33
- deixis, 123, 168, 197
- Delta Modulation, 46
- desktop audio, 268–296
- demi-syllable, 93
- dial tone, 214
- Diamond, 65–66
- dictation, 63–64, 157–158
- dictionary, *See* lexicon
- Digital Equipment Corporation, 276
- digital signal processor, 12, 37, 93, 232, 246
- diphone, 93
- diphthong, 26
- Direction Assistance, 117–121, 200
- discourse, 9, 194–199
- discourse segment, 197
- discrete speech recognizer, 139, 158, 159, 161
- distinctive ring, 218, 223, 233, 236
- divided attention, 105, 156, 174, 175
- documents, voice, 64–68
- dual tone multifrequency signal (DTMF). *See* touchtone
- duration of syllables, 87, 91, 93, 94
- dynamic programming. *See* dynamic time warping
- dynamic range, 44–46
- dynamic time warping (DTW), 137, 141–144

- echo cancellation, 211, 219
- editing, 235
 - and coder algorithms, 54–55
 - of voice, 69–81
 - of text, 159
- electronic mail (email) 65–66, 82, 108, 108, 110, 111, 124–130, 127, 157, 231, 244, 252, 271, 288–289
- encryption, 37
- English, 84, 95, 180
- error
 - of speech coders, 43, 54
 - recognition, 141, 158, 160–169, 174, 186, 204, 205, 299–301
- Etherphone, 67–68, 77, 234–237, 249, 250, 262, 275, 283
- expressiveness, 6, 65, 104
- eye trackers, 168

- filtering, 92, 132
 - of electronic mail, 129, 290
 - of information, 106, 124–126
- Fletcher-Munson curves, 33
- focus, 197–199, 207
- formants, 23–24, 25, 26, 27, 28, 34, 91–93, 149
- frame, 176, 191, 256
- fricatives, 21–22, 27, 89
- fundamental frequency, (F0) [note: (F zero)]. *See* pitch

- G.722, 224
- German, 85
- gesture, 168, 176, 196
- glides, 28
- glottal pulses, 28
- grammar, 138, 180, 185, 205–206
- graphical interfaces to stored sound, 282–285
- graphical interfaces to stored speech, 60–81, 271–273, 292–295
- Grice's maxims of conversation, 193–194, 299
- Grunt, 199–204, 208, 303

- hand-held device, computer, 155–156, 165
- Harvard psychoacoustic sentences, 95
- Haskins syntactic sentences, 95
- Hawaiian, 85
- hearing, 28–34
- Hidden Markov Models (HMM), 133, 141, 144–147, 152
- Hyperspeech, 277

- IBM, 151
- indirect speech acts, 192

- Integrated Services Digital Network (ISDN), 221–224, 221–226, 228, 240, 244–249, 263
- Intelligent Ear, 74–75, 282
- intelligibility, 88, 94, 95–96, 101, 110
 - of speech coders, 54
 - of synthetic speech, 127
- intention, 7, 8
- intentional structure, 198
- interactive voice response, 62, 100–131, 155, 287–291
- interaural intensity difference, 32
- interaural phase difference, 32
- International Phonetic Alphabet (IPA), 15–16
- interruption, 6, 9, 106, 109–110, 113, 118, 124, 125, 196, 202
- intonation, 11, 87, 88, 89–90, 94, 196, 199, 201–202, 301
 - in synthetic speech, 85
- isolated word recognition. *See* discrete word recognition
- keyword spotting, 103, 139
- knowledge representation, 189–191
- layers of communication, 6–8
- lexical, 9, 10, 86, 87, 150, 187, 302, 303
- lexicon, 17, 85–91
- Linear Prediction Coefficients, 14
- Linear Predictive Coding (LPC), 37, 52, 59, 61, 66, 79, 92, 93, 132, 135, 147, 149
 - editing, 55
 - Multipulse LPC, 53
 - Residually Excited LPC, (RELPC), 53
- Link Access Protocol (LAP-D), 222
- local loop, 213
- localization, 31–33
- logarithmic quantizer, 45
- manner of articulation, 25
- masking, 34
- meaning, 8
- menu, 108, 110
 - touchtone, 112–113
- microphone, 11, 12, 37, 78, 157, 161, 162, 164, 165, 170, 171, 174, 177, 178, 210, 218, 219, 235, 269, 273, 286
- middle ear, 29
- MIME, 270
- MMConf, 66
- modem, 220, 245,
- Modified Rhyme Test (MRT), 95
- (Modular Integrated Communications Environment) MICE, 232–233, 237–239, 248, 249, 250, 275
- morphology, 86, 94
- MUSE, 68
- mutual beliefs, 199
- names, synthesis of, 95
- nasals, 23, 25, 27, 94
- naturalness of synthetic speech, 96
- neurons, 28, 31
- normalization, 86
- noun phrase, 181, 182, 185–186,
- Nyquist rate, 38
- Office Manager (CMU's OM), 168
- Olivetti, 263, 276
- packet switch, 212
- palatal, 26
- parametric synthesis, 91–93
- parse tree, 17
- parser, 176–177, 180, 185–186, 205
- pattern matching, 133–137, 141, 149, 152
- perception, 11
- performative utterances, 192
- perplexity, 150, 169
- Personal eXchange, (PX) 77–78, 239–242, 249
- Phone Slave, 251–256, 282
- phonemes, 10, 13, 14–17, 24–25, 28, 58, 84–94, 102, 110–111, 133, 142, 146, 151, 152, 161
- Phoneshell, 287–291
- phonetics, 9, 10
- Phonetool, 242–244, 254, 256
- phrase, 17, 302
- pitch, 6, 11, 13, 17, 21, 34, 42, 52, 58, 69, 79–80, 87–94, 220, 300, 301, 302, 303
- Pitchtool, 78–79
- place of articulation, 25, 28
- plan recognition, 190, 191
- planning systems, 191
- pragmatics, 8, 9, 189–194
- privacy, 103–104
- Private Branch eXchange (PBX), 226–228, 237, 239, 242, 246, 248, 262,
- proofreading, 98
- prosody, 87, 94, 301–303
- psychoacoustics, 28, 33–34
- Pulse Code Modulation (PCM), 38, 44
 - editing, 55
- Put That There, 168, 175–178, 207
- Q.921, 222, 223
- Q.931, 222, 223
- Q.932, 222

- quality
 - of speech, 106–107
 - of speech coders, 54, 61
 - of synthetic speech, 96
- Quality factor, Q 23, 31
- quantization, 43–44
- Quilt, 66–67
- recursive transition network (RTN), 183–184, 185–186
- referent, 197
- repetition, 109–110
- representations, 12–14
- rewrite rules, 182, 185
- ringback signal, 215
- robust communication, 58–59, 298–301
- sampling, 12, 37–43
- scripts, 191
- sedition, 78
- selectional restriction, 187, 188
- semantic network, 17, 188–189
- semantics, 8, 10, 17, 177, 186–189, 204
- semivowels, 28
- senses of words, 187, 192
- serial nature of speech, 102
- servers
 - audio, 273–278
 - telephone, 244–251
- signaling and telephone networks, 211–212, 213–218, 240
- signal-to-noise-ratio, 43, 45, 220
- silence removal, 55–57
- Simple Message Desk Interface (SMDI), 248
- sound pressure level (SPL), 33
- SoundViewer, 253, 278, 280, 282–285, 293
- source coders, 37, 51–53
- Spanish, 180
- speaker adaptive recognition, 137–138
- speaker dependent recognition, 137–138, 177
- speaker independent recognition, 137–138, 155
- speaker phone, 219
- spectrogram, 11, 13, 132
- spectrum, 21, 23
- Speech
 - rate, 36, 101, 108, 121, 139, 142, 157–160, 251
 - speech coders, 36–59
 - algorithms, 44–53, 58–59
 - intelligibility, 54
 - speech act theory, 7, 189, 192–194
 - speech recognition, 104, 109, 132–153, 154–178, 252, 299–301
 - software-based, 273
 - speech synthesis, 82–99, 101, 107
 - concatenative, 93–94
 - parametric, 91–93
 - reading electronic mail, 124–130, 288–291
 - software-based, 273
- speed, 101, 106
 - dialing, 232, 233, 251, 257
- Sphinx, 151–152, 165
- stack, discourse focus, 18
- stress, 87–88, 91, 94
- stutter dial tone, 218
- subset of language, 164–165
- SUMMIT, 152
- Switch-Computer Application Interface (SCAI), 248
- syllable, 17, 88, 93
- Synchronous Overlap and Add (SOLA), 58
- syntax, 8, 10, 150, 180–186, 204
- T1.607, 222
- Tangora, 151
- telephone, 104, 210–229
 - remote access, 271
 - speaking number, 83–84
 - speech quality, 44
- telephone-based interfaces, 106, 107
- template, 133, 134–135, 140, 142, 147, 148, 172
- temporal nature of speech, 102
- termination of touch tone input, 116, 120
- threshold, rejection, 135
- Time Assigned Speech Interpolation (TASI), 56
- time compression, 108, 273, 281, 284
 - of recorded speech, 57–58
- Tioga Voice, 75–77
- top-down parser, 185–186,
- touchtones, 100–101, 104, 109, 112–117, 155, 216–217, 223, 246, 252,
- toys, 61
- transition network, 183
- transmission in telephone networks, 212, 240
- tree, 180
- trigram grammar, 151, 159
- trunk, 214–215, 226–227, 248
- Turkish, 84–85
- turn, 6
- type hierarchy, 188–189
- ubiquitous audio, 278–281
- Vector Quantization (VQ), 141, 147–149, 152
- verb phrase, 181, 182, 185–186
- vestibular organs, 29
- vocabulary size, 140
- vocal cords, 21

- vocal tract, 19–24, 23
- voice mail, 36, 53, 54, 62–63, 82, 210, 231, 250, 251–256, 269, 271, 287, 292
- Voiced Mail, 124–130, 269, 289, 290
- voicing, 21, 25, 28
- vowel reduction, 89
- vowels, 15, 25–26
- VOX audio server, 78, 276

- warnings (by speech), 62
- waveform coders, 37, 44–51

- window manager, 171, 172
- written language, 84

- X Window System, 78
- Xcapture, 280–281
- Xerox Palo Alto Research Center, 234
- Xmedia, 276
- Xphone, 244, 256, 256–257, 259
- Xrolo, 256, 258–260, 259
- Xspeak, 157, 170–175