From: Marics, M.A., & Engelbeck, G.E. (1997). Designing voice menu applications for telephones. In M. Helander, T.K. Laundauer, and P.V. Prabhu (Eds). Handbook of Human-Computer Interaction.2Nd Ed., Elsevier Science Ltd

Designing Voice Menu Applications for Telephones

1. Introduction

Millions of people use Phone-Based Interfaces (PBIs) every day. Because of its vast coverage, the touchtone telephone has become the primary terminal used by many software applications. Example applications include: automated banking, data base querying, order entry, remote self-help, directory assistance, audiotext, automated call routing, voice mail, and call screening. PBIs are popular primarily because they are readily available to many users. In 1993, over 70.5 percent of Americans owned a touchtone telephone (Yankee Group, 1994).

1.1. Purpose

This chapter is intended as a "how to" guide for designing PBIs. We concentrate on how to create usable applications rather than explaining how and why certain designs work. Our experiences include developing PBIs, doing research on PBIs, and participating in domestic and international standard bodies. Both of us have practiced at regional telephone companies within the United States. Although we have little experience developing PBIs for markets outside the United States, we believe much of our design experience is applicable to international settings.

1.2. Scope of this Chapter

Our focus is narrow. We discuss how to design voice menu applications for mass market consumers. These PBIs use the telephone's keypad and mouthpiece as an input devices and the telephone's earpiece as the output device. Mass market products can be divided into single use and repeated use applications. Single use applications include those where users unexpectedly encounter the application, or use the application infrequently. In repeated use applications, users gain enough experience with the application to begin memorizing menu choices and interrupting prompts. Design guidelines for both application types are covered.

Traditionally, application user interface development consists of five areas: requirements, functional specification, design, development, and test. This chapter discusses issues mainly in the design phase–interaction components, guidelines and standards for voice menu applications. We assume that the application has

been appropriately scoped, users have been identified and profiled, and a user task analysis has been performed.

For other voice menu design guideline documents, refer to the Voice Messaging User Interface Forum (Information Industry Association, 1990), User Interface to Telephone-based Services-Voice Messaging Applications (ISO, 1995), User Friendly Recommendations for Voice Services Designers (France Telecom, 1991), U S WEST Touch Tone Standards for Voice Prompted User Interfaces (Bain, 1990), Ameritech Phone-based User Interface Standards and Design Guidelines (Schwartz, Hardzinski, 1993), and Designing Phone-based Interfaces (Halstead-Nussloch, DiAngelo, Thomas, 1989). We do not discuss how to design for telephone interfaces using speech recognition (see Dobroth, Karis, Ziegler, 1990; Waterworth, 1982), or designs for screen-based telephony (see Davies, 1995).

1.3. PBI User Interfaces

PBIs are appropriate when:

- potential users have access to a touchtone telephone,
- information needs to be accessible from many locations,
- users' tasks are constrained and goal directed,
- interactions are relatively short, and
- modest amounts of information are transferred.

Human conversational speech averages from 175 to 225 words per minute, which translates to approximately 300 baud (Schmandt, 1994). In addition, audio output is transitory so users have to remember more information than they would with a comparable visual display. Poorly designed PBIs can tax the working memories of their users to the point where users cannot successfully use the application (Kidd, 1982).

There are three primary styles for PBIs: voice menus, command driven, and skip and scan.

Voice Menu

Voice menus are the most frequently used interface style for phone-based applications. Applications using this style typically present a title, ask a question, or present a menu of instructions to users. Users choose the action they want by pressing the action's corresponding key.

The voice menu style is frequently chosen for phone-based applications because it is easy to use and requires little training. These characteristics are important since many phone-based applications are intended for use by populations that do

not receive training. The success of these applications depends on users being able to dial up "cold" and use them effectively.

Command Driven

Command driven interfaces have key sequences (e.g., *72 or *h) that correspond to application operations. Example applications of this type include Call Forwarding and Call Trace. Compared to voice menu interfaces, command driven interfaces tend to be difficult to learn and difficult to remember how to use. One disadvantage is that users must remember, rather than recognize, command names and entry syntax. This situation is exacerbated by the 12-button keypad which leads to cryptic command strings. However, they may be more efficient for applications where users are willing to take the time and effort needed to become proficient.

Skip and Scan

Resnick and Virzi (1992) describe a PBI interaction style called skip and scan. A skip and scan application maps navigation commands to the keypad. During playback of information, pressing 7 takes users to the previous item, pressing 9 takes users to the next item, and 1 selects the item. Skip and scan is good for information retrieval applications where long audio files can be easily skipped. With a skip and scan application, users can skip ahead to information of interest without listening to all of a current message.

However, the advantages of skip and scan are also its limitations. There is an initial time cost while users learn the navigational commands. Skip and scan does not allow experienced users to choose an item directly. Additionally, as Resnick and Virzi report (p. 424), the voice menu style is becoming a *de facto* standard. Skip and scan is seen by users as being a non-standard way to interact with a phone-based application.

1.4. Structure of this Chapter

Section 2 discusses the structure of applications and voice menus. Section 3 discusses prompts and prompt recording. Section 4 discusses data input and common data elements. Section 5 discusses general application issues such as timeouts, prompt interrupt and error recovery. Finally, Section 6 contains a summary of guidelines and recommendations.

2. Menu Structure

Voice menu applications present prompts, menus and messages to users. Specifically, a *prompt* is an application request for input–for example, "Please enter your password." A *menu* is a collection of prompts. For example, "To erase, press 1. To send, press 2. To exit, press star," is a menu consisting of three prompts. There are also application *messages* which merely convey information without requesting action–for example, "Loan amounts are updated on the first of each month."

Schmandt (1994) wrote, "The awareness that speech takes time (and acceptance that time is a commodity of which we never have enough) should permeate the design process for building any speech application." This is the essence of a successful voice menu application design. To make users' interaction with the application most efficient, the auditory information should short and concise. Encourage users to interrupt menus by making selections. This highlights the importance of the wording and structure of the menus.

2.1. General Structure

Voice menu applications take the form of a branching tree structure. Typically, there is a Main Menu which users hear upon entering the application. The options on the Main Menu lead to sub-menus containing more choices, or to task paths. The standard factors concerning the depth versus breadth of menus apply (see Lee and MacGregor, 1985). Users progress forward and backwards through the tree by making selections. At each node in the tree structure, context sensitive help may be provided. A global command to back-up through or exit the tree structure may also be provided.

2.2. Titles

The Main Menu and all other important menus should have titles. It is easy for users to become lost in an interactive telephone application. Given the auditory nature of voice menus, users can forget where they are within the application, and how to return to a previous point in the application's flow. To help users understand the structure and where they are within the application, important menus should be titled. To be effective sign-posts, titles should be task related.

2.3. Items per Menu

The majority opinion is that four items per menu is about right. If more than four menu items are offered, users can forget which items were offered, and which one they wanted (Engelbeck and Roberts, 1990). Although menus should be limited to four items, global commands such as Help and Exit need not be included in this count. Also, menus may have additional items that are not stated in the menu for expert users.

2.4. Order of Items

Menu items should be ordered according to frequency of use, natural order, functionality or consistency.

Frequency of use

Frequency of use is the preferred way to order menus. The most frequent menu choice should be the first item on a menu. If frequent choices are listed first, users do not have to listen to and remember the entire menu. This saves users' time and application connect time. Users will feel that the application is well designed because they do not have to wade through lengthy prompts to find the item of interest.

The frequency of use guideline can be overridden if the menu choices have a natural or functional order. If the same menu is offered in several places, consistency of menu choices should take precedence over frequency of use.

Natural order

Some menu choices have a natural order. For example, you have to *add* things to a list before you can *delete* them. Preserving the natural order of items can help users remember the choices.

Functionality

Related menu items should be mentioned together to help users keep track of the choices. For example, suppose you have the following menu items for a telephone banking application: savings account information, pay bills, transfer funds, and checking account information. You might place the savings and checking account information items next to one another in the menu since they are functionally similar, although user testing should confirm such design decisions.

Consistency

If the same menu choices are offered several times in an application, then those choices should be ordered consistently, and consistently assigned to the same key. Confirmation of data should be placed on the 1 key, and negation on the 2 key consistently throughout the entire application. For example, "If you own a car, press 1. If you don't, press 2." In many cases, you can even rephrase the question so that confirmation is the most frequent answer in addition to being consistently offered first. Consistency between menus can help users quickly learn an application's structure.

2.5. Numbering Menus

Menu choices should be numbered consecutively starting at 1. Users expect that the first menu choice they hear will be on the 1 key. Likewise, they expect the second choice to be on the 2 key and the third choice to be on the 3 key. Numbering choices consecutively helps users keep track of the choices. It also allows them to guess which key is next once they hear the beginning of a prompt.

Avoid skipping numbers or presenting numbers out of order. In some cases, this may be difficult. For example, some users may not have access to all the features, or you may be reserving a number in a menu for a future feature.

Menu items should be stated before the key corresponding to the action, e.g. "To do action, press 1". With this wording, if users are not interested in the action, they do not have to actively remember the associated key. Authors such as Halstead-Nussloch (1989) and Engelbeck and Roberts (1989) recommend an action-key ordering.

2.6. Mnemonics

Mnemonics should not be used to specify menu choices. For example, it may seem natural to specify, "For yes, press y. For no, press n." However, there are several reasons why mnemonics are not desirable:

- It takes longer for users to locate a letter on the telephone, such as "y," rather than a number, such as "9." (Try dialing "1-800-Rentals" rather than "1-800-736-8257.")
- There are often several terms for the same command. For example, "erase," "delete" and "remove" do approximately the same thing. Users have to remember which form of the command is being used in order to remember which letter to press.
- Not all phones have letters on the keypad.
- The position of Q and Z is not consistent across telephones, if they even appear at all.
- If commands begin with different letters and those letters are on the same telephone key, there is a conflict. If commands begin with the same letter, there is also conflict. For example, P, R, and S all appear on the 7 key. In voice messaging, which command would you put on the 7 key: Pause, Play, Rewind, Reply, or Repeat?
- Mnemonics won't work if the application is expanded to international markets. Not only are words spelled differently, but languages which do not use the Roman alphabet, such as Russian, Japanese, Chinese and Arabic, will not be able to use the interface.

2.7. Active Menu Options

If a menu option is currently not available to users, don't speak that option in the menu. For example, users should not be prompted to "delete" unless something has already been "created". Likewise, don't prompt users to turn a feature off if the feature is already off. (Note: this may create a hole in the order of menu items.)

2.8. Global Commands

The keys 0, * and #, are typically mapped to the global commands help, cancel, and delimit input. Global commands should always perform the same action throughout an interface (with the exception of 0 in data entry). In addition, the keys should be active in all parts of the application. If the keys only work in some parts of the application, users may be unable to descriminate when the keys are active, and when they are inactive. Since these keys perform the same functions throughout an application, they need not be counted as menu items.

2.9. Vocabulary

The wording of the menu choice should clearly represent the functionality accessed by that choice, and should be used consistently across the application. Words mean different things to different users. Succinctness (e.g., "Turn on, press 1.") and clarity (e.g., "To turn on toll blocking which disallows all outgoing toll calls, press 1.") should be balanced (e.g., "To block toll calls, press 1.") If the words are ambiguous, even users who know what they want to do will be confused about which menu option to select.

We have found that the verb *enter* works well when asking users to enter a sequence of touchtone keystrokes; *dial* works well when specifically asking users to enter a telephone number; *press* works well for single keystrokes, and *speak* works well for asking users to record or input speech. (The verb *dial* also works well for interfaces that accept both touchtone and rotary input, although this should be balanced against the proportion of expected touchtone and rotary users.) In phrasing menu options, use *for* when referring to an object, and *to* when referring to an action. Examples include "For schedules, press 1. To change your itinerary, press 2." It is not necessary to add the word *key* or *button* after each menu item (e.g., "To listen, press the 1 key."). From context, users understand that 1 refers to the keypad. In addition, it lengthens the prompt and sounds repetititve across several menu items.

If users are sure about which menu item maps to their goal, they do not have to remember other menu items, and can choose that item immediately after it is spoken. Decreased ambiguity can allow the presentation of a greater number of menu items without degrading performance. When users are unsure of which menu item they want, they may listen to the entire menu and try to remember every menu choice offered.

Additionally, users may have to listen to the menu several times before choosing a menu item. This greatly decreases users' satisfaction with the application's interface.

3. Application Prompts and Recordings

Recording prompts and messages can be a very costly and time intensive process. Selecting a voice for the application is important. In addition, for each individual recording attention must be paid to the speech rate, inflection, intonation, volume, and phrasing of the voice.

3.1. Talent Selection

Voice preference is very subjective. We often get user input on different voices prior to making a talent decision. If the application is one of a family of applications, then making a mistake with the original voice can have long lasting ramifications. Users are sentsitive to voice changes once an application is in the field. When we changed the voice talent for a voice messaging product, we received numerous user comments like "is she sick" or "was she fired." Keep in mind that the voice is the major affective element of your application, and becomes a trademark of your service.

It is well worth the expense to hire a professional voice talent to record application prompts. The voice talent will need to understand the application in order to record prompts with the correct intonation. If a prototype of the application is available, letting the talent go through the prototype will help him or her get a feel for the interface. If larger messages are created from smaller pieces, the talent must be able to inflect each phrase so that it sounds natural when strung together. A trained voice and ear are invaluable for this process.

When selecting an application voice, the specific characteristics of the voice seem to be far more important than the gender of the voice. In our experience, users prefer a good voice over an unpleasant voice regardless of the gender. Traditionally, female voices have been used for instructions and information in the telephone network. Empirical data (Cox and Cooper, 1981) show that male and female voices are both appropriate depending on the degree of "agreeableness" and "assertiveness" in the specific voice.

Use only one voice for your application's prompts, menus and messages. Using different voices disrupts the application's flow. Introducing a new voice focuses attention on the voice, rather than on what is being said. There is one exception to this rule. A different voice may be used to differentiate or emphasize an example from the rest of the prompts. For example, if you offer a sample greeting

to voice messaging users, the sample greeting might be recorded in a voice different from the application's voice.

Synthesized speech (computer-generated, rule-based speech) is useful for speaking information that continually changes, or can't be predicted. For example, synthesized speech might be used to read out inventory information to delivery people, or to pronounce customer names and addresses. This type of information is so varied that it would be nearly impossible to pre-record. However, the unnaturalness of synthesized speech limits it applicability. Because the pronunciation software does not understand the meaning of what it is saying, it cannot provide natural phrasing, correct word accents, inflection or emotion. In addition, synthesized speech has lower intelligibility and imparts a higher cognitive load on the listener (see Luce, Feustel and Pisoni, 1983; or Chapter x of this volume).

3.2. Voice Characteristics

The voice for an application should be selected to match the context and purpose of the interactive application. (Marics, 1989; Rosson and Cecala, 1986) While the words in a prompt give users information, the tone and cadence of the speech prompts give users affect and atmosphere. Different voices have different affect. For example, Oksenberg and Cannell (1988) review how the vocal characteristics of telephone survey administrators significantly affected their customer response rate. A regional accent may be appropriate for a local calendar of events, but not for a world news broadcast. An excited tone is appropriate for lottery or sports results, but not when calling in for a bank balance. The right voice makes a large difference in users' perception of the application.

3.3. Prompt Files

Human voices change from day to day and hour to hour. Record all the prompts for an application in a single session if at all possible. Prompts recorded at different times will have obvious differences in volume, tone, pitch, and inflection. After recording, play back the prompt files over a speakerphone to see whether any of the prompts are misinterpreted by the application as touchtone input. This phenomenon, commonly called "talk off," occurs more frequently with female voices. If any of the prompts are "talking-off" the application, those prompts will need to be re-recorded, filetered, or a new voice talent selected.

While it is tempting to reuse prompt files within different parts of an application, this is a risky approach. If you later decide to change the wording in one area of the application, you will need to cross check all the other instances where that prompt is used before understanding the effects of the desired change on the ap-

plication. The complexity of reusing prompts between applications is even greater since it is difficult to track how every prompt is used in each application.

Some applications will attempt to save memory by recording prompt segments and concatenating them into longer prompts as needed. Although a necessity for prompts with variable information such as times, dates, and telephone numbers, this approach makes the prompts sound choppy and artificial. Playback time is usually increased since the application has to locate and play multiple files rather than a single file. Recording is more difficult since each fragment requires intonation suitable for several contexts rather than a single context. Finally, maintenance of the prompts is also more difficult since fragments are reused in multiple instances. In general, this approach is not recommended.

3.4. Phrasing

Rate of Speech

In many applications, prompts are spoken at a slow, constant rate so that users can catch every word. Unfortunately, this also has the effect of making the prompts tedious and boring. We recommend a fairly rapid speaking rate for prompts. DeGroot and Schwab (1993) report that time compression of prompts did not degrade task performance or users' ratings of the application. However, the time compression did not lead to shorter task times due to a slower response to menus. Mulligan, Whitten and Tsao (1988) found that compression of natural speech up to 275 words per minute had no effect on retention, however the speech was rated less favorably by participants. It is our belief that prompts spoken in an energetic and brisk manner contribute to users' perception that the application is quick paced, even if the overall task duration times do not bear this out.

Intonation

The intonation of prompts and prompt phrases is critical to perceived application quality. Users seem to be particularly sensitive to the intonation used when recording error messages. For example, in two different applications the authors have worked on, users have complained that the application scolds them for not taking the correct action. Re-recording the same prompt with a softer intonation ended the complaints.

Intonation is also important when combining prompt segments into a longer prompt. For example, when recording digits which will be combined and played back as a local telephone number, it will sound most natural if each digit is recorded three times using three different intonations (France Telecom, 1991). A neutral intonation is used for digits when they occur within a sequence. A "mid

tonal" intonation is used for digits prior to a pause in the digit string, and a "final descending" intonation is used for digits at the end of the string. If a single intonation is used for all positions, the telephone number playback sounds awkward and seems to end suddenly since there is no lowering of the tone for the final digit. Intonation is also important for other prompt strings such as times, dates, spelled letters, or monetary amounts.

Silence and Pauses

Pauses during speech can convey several meanings to users. For example, a pause might indicate a change of topic, or that the information to follow is very important. Pauses are also used to group related pieces of information. In the example below, the pauses are inappropriately placed. Users might become confused as to whether "press 2" refers to "Ford" or to "Chrysler."

Incorrect:	"For General Motors.....press 1. For Ford.....press 2. For Chrysler.....press 3. To exit.....press star."
Correct:	"For General Motors, press 1. .....For Ford, press 2.....For Chrysler, press 3. .....To exit, press star."

3.5. Content

Repetitiveness

Early phone-based interfaces made liberal use of "please" and "thank you" in the application prompts. However, over time users have found continual (and artificial, since it is a computer) politeness annoying. While politeness is useful to soften the tone of some prompts, such as error and help messages, use it sparingly to avoid annoying users.

Similarly, early applications appended the word "now" at the end of each menu phrase (e.g., "For checking, press 1 now. For savings, press 2 now. For money market, press 3 now...") Stuart, Desurvier and Dews (1991) recommend against this practice for two reasons. First, it implies that users should wait for the prompt to finish before entering the appropriate keystroke. Second, it implies that the only time users can enter the keystroke is after the prompt. (Use of "now" may be appropriate if the application can, in fact, only accept input after a prompt. Also, see Section 5.5 for information on uninterruptible prompts.)

Sometimes "thank you" can be used to acknowledge user input. However, removing excessive repetition from prompts allows users to complete their tasks more quickly and with less annoyance.

Brevity

Prompts should be as brief as possible while still conveying the intended meaning to users. Although some applications allow users to toggle between verbose "novice" prompts and succinct "expert" prompts, we do not recommend this practive. First, it doubles the amount of prompts to write, record and maintain. Second, if prompts are concisely phrased to begin with, there is little gain in adding more verbiage. Remember that users hear the prompts in the context of the larger application. This contextual knowledge can be used to eliminate redundancy in the prompts. Samples are shown below.

Table 1 Prompt Brevity

Original	Revised
"For a listing of all hotels within your local area, press 1. For a listing of all vacation cottages in the area, press 2. For listings of rental homes and condos in the area, press 3."	"For local hotels, press 1. For cottages, press 2. For rental homes and condos, press 3."
"If you would like to add a name to the list, press 1. To remove a name from the list, press 2. To hear all the names on the list, press 3."	"To add a name, press 1. To remove a name, press 2. To hear the list, press 3."
"You have entered 555-1234. If this telephone number is correct, press 1. To change it to a different telephone number, press 2."	"555-1234. If this is correct, press 1. To change it, press 2."

While making prompts concise, care should be taken not to be abrupt. Recording these prompts with a softened intonation is important.

Vocabulary

Prompts for interactive applications should use language common to users. State things in the simplest way possible. Make sure users understand terms that are specific to your application. For example, users might not know their "taxpayers identification number," but they do know their "social security number."

Conversational Style

An application should not refer to itself using a pronoun, nor over-naturalize the interaction to the point of pretending to be a person. For example, users of a particular voice mail application disliked prompts where the application pretended to apologize (e.g., "Sorry you are having trouble.") and prompts where the application pretended to converse with users (e.g., "Are you still there?"). While error messages are necessary, these prompts offer no information or assistance to users, they simply waste time.

4. Data Elements

This section describes user interface practices for the input and manipulation of common data elements such as times, dates, and alphabetic characters.

4.1. Sequence of Data Entry

Data entry usually includes the following sequence: prompt for input, receive input ending with a delimiter, and confirm input.

Prompting for Input

Tell users how to enter their input, and what format their input should take. With applications that mix voice recording and key input, users might need to be reminded when to use the keypad and when to speak. For example, "Using the telephone keypad, enter ...," "After the tone, please speak your ..." Whenever the mode of input changes, remind users of the change. Be as specific as possible when asking users for input. This will reduce input errors and make the application more friendly to users. For example, "Enter your *four digit* password," "Enter the *hour and minutes* you want your message delivered," "Enter your *area code and telephone number*."

Input Delimiters

The purpose of an input delimiter is to let the application know users are finished with their input. For fixed length input strings, delimiters are not necessary. For variable length strings, the # key or a timeout are typical delimiters.

There are several tradeoffs to consider between requiring users to enter a specific keystroke delimiter, such as the # key, and using a timeout to delimit the data entry. Studies have shown (Aucella and Ehrlich, 1986; Davis, 1988; Halstead-Nussloch, Logan, Campbell, and Roberts, 1989; Marics, 1990; Stuart, Desurvier, Dews, 1991) that users often forget to enter keystroke delimiters. In addition, sometimes it takes as long to prompt for the delimiting keystroke, as it takes to merely timeout (Stuart, et. al., 1991). Using a timeout to delimit input requires the same amount of time for users who do not interrupt prompts, but is much easier since users do not have to remember anything or take any additional steps.

In applications designed for repeat users, delimiting keystrokes are desirable since they allow users to avoid waiting for the timeouts to expire. Even when delimiting keystrokes are accepted in an application, if users forget to enter the delimiter, a timeout should also delimit the data entry.

Our recommendation is as follows. If # is entered after a menu choice, fixed, or variable length data string, always accept it as a delimiter. If # is not entered by

users, assume the data is delimited after an application timeout. (For timeout length, see Section 5.1.) Applications intended for repeat users should prompt for the # key. This prompt helps users learn the keystroke delimiter, and saves them time since repeat users can be expected to interrupt prompts as their familiarity with the application grows. Applications intended for one-time or infrequent users should not prompt for the # key (but accept it if entered), and use a timeout to delimit data entry. This reduces the complexity of the prompts for infrequent users and does not increase the time penalty of application use.

Confirmation

During the confirmation step, the application should repeat the data input and ask users to confirm the information (e.g., "June 22 at 2:00 pm. If this is ok, press 1. To change it, press 2. To cancel and exit, press *.") The confirmation step tells users what the application recognized, and gives them a chance to change or cancel the entry.

When the application confirms data input, the data should be spoken using common language. If users enter a product code or account number, avoid reading back the number entered. Instead, give the product name (e.g., blue cotton sweater) or the account title (e.g., checking account). If users enter a dollar amount, time or date, speak the information using its natural form. Examples are shown in Table 2.

Table 2. Confirming Input

Original	Revised
"One two dollars and zero six cents."	"Twelve dollars and six cents."
"Eight two three am."	"Eight twenty-three am."
"March two three, one nine nine nine."	"March twenty-third, nineteen ninety-nine."

4.2. Time

A standard sequence for time entry is: request time, request if am or pm , confirm entry. If only 24 hour time is accepted, the request for am/pm can be eliminated. An example is shown below.

Service: "Enter the hour and minutes you want the message delivered."

User: 8 3 0

Service: "For am, press 1. For pm, press 2."

User: 2

Service: "Eight-thirty pm. If this is correct, press 1. If not, press 2."

An application should accept time entry of 1, 2, 3 or 4 digits. If the entry is one or two keystrokes, assume that the user has entered an hour. If the entry is three or four keystrokes, assume that the last two digits entered are the minutes and the initial digit(s) are the hour. Do not require leading zeros.

After the time has been entered, ask if it is am or pm. If the time entered was 12:00, rather than asking for am/pm ask if the time was noon or midnight since these terms are more commonly used.

The application should accept 24 hour time, but not require it unless it is commonly used in the environment of the application (e.g., non-U.S. or military). If the application can determine that an entry is in 24 hour time, the am/pm step can be eliminated. However, you should still ask users to confirm the time entered.

4.3. Days of the Week

Days of the week can be entered as a digit between 1 (Monday) and 7 (Sunday). Assume that all times are in the future. For example, if today is Friday and a user specifies a 2 for Tuesday, assume that the user means next Tuesday rather then the one that has already past. As shown in the example below, users will need to be prompted for this convention.

Service: "Days are numbered from 1 to 7 starting with Monday as 1. Please enter the day."

4.4. Dates

Dates should be entered in two steps–enter month, and enter day. The order of the steps is dependent upon local convention. Do not require a leading zero. If necessary, ask for the year. Finally, confirm the entry. An example of date entry is shown below.

Service: "Please enter a number from 1 to 12 for the month."

User: 3

Service: "Enter the day."

User: 21

Service: "Enter the year."

User: 98

Service: "March twenty-first, ninteen ninty eight. If this is correct, press 1....."

4.5. Dollar Amounts

The format for dollar amounts depends on whether users need to enter whole dollars or both dollars and cents. If only whole dollars are needed, ask users to enter "the whole dollar amount." If dollars and cents are required, either break the request into two separate parts, prompt users to enter the decimal point, or have the application insert the decimal point automatically.

Prior experience (Goodwin, 1988) has shown that if the latter is implemented, the prompt should explicitly state that users have to include cents, and that the decimal point will be included automatically. The application studied by Goodwin assumed that users were automated bank teller machine users. Asking for dollars and cents with a single prompt was chosen, in part, because it was similar to the format required by most automated bank teller machines. This style of entering dollar amount may not be the best style for other domains. An example is shown below.

Service:	"Enter the dollars and cents, a decimal point will be inserted automatically."
User:	3 7 5 2
Service:	"Thirty-seven dollars and fifty-two cents. If this is correct, press 1. If not, press 2."

4.6. Telephone Numbers

Telephone numbers vary in length depending on the local network. In the input prompt, specify the expected length of the telephone number, and whether or not special prefixes are required for long distance calls. Sample input prompts are shown below:

Service:	"Please enter the 4 digit extension."
Service:	"Enter your area code and phone number, then press pound."
Service:	"Dial the forwarding number as you would if you were placing a call."
Service:	"Dial the 7 digit phone number. For long distance calls, include 1 plus the area code."

4.7. Spelling

Spelling information is accomplished using the letters on the touchtone keypad. The letters Q, Z and punctuation are typically not printed on the touchtone keypad. However, the ISØIEC international standard (ISØIEC 9995 Part 8, 1994) places the letter Q on the 7 key, and Z on the 9 key (see also Blanchard, Lewis, Ross and Cataldo, 1993; Davis, 1991; and Marics, 1990). In most applications, users can skip over punctuation when entering information. If the punctuation is essential, such as in an inventory or stock number, then users need to be given instructions on how to enter the punctuation.

Figure 1. Alphabet Assignments to a 12-key Numeric Pad

There are two styles of alphabet entry in use: unique letter entry, and *ambiguous* letter entry. Ambiguous entry uses one keypress per letter. A person using this method to enter the name "Pat" would press the 7 key for P, the 2 key for A, and the 8 key for T. There are 27 possible letter combinations formed by the keys 7-2-8. A database look-up or statistical algorithm can be used to guess which word was entered. If several matches are found, the user is presented with a menu of the possible choices and is asked to choose which item was intended.

In unique entry, each individual letter is specified. For example in the repeat key method, users press a key 1, 2 or 3 times depending on whether the first, second, or third letter on the key is desired. For example, to enter the name "Kim," a user would press the 5 key twice, the 4 key three times, and the 6 key once. Although each letter is entered, there may be problems distinguishing between adjacent letters if both letters fall on the same key. Additionally, users may forget to account for the Q and Z when entering letters using the 7 and 9 keys. For a description of the various unique entry methods, see Kramer (1970) or Detweiler, Schumaker, and Gattuso (1990).

4.8. User Recordings

User recordings are a form of data input. Examples of user recordings include greetings, messages, and names. User recordings should be preceded by a record tone, which indicates to users that they can begin speaking. In telephone answering applications after an outgoing greeting has been played, the record tone can also be detected by other automated applications to indicate that a machine, not a human, has answered a call. The ISO standard on voice messaging user interfaces (ISO, 1995) specifies a record tone sequence of 150 ms of a 500 Hz pure sine wave, a pause of 75 ms, and 150 ms of 620 Hz sine wave. The frequencies and durations should be accurate to plus or minus 2% to be standard compliant.

User recordings can be delimited by either a keypress, typically #, or by a silence timeout. This timeout is calculated by measuring the duration of silence in the recorded string. If the silence is longer than, for example, 4 seconds, the application assumes the user is finished recording and asks for confirmation. This silence time should then be removed from the end of the original voice recording prior to processing.

4.9. Lists

Lists occur in many phone-based interfaces. For example, users might have message distribution lists, dialing lists, or lists of calls to screen. List elements gener-

ally require many different actions - adding, removing, skipping, selecting, reviewing, and changing the contents of an element or an entire list. These actions can be direct or implied.

Implied Action - In this situation, only two actions, add and remove, are allowed. Users enter an item and if the item is not on the list it is added; however, if the item is already on the list it is removed. While conceptually elegant, this paradigm is somewhat awkward for users. Since users cannot see the actual list, they must remember the status of a given element in order to infer that the appropriate action that will take place.

Direct Action - In direct action, users request the specific action to be taken (e.g., add, remove, change, review, etc.). Users can specify the object of the action either before or after specifying the action itself. This method seems less confusing to users since the action is specified and not implied. In addition, it allows for a greater variety of actions such as reviewing or changing a portion of the data element.

In many cases, lists and list elements can be "tagged" with recorded names to aid in identifiying the object of the list actions. For example in voice messaging group distribution lists, it is easier for a user to identify group list "management action committee" rather than group list "number 5."

4.10. Schedules

Schedules can be difficult to implement in phone-based interfaces because of memory constraints and the imperfect mapping of two dimensional calendars onto a single auditory dimension. There are several different types of schedules which vary in complexity: a single, one-time event; a simple repetitive event; complementary schedules which flip between two alternatives; schedules which vary by day-of-week and time-of-day; and complex repetitive events. Little has been published about scheduling in PBI applications. See Plaisant and Shneiderman (1992) for a brief discussion of scheduling issues for graphical user interfaces.

For single events, users generally enter the day and time of the event. The day can either be a date, or a day of the week where Monday = 1 and Sunday = 7. For simple ongoing events, users generally enter the day of week and time of the event. An example of a single event might be scheduling a birthday reminder, while a simple, ongoing event might be remembering to take the trash out each Wednesday. These events are fairly straightforward and have been successfully implemented in a variety of phone-based interfaces. This type of schedule can be used successfully by first time users.

The following schedule types should be used with caution. They are most appropriate for infrequently changed schedules and motivated users. In some cases, successful usage will require training, written documentation, or memory aids.

- Complementary schedules occur when either of two schedules is active. Examples might include separate schedules for weekdays versus weekends, or school nights versus weekend nights. These schedules require entry of separate event times for each schedule, and knowledge of when the different schedules take effect. These schedules are complex and there is a high probability of confusion surrounding events which occur across a schedule boundary (e.g., between the weekday / weekend boundary).
- Time-of-Day/Day-of-Week schedules offer more flexibility and are likewise more complex. For each day, users enter one to several time intervals. The intervals can be copied across days if necessary. For example, a small business might be open Tuesday through Saturday, with early closing on Saturday and extended hours on Thursday night.
- Complex, repetitive events require entry of the start date and time, and of the repetition parameter. For example, someone might have class every other Tuesday, or meet the first Wednesday of every month. These schedules are extremely difficult to implement in a phone-based interface because of the large variety of repetition parameters to select from, and because feedback about the schedule setting is auditory rather than visual.

For all schedules, care must be taken when events cross day boundaries. For example, suppose a business owner wants to forward calls when the business is closed. The business closes at 5 pm and reopens at 8 am each morning. However, poorly designed applications might not allow schedules to cross the day boundaries. In that case, the owner would have to translate the closed hours to a 24 hour day (e.g., each 24 hour day, the store is closed from midnight to 8 am and then from 5 pm to midnight). In addition, if the store is closed on the weekend, a second schedule would have to cover Saturday and Sunday. Error checking should alert users to overlaps in schedules or unscheduled blocks. Holidays and other temporary suspensions of shedules merely increase the complexity.

5. General Application Issues

5.1. Timeouts

Voice menu applications have timeouts for user input. There are two reasons for this. First, each user of a PBI application ties up a telephone line dedicated to the application. Users who are not interacting with the application may be denying access to other users. The second reason is that if users do not respond to an input prompt, they may be having a problem. They may not have understood the prompt; They may have understood the prompt, but not know what to enter; They may not want to be at this spot in the application and they are trying to figure out how to exit; Or, they might just need more time to look up an account number of phone number.

Timeouts can occur in two contexts: When users do not take action after an application prompt (*no action timeouts*), and when users stop interacting within a sequence of actions (*inter-key timeouts*). Figure 2 gives an example of a no action and an inter-key timeout.

Figure 2. Timeouts

No Action

A no action timeout occurs when the application has prompted users for input but has not received any. In most cases, a no action timeout of about 5 seconds works well.

If users need to refer to external materials for information, such as a credit card from their wallet, the timeout should be extended. Users may require different amounts of time to enter different pieces of information. For example, users will be able to enter their own telephone number fairly quickly. However, it may take users much longer to enter their social security number. It will take users longer to spell alphabetic information into the interactive application than it will take them to enter numeric information. Elderly and differently abled users may take longer to enter information. Additionally, applications where users use telephones with the keypad built into the handset may need to allow extra time for users to access the keypad.

When a no action timeout occurs, the application should repeat the prompt or menu, and may include additional information about how to cancel a task or access help. If users timeout repeatedly, the application might return users to the nearest titled sub-menu, or disconnect them from the application.

Inter-key Timeout.

An inter-key timeout occurs when the application is receiving a string of touch-tone inputs from users. Again, a timeout of about 5 seconds works well. This

time may need to be adjusted according to the situation. For example, it may be shortened for security codes, or lengthened for credit card numbers.

When an inter-key timeout occurs, the application should do one of two things. If a timeout occurs and the input is a valid, the application should use the timeout as a delimiter (see section 4.1). If a timeout occurs and the input is not valid, the application should present an error message and ask users to reenter the data.

Timeout During Record

Timeouts during recordings should be treated as though the users had entered the delimiter. If nothing was recorded prior to the timeout, this can be treated as a recording error condition.

5.2. Hang-ups

Frequently, users will implicitly exit a voice menu application by hanging-up the telephone. Unfortunately, users will also hang-up the telephone when they get confused or wish to cancel a task. To avoid premature hang-ups, applications should confirm the completion of tasks (see also section 5.7).

When the completion of tasks is confirmed, frequent users will listen for the start of the confirmation before hanging up. First time users who listen to the confirmation, will know that the task has been performed. However, users who make a menu selection, change their mind, and then hang-up will not hear the confirmation, nor know that their action had been completed. The application will fail to meet the expectations of these users. However, users who have heard prior confirmations for tasks tend to listen for the confirmations.

Tasks should be designed around users' goals. Information central to the users' goals should be gathered last so as not to give users a false sense of task completion. For example, when leaving a message in a messaging application, expect users to hang-up after speaking the message. This means that the application should ask for the telephone number needed to deliver the message, before asking users to record the message.

Optional parameters can be set after the goal central information is gathered (Polson, Lewis, Rieman, and Wharton, 1992). Again, in a messaging application it might make sense to prompt for delivery options after users have recorded the message. Those users who wish to change delivery options are unlikely to hang-up after recording the message. Those users who are comfortable with the defaults for the delivery options are free to hang-up.

5.3. Error Conditions

Error messages should help users determine what went wrong. These messages should not blame users, nor should they scold them. Messages should suggest how to fix the problem. For example, "Invalid stock number. Stock numbers are 6 digits long. Please enter a stock number." After several consecutive errors, the application may offer context sensitive help, return users to a titled sub-menu, or disconnect.

5.4. Prompt Interrupt

The ability to interrupt a long menu with a keystroke is critical. If users know or hear the choice they want, they should be able to select it immediately. Once users have entered a command, users should be taken to the requested place in the application. This will speed a user's path through the application, increase user satisfaction and decrease call connect time.

Non-interruptable prompts should be avoided. If necessary, divide a prompt into non-interruptable and interruptable segments. The non-interruptable segment should be as brief as possible. Users should not be forced to listen to any part of an application. They should control the application, not vice versa. Consider the situation where an incorrect password has been entered. Often users will realize that they made a mistake before they have even finished entering the number. Why force users to listen to an error message when they may already know the problem? Simply let users interrupt the error message and re-enter their correct password.

Non-interruptable messages should contain only exceptionally important information. Input entered during a non-interruptable message should be ignored by the application. Only input during regular interruptable messages and prompts should be stored and acted upon.

5.5. Dial Ahead

Dial ahead is the ability to enter touchtone input before the application has requested it. If the application is one that users use frequently, they quickly will become familiar with the menu choices and keystrokes. Repeat users will be able to traverse the menu structure without listening to intervening prompts. For example, in a banking application, repeat users may enter 3-1-2 at the Main Menu to quickly learn their account balance. If repeat users want to skip menus without listening to them, that capability should be available.

If an entry error occurs when users are typing-ahead, the stacked input should be discarded and an error message played where the entry error occurred. For example, suppose a user wants to log onto voice mail and listen to the third message in the mailbox. Users enter "9-9-9-9" for their four digit security code, a "1"

to listen to messages, and two "#'s" to skip to the third message. Users' stacked input is "9-9-9-9-1-#-#". If users make a mistake entering the security code, then the stacked input should be discarded (i.e. 1-#-#) and the error message, "Invalid security code. Please re-enter..." should be played.

5.6. Help

Help generally takes two forms, either transfer of the call to a human attendant, or playback of a help prompt containing additional, context sensitive information. Help prompts should be interruptible so that any input during the prompt has the same effect as if that input had been entered from the application menu where help was accessed.

Service:	"For gift ideas, press 1. For mail order, press 2. For help, press 0."
User:	0 (wants bridal registry, but did not hear that option)
Service:	"Gift ideas include suggestions for birthday gifts, wedding gifts, and bridal registry. Mail order...."
User:	1
Service:	"For birthday gifts, press 1. For wedding gifts, press 2. For bridal registry, press 3."
User:	3

Most PBIs place the help functionality on the 0 key. During data entry, an isolated 0 followed by a timeout may be interpreted as a request for help. Otherwise, the 0 key is interpreted as a digit. During data input, users have to cancel or finish the input (either by a timeout, by pressing * or by pressing #) before being able to request help.

Brevity of prompts also applies to help messages. Elaborate explanations are easily tuned-out or forgotten by users. Help messages of over two minutes are generally too long for users to remember. In addition, users like to refer to information while performing tasks. Placing detailed help within the application, such as the steps needed to set up a schedule, prevents its effective use while users are actually performing the task. Finally, help messages can be softened by recording in an appropriate tone of voice and using "please" judiciously.

5.7. Feedback

After users have made a selection, the application should begin playing the next prompt within three seconds of receiving the input 90 percent of the time. If users interrupted a prompt with their selection, the prompt should stop playing within 0.5 seconds 90 percent of the time (ISO, 1995).

When an application performs an action, users should hear feedback telling them what has happened. Confirmation reassures users that the intended action has

occurred, and provides a sense of task closure before continuing with the application. For example,

Service:	"Transfer three hundred dollars from savings to checking. If this is ok, press 1. If not, press 2."
User:	1
Service:	"Money has been transferred. Main menu..."

The application should also confirm cancellation of input. In this situation, the application should state exactly what *did not* happen. For example,

Service:	"Transfer three hundred dollars from savings to checking. If this is ok, press 1. If not, press 2."
User:	* (implicit cancel)
Service:	"Transfer canceled. Main menu..."

5.8. Terminology

PBIs should use terminology common to users. In our experience, the users are not necessarily familiar with the names for the * and # keys, the differences between "rotary" and "pushbutton" telephones, and the differences between "pulse" and "tone" dialing.

- * - We recommend you refer to this as the "star" key. Another common name for this key is the "asterisk" key.
- # - In the United States, we recommend you refer to this as the "pound" key. American users are not always familiar with the name of this key. It may help to prompt users, upon timeout, that the "pound key is located under the 9 key." Other common names for this key vary by country and include: number sign, tic-tac-toe, box, square, hash, or sharp.
- Rotary and Pushbutton - Rotary phones are phones with a circular dial rather than a button keypad. Rotary phones do not have * and # on their dial, and use pulse (loop) signaling to the switch. Pushbutton phones typically have a 12 key keypad with * and #, and use tone or pulse signaling, see below.
- Pulse and Tone Signaling - In pulse signaling, the telephone emits a series of clicks for each digit dialed. All rotary phones use pulse signaling, and all telephone lines accommodate it. In tone signaling, the telephone emits a DTMF (dual tone multifrequency) signal for each digit dialed. Unlike pulse signaling, users' telephone lines have to be configured for DTMF signaling.

Most PBIs require access to pushbutton phones that emit DTMF. However, some applications are able to accommodate both DTMF and pulse input.

5.9. Service Set-up

Many PBIs guide users in setting up a service the first time they call in. These set-up sequences prompt users through a procedure, and explain the various elements in the service. For example, in a voice messaging service, an introductory set-up sequence might include having users set a security code, record a greeting, and explain how to retrieve messages. Some set-up sequences are only available the first time users call the service, others allow users to save the tutorial to share with all household members, and others are reached by a 1-800 number and are always available. Set-up sequences are highly effective, and are recommended for complex applications that need to be set-up before they can be used.

5.10. Rotary Callers

In every application, there will be callers who try to access the application from a rotary dial telephone, or from a push-button telephone with pulse dialing. Application designers must be prepared for this situation and give such callers a graceful exit from the application.

There are several ways to address this situation. The first prompt might tell callers that a touchtone telephone is required and give alternate instructions for rotary callers. Or, callers might be asked to input a touchtone digit. If they don't respond, the application can assume they are rotary callers and transfer them to a live attendant. A slightly more elegant approach is to first assume that callers do have touchtone phones. If they don't respond to the first request for actual input, transfer the call to a live attendant, or play a help message which gives them an alternate number to call. In considering the alternatives, evaluate what percentage of users can be expected to call with a rotary telephone, and how long they will have to wait before being given additional instructions.

5.11. Modifying Existing Applications

As with most software applications, changing requirements often dictate the need to add to, or less frequently, remove functionality from an application. PBI's are more similar to main-frame applications than they are to desktop software. Since PBI users call into a central application, whenever the central application is changed, all users are affected. Because users have not purchased or installed updated software, they might not be aware that an application has changed.

The degree to which application changes affect users depends upon the change and the user population. Sometimes significant changes in menu structure are required to add functionality. The trade-off between accommodating existing users and optimizing for the new design depends on the size of the current user base, usage patterns, and expected growth. For repeated use applications, such as voice mail, it can be expected that many users have memorized functionality/keystroke bindings and possibly programmed common sequences into a memory string. If users are interrupting prompts and navigating through an application by rote, they might not hear the newly worded prompts prior to entering keystrokes. Users can end up in an unexpected place in the interface because their old keystroke sequence has a new meaning under the revised interface.

For repeated use applications, we recommend the following. (Note that these recommendations may conflict with guidelines in Section 2.) When revising applications, avoid renumbering existing menu items. If possible, new items should be added to the end of the menu. Removed items should leave menu gaps so that users do not have to learn new key bindings. Do not move destructive functions to a previously assigned key. For example, if "repeat message" was on the 3 key, do not replace this function with "erase message" so as to avoid having users erase messages when they intend to merely repeat them. Avoid inserting steps, lengthening prompts, or making the interaction appear longer to users. Finally, because experienced users may not listen to prompts, alert users to a change in application functionality up front. For example, place a brief, non-interruptable, alerting message prior to login or directly before the Main Menu.

For single use applications, application changes are less problematic. Users of these applications are unlikely to have memorized keystroke bindings, and they are more likely to listen to application prompts. In these situations, application changes can be fairly extensive without significantly impacting users.

6. Summary

The following summarizes user interface design recommendations for voice menu applications for telephones.

Table 4. User Interface Guidelines Summary

Menu Structure

1. Title all important menus.
2. Limit each menu to 4 or fewer items.
3. Order menu items according to frequency of use, natural order, functionality and consistency.
4. Number menu choices starting at 1.

5. Number menu choices consecutively.
6. Word prompts for actions "To do action, press Y."
7. Word prompts for attributes "For attribute, press Y."
8. Use numbers, not letters or other mneumonic prompts, for menu choices.
9. Don't speak that option in the menu, if an option is not available.
10. Provide global command keys for Help, Cancel, Back-up and Skip.
11. Use global commands consistently.
12. Word menu choices so they clearly represent the functionality of that choice.
13. Keep menus as brief as possible.
14. Carefully choose menu vocabulary.

Application Prompts and Recordings

15. Use a professional voice.
16. Use a single application voice.
17. Choose a voice to match your application's personality.
18. Use synthesized speech for speaking information that continually changes and can't be assembled from prompt segments.
19. Record all prompts, menus and messages during a single session.
20. Don't design prompts just to facilitate reuse of prompt files.
21. Check to make sure your prompts don't "talk off" the application.
22. Record prompts at a fairly rapid rate.
23. Prompts should be spoken with energy.
24. Make sure the intonation of prompts does not mislead or annoy users.
25. Use pauses between phases to convey meaning.
26. Use "please" and "thank you" conservatively.
27. Keep prompts simple and concise.
28. Use language that is familiar to users.
29. Don't have the application refer to itself using pronouns, nor over-naturalize the interaction.

Data Elements

30. Tell users how to enter their input.
31. Tell users what format their input should take.
32. Give users enough time to enter the input.
33. Always accept #, if entered, as a delimiter.
34. Repeat data back to users using common terms.
35. Let users actively confirm input.
36. Applications should not require input delimiters.
37. Design for Q, Z and punctuation during alphabetic entry.
38. Precede user recordings with a record tone.
39. After an activity is completed, return users to a familiar place.
40. Tell users how to exit.
41. Match tasks to a user's goal structure.

General Application Issues

42. Give users more information if users do not enter anything.

43. Confirm the completion of application tasks.
44. Error messages should state the problem and how to correct it.
45. Let users interrupt the application with keystrokes.
46. Avoid non-interruptable messages.
47. Make non-interruptable messages as short as possible.
48. Let users type-ahead through several menus.
49. Discard input entered during a non-interruptable message.
50. Let user interrupt help prompts.
51. Tell users what action the application has taken.
52. Confirm the cancellation of input.
53. Use a set-up sequence for applications that need user data to function.
54. Give rotary callers a way out.
55. Tell users when applications have changed.
56. Avoid renumbering exiting menus.

7. References

Aucella, A.F. and Ehrlich, S.F. (1986). Voice messaging- enhancing the user interface based on field performance. *Proceedings CHI-86 Human Factors in Computing Systems*. New York: ACM. 156-161.

Bain, L. (1990). U S WEST Touch Tone Standards for Voice Prompted User Interfaces. U S WEST Technical Report.

Blanchard, H.E., Lewis, S.H., Ross, D. and Cataldo, G. (1993). User performance and preference for alphabetic entry from 10-key pads: Where to put Q and Z. *Proceedings of Human Factors Society 37th Annual Meeting,* 225-229.

Cox, A.C. and Cooper, M.B. (1981). Selecting a voice for a specified task: The example of telephone announcements. *Language and Speech*. 24(3), 233-243.

David, J.R. (1991) Let your fingers do the spelling: Implicit disambiguation of words spelled with the telephone keypad. *Journal of the American Voice I/O Society*. March 1991, 57-66.

Davies, S. (1995). Lessons learned in screenphone user interface deisgn: A human factors perspective. *Proceedings of the 2nd Advanced Screen Telephony Business Issues Forum*. April 18-19, 1995. Atlanta, Georgia.

Davis, J. R. (1988). When things go wrong: An analysis of user errors with Direction Assistance. *Proceedings of Speech Tech '88*. American Voice Input/Output Society, New York, NY. 304-306.

DeGroot, J. and Schwab, E.C. (1993). Understanding time-compressed speech: the effects of age and native language on the perception of audiotext and menus. *Proceedings of the Human Factors Society 37th Annual Meeting,* 244-248.

Detweiler, M.C., Schumacher, R. M., and Gattuso, N. L. (1990) Alphabetic Input on a Telephone Keypad. *Proceedings of the Human Factors Society 34th Annual Meeting,* 212-216.

Dobroth, K.M., Karis, D., and Zeigler, B.L. (1990). The design of conversationally capable automated systems. *International Symposium on Human Factors in Telecommunications.*, 389-396.

Engelbeck, G. and Roberts, T. (1990). The effects of several voice-menu characteristics on menu-selection performance. *Proceedings of the Bellcore Symposium on User Centered Design*. Bellcore Special Report SR-STS-001658. Red Bank, NJ. 50-63.

France Telecom (1991). User -friendly recommendations for voice services designers. Ref: NT/LAA/TSS/426-TARIF:150 F HT (177, 90 TTC).

Goodwin, Pat (1988). Interface usability evaluation of project Aries. U S WEST Advanced Technologies Technical Report. PS-08-05-00003-001.

Halstead-Nussloch, R. (1989). The design of phone-based interfaces for consumers. *Proceedings CHI-89 Human Factors in Computing Systems*. New York: ACM. 347-352.

Halstead-Nussloch, R., DiAngelo, M., and Thomas, J. (1989) Designing Phone-Based Interfaces. Workshop presented at the Human Factors Society Meeting, Denver, Colorado. October 16-20, 1989.

Halstead-Nussloch, R., Logan, R., Campbell, R., Roberts, J. (1989). Usability test of the self help phone interface. *IBM Technical Report TR 00.3534.*. Watson Research Center, Yorktown Heights, NY.

Information Industry Association. (1990). Voice Messaging User Interface Forum Specification Document. Information Industry Association, 555 New Jersey Ave. NW, Washington, D.C. 20001

International Standards Organization. (1994). Information Technology - Keyboard Layouts for Text and Office Systems - Part 8: Allocation of letters to the keys of a numeric keyboard. ISØIEC 9995-8. Geneva, Switzerland.

International Standards Organization. (1995). Information Technology - Document Processing and Related Communication - User Interface to

Telephone-based Services - Voice Messaging Applications. ISØIEC 13714:1995(E). Geneva, Switzerland.

Kidd, A.L. (September, 1982). Problems in man-machine dialog design. *Paper presented at the Sixth International Conference on Computer Communication* (531-536), London.

Kramer, J.J. (1970). Human factors problems in the use of push-button telephones for data entry. *Proceedings of Symposium on Human Factors in Telephony.* Berlin. 241-258.

Lee, E. and MacGregor, J. (1985). Minimizing users search time in menu-retrieval systems. *Human Factors*, 27(2), 157-162.

Luce, P.A., Feustel, T.C., Pisoni, D.B. (1983). Capacity demands in short-term memory for synthetic and natural speech. *Human Factors*, 25, 17-32.

Marics, M.A. (1989). Voice Characteristics for Auditory Interfaces. Poster session at *The Human Factors Society 33th Annual Meeting.* Denver.

Marics, M.A. (1990). How do you enter D'Anzi-Quist using the telephone keypad? *Proceedings of the Human Factors Society 34th Annual Meeting.* Orlando, FL. 208-211.

Mulligan, R.M., Whitten II, W.B. and Tsao, Y-C. (1988). Parameters of information-rich auditory announcements. *Proceedings of the Human Factors Society 32th Annual Meeting,* 242-246.

Oksenberg, L. and Cannell, C. (1988). Effects of interviewer vocal characteristics on non response. Chapter in *Telephone Survey Methodology.* 257-269.

Plaisant, C. Shneiderman, B. (1992) Scheduling home control devices: design issues and usability evaluation of four touchscreen interfaces. *International Journal of Man-Machine Studies.* 36, 375-393.

Polson, P., Lewis, C., Rieman, J., and Wharton, C. (1992). Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, 36, 741-773.

Resnick, P and Virzi, R.A. (92) Skip and scan: Cleaning up telephone interfaces. *Proceedings CHI'92, Human Factors in Computer Systems.* New York: ACM. 419-426.

Rosson, M. B. and Cecala, A. J. (1986). Designing a quality voice: an analysis of listeners' reactions to synthetic voices. *Proceedings CHI '86 Human Factors in Computer Systems.* New York: ACM. 192-197.

Schmandt, C. (1994). *Voice Communication with Computers - Conversational Systems*. Van Nostrand Reinhold, New York.

Schwartz, A. and Hardzinski, M.L. (1993). Ameritech phone-based user interface standards and design guidelines, Rel 1.0. Ameritech Services. Hoffman Estates, IL.

Stuart, R., Desurvier, H. and Dews, S. (1991). The truncation of prompts in phone based interfaces: Using TOTT in evaluations. Proceedings of the Human Factors Society 35th Annual Meeting. 230-234.

Waterworth, J.A. (1982). Man-machine dialog acts. *Applied Ergonomics*, *13*, 203-207.

Yankee Group (1994). White paper on consumer communications. YankeeVision, 11 (7), May 1994.