*Cognitive Engineering in the Design of Human-Computer Interaction and Expert Systems,*
edited by G. Salvendy
Elsevier Science Publishers B.V., Amsterdam, 1987 — Printed in The Netherlands


# Conversational Telecommunications Environments

Christopher Schmandt
Media Laboratory, Massachusetts Institute of Technology
20 Ames St.,Cambridge, MA 02139, U.S.A

## Abstract

This paper describes several conversational voice interactive computer systems, each capable of handling a small number of office and telecommunication functions. These systems both utilize *dialog* as an integral component of the user interface, and this dialog is essential to the successful completion of a transaction. The *Phone Slave* is a conversational answering machine utilizing digitized speech. The *Conversational Desktop* uses voice recognition and synthesis as part of an networked office environment.

## 1 Conversation in the speech interface

Conversation as a mode of interaction for human-computer voice interfaces is both natural and necessary. Conversation is intuitive, as it is the technique humans ordinarily use to convey information or make requests. Dialog is rational; it is used as a means for the hearer to request clarification from the talking in the event of misunderstanding or transmission errors. This use will prove essential with speech recognition systems, which are prone to high error rates.

The ability of a computer system to maintain a conversation depends on the quality of its speech input and output, its model of human dialog, and its knowledge about the domain in which it is operating. This paper discusses two systems implemented at the Media Laboratory to explore issues in the design of conversational systems.

Implicit in the design of these two systems is recognition of the trade-off between dialog complexity and acoustic quality. For word recognition devices, the probability of error increases with vocabulary size and sentence length (number of words). For speech output, text-to-speech synthesis offers greater flexibility, but at a steep price in terms of intelligibility as compared to pre-recorded human speech.

Conversation, and voice in general, may be used as a data channel or as a control channel [Wat83]. In some cases, such as a voice storage and forward system, the voice is itself the data, and the act of playing that voice back to a user consumates the transaction or request for information. In other cases, such as accessing an airline reservation system or schedule utility, voice is merely the channel to initiate and control the transaction, which causes some database to be updated appropriately.

# 2 Phone Slave

The *Phone Slave* [SA84,SA85] is an example of a restricted domain, low branching factor conversational system in which voice is itself the data. Because the branching factor in the dialog is so low (at least from the point of view of the computer's role) it is possible to use high quality pre-recorded speech for output. Although it uses no speech recognition in the interactions of interest here, it may appear to understand the talker quite well, due to the constraints in the particular domain of taking a telephone message

The Phone Slave is a conversational answering machine, which takes telephone messages by asking callers a series of questions and recording their responses digitally. Each recording goes into a separate audio file to facilitate later retrieval. Each question/answer transaction is terminated by an adaptive pause detection algorithm which detects when the human caller has finished speaking.

## 2.1 Voice as data

When the telephone rings, the Phone Slave answers it and immediately takes the initiative in the conversation, an initiative which it cannot afford to lose because it has no ability to answer questions and does not use speech recognition to try to understand what the caller is asking. A message is taken by asking a standard series of questions, such as *"Who's calling please?"*, *"What's this in reference to?"*, and *"At what number can you be reached?"*

Each reply by the caller is recorded into a separate audio file to facilitate access by the system's owner. Recorded speech is a difficult medium for perusal or filing because it is slow and sequential in nature; these factors certainly interfere with acceptance of voice storage and forward systems. The Phone Slave does not attempt to understand the content of a caller's response, but rather notes the *context* in which it was recorded to understand something about how the recorded audio might be used, and when to present it to the machine's owner.

For example, the owner may ask *"Who left messages?"*, for which it should suffice to play back serially each of the callers' responses to the machine's query of *"Who's calling please?"*. While playing the third calling party identifying herself, the owner may ask *"What does she want?"*. A suitable response by the machine would be to play the next audio file associated with that message, the caller's response to *"What's this in reference to?"*

## 2.2 User expectation

This interface proved to be surprisingly effective in elliciting the desired responses to its dialog in order to gather messages. The success of this conversational system is due in large part to very strong user expectations, and reinforced by the apparent high quality of the voice interaction.

Phone Slave has a very limited branching capacity in its conversations, triggered mostly

by error conditions (see below). It can succeed only in interactions wherein it can maintain the initiative. Such an approach works successfully in a message taking application because this is an extremely focused discourse domain and users have very strong expectations of what is to happen in such a situation. A caller who phones my office and gets a receptionist instead is not surprised to be asked questions which are rational in the context of answering a telephone, and in fact finds it very difficult not to answer them.

Taking a message is a cooperative behavior; there is every reason for the caller to participate according to conventional roles. In fact, asking a series of questions, as opposed to simply *"Leave your message at the beep..."* makes it easier for the caller to leave a complete message. This maintains the computer's ability to control the conversation; the limited discourse domain protects its fragile "intelligence".

There are no beeps or further prompts in the conversation, nor is there any explanation of the limitations or even existence of the answering machine. This was a deliberate design decision prompted by several considerations:existing preconceptions as to answering machine behavior, and our desire to maintain as "natural" a conversation as possible. Callers are used to conventional answering machines: one hears a beep and has a certain amount (usually unknown) of time to spew out a message. Beeps reinforce the expectation that the caller has reached a machine which operates in this manner, i.e., no conversational ability, which is exactly what the Phone Slave seeks to avoid.

## 2.3 Failure modes

The Phone Slave's conversation can break down for several reasons, and system design has to cope with these failure modes.

One problem is reliable pause detection on possibly noisy telephone lines. The system desires to respond with each new query as quickly as possible; a long pause will be treated by the caller as an invitation to continue speaking, but the owner prefers short and specific replies. On the other hand, it is not helpful to interrupt a slow speaking talker when a short pause is detected.

The pause detection algorithm is therefore adaptive both to phone line noise as well as talker speech rate. Background noise adaption is done by dynamically readjusting the "speech present" audio level threshold during pauses. The rate adaption is done by increasing the pause length detection time constant for those conversations in which pauses of greater than 500 ms seconds are detected before the "speech finished" timeout has been reached.

Another failure mode is a rambling reply by the caller. Except for the final question in the conversation (*"Can I take a longer message for you?"*) it is desireable to record relatively short and specific responses to each question, to improve the owner's access to message contents. Sometimes callers give vague and lengthy replies. This is detected by having a maximum length associated with each response; if the caller exceeds it, the machine interrupts, indicates that it is only an answering machine, and asks the caller to be specific.

Clever arrangement of the dialog may sidestep other failure modes. The most common

of these is for the caller to ask *"Is he there?"* when asked who is calling. The machine's second question, *"What's this in reference too?"* just happens to work reasonably well as a response to this situation as well.

# 3 Conversational Desktop

The *Conversational Desktop* [SS85,SA86] is based on the concept of an integrated office workstation which combines the functions of a powerful personal computer and an intelligent telecommunications system. In addition to conventional personal computer applications, this workstation is actually an active node on a digital network. It handles its owner's schedule, travel plans, telephone management and message taking, and event-activated audio memoranda or reminders.

Each workstation is equipped with a variety of speech peripherals, including recognition, synthesis, and digital record/playback hardware. The workstation is designed to be driven entirely by voice, engaging its owner in a conversation interleaved with transactions with remote nodes. The repertoire of available operations includes: scheduling meetings with individuals or groups, contacting remote databases of airline schedules and automobile commuter traffic, placing outgoing calls, taking incoming voice messages, and recording voice memos related to the above activities.

## 3.1 Mixed initiative conversations

This system tries to take a more interactive role in conversations, allowing for a mixed-initiative [BKK*86] interaction in which the human starts a transaction and the computer then builds up a series of sub-tasks to try to understand what the human wanted. The sub-tasks are completed by having a conversation with the talker, and asking a series of questions. Because the questions incorporate various parts of the discourse (see below) the branching factor in the conversation far exceeds a reasonable number of sentences for pre-recorded storage, necessitating the use of speech synthesis for output.

The source of gaps in the computer's understanding of the talker's request may come from an incomplete input utterance or from errors in speech recognition. As this system employs *connected* speech recognition, a number of different classes of error are possible, and at least some errors are likely to occur in nearly every interaction. This makes it necessary to parse the input utterance, not only to discover the errors, but also to try to extract whatever useful information may be present. The output of the parser is sent to a dialog generation module which builds a sentence to be synthesized and spoken to the user.

## 3.2 Robust parsing of error prone input

A parser is used to analyze speech input and detect errors; this analysis is based on a formal description of the syntax of the set of input utterances. The parser also generates a

description of the input, in a frame-like [SA77] representation which is convenient for both the dialog generator as well as action routines embedded in the application itself. What is unusual about this parser is that, in addition to the usual syntax rules, it is designed to detect input errors and parse the any remaining correct sentence fragments.

Connected speech recognition errors can be classified into three categories: substitution, rejection, and insertion. A *substitution* error is one in which some number of words are spoken and the same number are recognized, but one or more of them is recognized incorrectly. A *rejection* error is one in which less words are recognized than were spoken, i.e. one or more input words were simply not recognized. An *insertion* error is one in which more words are reported than were spoken, perhaps because one input word was recognized as several words or perhaps breath noise was matched against a word.

Most conventional parsers [Win83] cannot cope with any of these problems, because they assume well formed input. Rather than detection of errors, their task is to correctly determine the syntactic relationships of the input tokens. This is inadequate for voice input.

Previous *speech* parsers [Lev78] successfully dealt with substitution errors, by considering a number of possible choices for each word, and choosing the most probable path through these choices based on syntax information. Since Levinson's parser dealt with discrete speech in which each word has to be spoken separately, the parser assumed that the number of input tokens was correct and would fail if an insertion or rejection error occurred.

To cope with *insertion* errors, the parser considers all *subsets* of the tokens returned by the recognizer. Thus if a spurious token is inserted in the recognizer's output, the complete string of tokens will probably fail the parse. Rather than simply reject it, the parser examines the substrings of the input, one or more of which will be syntactically acceptable.

To cope with *rejection* errors, the grammar accepts syntactically correct *sentence fragments* as well as complete sentences. This retains information about what was correctly recognized even though it may be incomplete. If a single word is rejected, for example, the rest of the input may be correct and should be accepted by the parser to trigger a dialog.

Having tested the substrings of the input tokens for syntax against the grammer, the parser must select the best from among those which are acceptable. This is accomplished by applying three simple but surprisingly effective socring rules:

- Completion: a complete sentence is preferred to a fragment, as one is more likely to speak a complete command to the machine.
- Number: of two possible substrings, the one with the larger number of tokens will be selected.
- Adjency: additional weight is given to adjacent tokens. For example, if the original input was *ABCD*, the substring *ABC-* has a higher adjacency score than *AB-D*.

Adjacency is a powerful metric specifically for connected speech, because a significant portion of the problem of connected recognition is *segmentation*, finding word boundaries.

If it is postulated that the second token in an utterance is correct, it is more likely that the first and third tokens will also be correct because at least one of each of their boundaries must have been determined correctly [RL81,Zue85].

## 3.3 Dialog generator

The parser outputs both the frame, with slots filled by specific instances from the vocabulary, and a simpler structure which indicates what information is missing for this particular parse path. Because discrete speech is easier to recognize than connected speech, the dialog generator initiates a series of questions, each designed to elicit a single word response. The dialog generator also employs an *indirect-echoing* technique [HR83] to allow implicit of what the system thinks it has understood so far; each question is phrased so as to echo as much as is assumed to be correct in the utterance.

For example, if the user said *"Schedule a meeting with Chris Friday afternoon"* and the recognizer reported *"Schedule a meeting ... Friday."* the first question generated would be *"With whom do you wish to meet on Friday?"* The query is generated from the frame information as a text string, and sent to the speech synthesizer to be spoken.

The dialog generator can also be used to generate queries that are not directly related to completing a user's command. After a user's request is completed, an *incomplete* set of tokens can be programmatically passed to the parser, and hence to the dialog generator. This will cause a *new* question to be generated, initiating further dialog.

For example, the user might initiate an interaction with *"Schedule a flight to Chicago Friday morning."* Note that the machine tracks the user's whereabouts, so it is not necessary to give the city from which you are leaving. The computer would first confirm this request, perform the appropriate action, and enter the event into its calendar database. The command *"Schedule a return flight from Chicago"* would then be passed to the parser, initiating the query *"When would you like to return from Chicago?"*, as flight scheduling commands require a place and a time for completion. In effect, incomplete user input is simulated to cause the proper prompt to be generated automatically by the dialog generator.

## 3.4 Modeling user attention

The Conversational Desktop can respond to external, asynchronous events (an incoming telephone call, a timer going off), and that response may include an audio component (playing a pre-recorded reminder, alerting the owner of the incoming call, or recording a phone message by using the Phone Slave). What behavior is most appropriate in response to these events is a function of *user attention*, which is of course a difficult input to capture.

To facilitate this, the Desktop assigns spatial orientation to the system; the computer is assigned the direction of the owner's right, and the telephone the left. The system display, which shows calendar entries and phone message status, along with the loud speakers through which the Desktop talks, are both situated on the right side of the office. The "telephone" is a hands-free arrangement using the head mounted microphone for input and

a speaker for output to the left side of the office.

A pair of microphones placed behind the user determines the direction towards which the person is speaking. The microphone receiving the *minimum* signal when speech is detected in the head-mounted (recognizer) mike is in the opposite direction of the voice addressing. Microphones were place to the rear to take advantage of the greater direction sensitivity based on the radiational characteristics of the human head [Fla60].

Recognition output is parsed only when speech is being transmitted in the direction of the recognizer. While speaking on the phone, the owner may have a private conversation with his Desktop by turning to the right; as soon as speech is detected in this direction, audio input to the telephone connection is temporarily disabled.

The direction-sensing microphones are also used to detect background noise (defined as signal present with no speech on the owner's microphone) which alerts the system to the presence of other humans in the office. The "background speech present" signal is used for a class of operations characterized by knowledge of the acoustical context of events occurring in the domain of the Desktop system. When it is time to play an audio reminder, for example, the system first checks this signal and can postpone the reminder until a time when the owner is alone in his office.

# 4 Conclusion

Two examples of conversational voice interfaces have been presented. Although they share the theme of dialog as a mode of interaction, they differ in their use of voice and conversational ability. This differences include: voice as data vs. voice as control, small branching factor with high quality voice interaction vs. large branching factor with recognition errors and synthesized output, and different error detection and recovery proceedures.

# 5 Acknowledgment

This work was supported by NTT, the Nippon Telegraph and Telephone Public Corporation.

The author wishes to credit and thank Barry Arons, currently at Hewlitt-Packard Laboratory, for his involvement in the design and implementation of both these projects. Without his participation they would not have happened.

# References

[BKK*86] D.G. Brobow, R.M. Kaplan, D.A. Kay, M. Norman, H. Thompson, and T. Winograd. *GUS, a frame-driven dialog system*, pages 595–604. Morgan Kaufman, 1986.

J. L. Flanagan. Analog measurements of sound radiation from the mouth. *J. Acoust. Soc. Am.*, 32(12), 1960.

P.J. Hayes and R. Reddy. Steps towards graceful interaction in spoken and written man-machine communication. *Int J. Man-Machine Studies*, 19:231::284, 1983.

S.E. Levinson. The effects of syntax analysis on word recognition accuracy. *Bell System Technical Journal*, 57(5):1627–1644, 1978.

L. Rabiner and E. Levinson. Isolated and connected word recognition – theory and selected applications. *IEEE Transactions on Communications*, 25(5):621–659, 1981.

R.C. Schank and R.P. Ableson. *Scripts, Plans, Goals, and Understanding.* Lawrence Erlbaum Press, 1977.

C. Schmandt and B. Arons. A conversational telephone messaging system. *IEEE Trans. on Consumer Electr.*, CE-30(3):xxi–xxiv, 1984.

C. Schmandt and B. Arons. Phone slave: a graphical telecommunications interface. *Proc. of the Soc. for Information Display*, 26(1):79–82, 1985.

C. Schmandt and B. Arons. A robust parser and dialog generator for a conversational office system. In *Proceedings*, American Voice Input Output Society, 1986.

B. Schmandt, C. Arons and C. Simmons. Voice interaction in an integrated office and telecommunications environment. In *Proceedings*, American Voice Input Output Society, 1985.

J.A. Waterworth. Man-machine speech 'dialogue acts'. *Br. Telecom Technol. J.*, 1(1), 1983.

T. Winograd. *Language as a Cognitive Process - Syntax.* Addison-Wesley, 1983.

V.W. Zue. The use of speech knowledge in automatic speech recognition. *Proceedings of the IEEE*, 73(11):1602–1615, 1985.

# CONTENTS

Advances in Human Factors/Ergonomics, 10B

# Cognitive Engineering in the Design of Human-Computer Interaction and Expert Systems

Proceedings of the Second International Conference on Human–Computer Interaction, Honolulu, Hawaii, August 10–14, 1987, Volume II

Edited by

**Gavriel Salvendy**
Purdue University, West Lafayette, IN 47907, U.S.A.

ELSEVIER

Amsterdam – Oxford – New York – Tokyo 1987

ELSEVIER SCIENCE PUBLISHERS B.V.
Sara Burgerhartstraat 25
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

*Distributors for the United States and Canada:*

ELSEVIER SCIENCE PUBLISHING COMPANY INC.
52, Vanderbilt Avenue
New York, NY 10017, U.S.A.

ISBN 0-444-42847-X (Vol. 10A)
ISBN 0-444-42848-8 (Vol. 10B)
ISBN 0-444-42849-6 (Set)
ISBN 0-444-42396-6 (Series)



Printed in The Netherlands