CHAPTER 8

VUI Design Principles and Techniques

THE FOLLOWING SECTIONS provide introductory guidance about best design practices for VUIs. Comprehensive coverage of the entire VUI design process is beyond the scope of this book. The intent here is to make you aware of some of the key issues that must be addressed in VUI design and provide an overview of some of the techniques that are available to address those issues.

Core Principles

Among speech interface designers, there's a credo: A good GUI and a good VUI are both a pleasure to use, a bad GUI is hard to use, but a bad VUI isn't used at all. This will be a major issue as VoiceXML puts voice technology in the hands of developers everywhere. You may remember in the early days of the Web, when HTML tools were just becoming available, that the number of truly awful Web site designs took off. Well, that's where VoiceXML will be very soon, and you can expect many unusable VUIs to be written.

Keep It Simple, Do It Well

Be realistic about what can be accomplished through a VUI. Focus on handling the easiest 80 percent of requests simply and cost-effectively.

A famous French proverb states, "Nothing succeeds like success." It seems obvious: A good interface is one that gets the job done. However, who has not experienced the frustration of calling a customer service phone number and immediately being thrust into a Kafka-esque maze of voice prompts and menu choices, none of which meet your need?

There is tension between the needs of people calling in ("users") and businesses that provide the service ("providers"). Users want to have their needs quickly and courteously met. From the user's perspective, nothing beats having a well-trained and empowered person on the vendor's end of the line. From the provider's perspective, well-trained and empowered customer service representatives are expensive. Vendors want the call center to please customers in

a cost-effective manner.[1] As Table 8-1 shows, users and providers do not use the same criteria when measuring VUI usefulness and effectiveness.

Table 8-1. Satisfaction Criteria for VUI

USER CRITERIA	PROVIDER CRITERIA
Can I get the information or perform the transaction I want?	Does it reduce the load on customer service reps?
Is the result worth my effort to get it?	Are users satisfied with the experience?
Do I feel like I'm receiving a valuable service?	Does it increase the number of users I connect with?

By these criteria, a VUI that gracefully and elegantly ends up routing most calls to a human operator is not meeting provider needs. On the other hand, a VUI that never routes a call to a human operator is not meeting the needs of some users. Achieving a balance between the sometimes conflicting requirements of users and providers is part of the design process. Skewing the balance too heavily to the provider's side results in a VUI that is overly comprehensive, loaded with options, and frustrating to use. Skewing the balance too heavily to the user's side results in a VUI that is expensive to build and operate.

When designing a VUI, be realistic about its capabilities. Don't assume that the VUI should be able to do everything a GUI can. Use the 80/20 rule: Aim to simply and effectively handle the easiest 80 percent of the load, and leave the other 20 percent to other means (usually human operators).

Accommodate Errors

Errors are not exceptions in speech and cannot be eliminated. A good VUI should be deceptively simple—simple in its basic dialog structure, but complex in its capability to perform in the presence of a multitude of errors.

Communication by speech is inherently a complicated and error-prone process. Computer programmers are biased to treat errors as defects—failings to be eradicated. However, when you develop VUIs, you have to overcome this bias: Errors are essential and inherent. The goal is not to eliminate errors, but rather to contain them and tolerate them. Natural languages are marvelously error

[1] All lofty rhetoric aside, a reason businesses are interested in VUIs is to reduce (or eliminate) the cost of maintaining call centers. VoiceXML will accelerate this interest because it opens the possibility of collapsing two infrastructures (call center and Web) into one.

tolerant. People can communicate effectively despite mispronunciations, jackhammers in the background, misunderstandings, verbal stumbles, interjections, false starts, incomplete sentences, poor grammar, uncouth accents, and so on. Developing a good VUI has a lot in common with programming a real-time process control system like that which runs a power plant. There's a lot going on in real time: There are basics laws of physics at work, relative timing of events is crucial, and errors that occur need to be managed so that they damp down rather than amplify over time.

As mentioned previously, VoiceXML has no cognitive model, and so it is not an appropriate instrument for implementing natural language. More precisely, VoiceXML cannot match the expressiveness of natural language, though it can borrow profitably from the error tolerance of natural language. The goal of a good VoiceXML VUI is to be deceptively simple—simple in that the basic structure of dialogs should be simple and easy, and deceptive in that the "simple" dialogs should be possible in the face of a multitude of errors. In terms of the effort invested in developing a VUI, the minority of effort should be spent on the basic dialogs. The majority of effort should be spent on detecting errors, recovering from them, and getting the conversation back on track.

Design for Everyone, Everywhere

People don't all speak alike. Bear in mind that users will possess a wide variety of voices, speech skills, and vocabularies. The Web is a public place: A voice-enabled Web site can expect to hear from people with all kinds of backgrounds, speech skills, vocabularies, and voices. In designing a VUI, keep the response vocabulary simple and generic. Avoid disenfranchising nonexpert users by using specialists' jargon. Make sure that the VUI can adapt to situations where speech recognition isn't working well.

Speech recognition technology currently achieves the highest recognition rates for adult native speakers of English, and it achieves lower rates for children and non-native speakers. In addition to natural variations in people's speech, speech recognition professionals also refer to "sheep" and "goats"[2]: "Sheep" are people for whom speech recognition works exceptionally well, and "goats" are people for whom it works exceptionally poorly. Only the voice recognizer knows what separates them. People can't predict whose voice will be recognized easily and whose won't. The best policy is to design the interface so it can handle all kinds of voices in all kinds of environments.

[2] From *How to Build a Speech Recognition Application,* by Bruce Balentine and David P. Morgan, Glossary.

Speech Design

The goal of designing an effective VUI in VoiceXML is to create a small speech dialect that is constrained in vocabulary and phrase structure, effective for the task at hand, and tolerates errors. In other words, you're not trying to make the computer talk at the level of a person. You're trying to get a person to speak at the level of the computer.

While this may sound like interface blasphemy (making the person accommodate the computer!), it is not unprecedented. People comfortably and effectively use a variety of dialects daily, without feeling put upon or constrained. For example, people talk differently to a child than they do to an adult, because a child's speech and comprehension skills are less than those of an adult. Similarly, people talk to their pets all the time, and successfully communicate attitudes and emotions, even though the speech is (in human terms) nonsense. When talking to a non-native speaker, people unconsciously adjust their speech patterns to communicate more effectively. A computer is less linguistically competent than a person is, whether that person is a child or adult, so it makes sense that people adopt a restricted form of speech to facilitate communication.

In terms of speech design, it's important that the designer make the computer speak like one. People's speech expectations are based on their perceptions of whom they're talking to. If a person doesn't realize he or she is speaking to a child, for example, that person will perceive that he or she is talking to a very strange adult. Similarly, if a computer attempts to sound like a person, people will have expectations of a person rather than a computer. So the designer should present the computer as what it is: an automated speaker of a simplified dialect. Unfortunately, people's expectations of computer speech tend to spontaneously fly to the HAL model, but this can be addressed by good design.

Modeling

In talking with one another, and with computers, people tend to model their speech on the other party's. If the second party speaks tersely and rapidly, the first party will tend to speak tersely and rapidly as well. This unconscious mimicry applies to many aspects of speech: vocabulary choice, phrase structure, volume, pitch, rate, and so on. Modeling is one of the most powerful tools available to a speech designer for tacitly cluing in users to acceptable forms of speech. The following are dos and don'ts for using modeling in a VUI:

- *Do* use computer prompts that are brief and to the point. The style of the computer's speech should instill the impression that the conversation is a professional one—not friendly or unfriendly, but directed to a purpose.

- *Do* start out with examples of acceptable speech when you provide help rather than try to explain what's going on. People are more often confused about the form of what they can say to the computer than they are about the meaning.

- *Don't* use long, wordy prompts in the vernacular because people will respond in kind.

Disfluency

One of the biggest inhibitors of continuous speech recognition is that people do not speak continuously. People's speech is peppered with "um"s, "ah"s, "er"s, pauses, and other fillers. People correct their own speech on the fly: "Give me two—no, three sodas," "I want—need—a cup of coffee." People may abandon forms of speech in midstream: "I was just wondering . . . oh, never mind." People subconsciously filter out these disruptions in the normal flow of speech, both as listeners and speakers, and consequently do not realize how frequent these *disfluencies* are. One way to gauge how much you expect disfluency in speech is to think about how unnatural synthesized speech sounds. It sounds unnatural for at least two reasons: its prosodic limitations (for example, lack of modulated pitch and emphasis) and its total (inhuman) fluency.

Disfluency causes problems in speech interfaces because they are resolved at multiple levels. Pauses and fillers can be fairly effectively filtered by a voice recognizer, but they also can lead to errors (for instance, inserting an extra word) that will baffle a semantic analyzer. Self-corrections occur at the semantic level and are hard to model in grammars.

The occurrence of disfluencies increases rapidly as a function of the length of an utterance. The longer a person speaks without a break, the greater the number of disfluencies. Therefore, the main tool for dealing with disfluency in speech design is limiting the length of utterance. There are tradeoffs in this approach, however. When a person starts using a VUI, there is a lot of disfluency due to the person's uncertainty about what to say. As the person becomes more familiar with the VUI and becomes more adept at speaking in a way the computer can understand, he or she wants to make longer utterances for efficiency's sake. Heed the following tips about how to design a VUI to minimize the effects of disfluency:

- *Do* structure dialogs using mixed-initiative combined with directed forms. This enables users to convey as much information as they are able in a single utterance, but it elicits information in smaller snippets if there are problems communicating.

- *Don't* try to address disfluency through grammar design. It can greatly increase the grammar complexity and slow down recognition, but the chances of ultimate success are slim.

Synthesized Speech

Listening to synthesized speech requires more concentration and effort than listening to human speech. People vary their intonation, pitch, volume; introduce pauses; and provide any number of behavioral "clues" that cue the listener as to the structure and meaning of the spoken content. Without these cues from the computer, people are working with less information and have to try harder to decipher what's being communicated. This also means that lapses in attention are harder to recover from, and long messages are harder to comprehend. Although techniques for maximizing the comprehensibility and effectiveness of synthesized speech depend on the particular speech synthesizer being used, here are some general tips on using speech synthesis:

- *Do* use recorded prompts wherever possible. Any fixed prompts (for example, menu choices) should be recorded. Speech synthesis should be reserved for reading information that varies (for example, a person's appointments today).

- *Do* pay attention to prosodic features when using synthesized speech. For example, when reading lists, introduce pauses to demarcate individual items.

- *Don't* use synthesized speech to read long lists to a person and expect the person to comprehend or remember the lists. The interface should be designed so that lists contain no more than four or five items. This may require introducing dialogs to refine the person's focus of interest (for example, rather than reviewing all 15 of today's appointments, start with five of this morning's appointments).

NOTE *See Table 9-1 in Chapter 9 for a list of the VoiceXML tags that affect how synthesized speech sounds.*

Getting the Most Out of Speech Recognition

Recognizing speech is not an exact, analytical science. It is a probabilistic art and incorporates elements of sophisticated guessing. There are some design principles that can help you make the most of this inexact but powerful tool. The basic principle is to make expected responses as distinct as possible. Unfortunately, a person's intuition about phrases that sound "close together" or "different" isn't a very good predictor of how a voice recognizer will perform. As anyone who has used a computer dictation system knows, voice recognizers make the oddest mistakes. This means that grammars need to be field tested for recognition problems.

Some gross properties of responses, however, do have a strong effect on recognition. The main property is response length. The best responses are brief phrases that give the recognizer some meat to chew on, but are not so long that disfluencies crop up. So, for example, "Speak louder" is preferable to "Louder." This is particularly true for background grammars. In VoiceXML, several grammars may be active at once:

1. Field grammar (awaiting a response to a prompt for information)
2. Form-scoped links
3. Document-scoped links
4. Application-scoped links
5. Voice interpreter default grammars (help, cancel, and so on)

Lower numbered grammars are "in front of" higher numbered grammars in the sense that in case of a toss-up, the recognizer prefers to match to lower numbered grammars. The shortest prompts should be reserved for the closest grammars (field and form), and links with broader scope should be longer phrases that can be recognized in a variety of contexts.

Numbers and letters are problematic because they are so short. For example, "six" (the shortest spoken digit in English) is commonly both falsely inserted (recognized but not spoken) and falsely deleted (spoken but not recognized) by recognizers. This does not mean that numbers and letters aren't recognizable and shouldn't be responses, but it does mean the designer should be aware of potential problems. Some possible solutions are to allow DTMF input where numbers are expected and to use the International Communications alphabet for letters (alpha, bravo, charlie, delta, and so on instead of a, b, c, d, and so on). The following are some techniques for effective use of speech recognition:

- *Do* use short phrases or multisyllabic words for links (for example, "Start over," "Speak louder").

- *Do* reserve the shortest, commonest responses for field-level responses.

- *Do* allow use of DTMF where precise input of numbers is important.

- *Don't* share recognition errors with the person. While the person may find it reasonable to ask for clarification with a question such as "Did you say Austin or Boston?", the computer will spontaneously come up with something goofy such as "Did you say Austin or hippopotamus?" Instead, ask for clarification directly (for example, "I didn't get that. What city?").

- *Don't* make field and form grammars needlessly broad (for instance, with lots of synonyms), because they will interfere with recognition of background links.

Interface Design

When using a good VUI, a person will feel oriented, in control, and able to anticipate what will happen next. A person is oriented if he or she can relate the current dialog to the task he or she wants to perform. A person feels in control if the person feels he or she knows what to do to affect the way the interaction with the computer proceeds. To be able to anticipate what will happen next, a person needs a mental model of "what the computer is up to." (Notice that the computer has no cognitive model, and therefore is not really "up to" anything. However, a good interface will superimpose a model purely for the person's benefit.)

Ironically, some of the biggest challenges in designing a good VUI bear a strong resemblance to challenges faced by developers in the days of command-line interfaces and small-screen displays. Probably the biggest challenge is keeping the person cued about what he or she can do next. In VUIs, this means knowing what you can say in response to a voice prompt; in command-line systems, it means knowing what commands (and options) you can type at the command prompt. The other big challenge is presenting information in small enough chunks that people can absorb and use it. In VUIs, reading long lists of options is time-consuming, frustrating, and ineffective. On small screen displays, printing out more than a screen full of data caused some of the data to scroll off the screen. In both cases, you have to rely on human memory to retain what's important.

The eventual resolution of command-line issues turned out to be windowed GUIs. Commands became menu options, options became property sheets, and windows became scrollable. This resolution is unfortunate for VUI designers,

because the graphical solution cannot be transferred to voice. For a variety of reasons, VUIs must succeed as an alternative to GUIs, and they must work where GUIs don't. Hopefully, during the upcoming decade we will see the evolution of interface techniques and technologies for handling the shortcomings of voice as a medium, just as in the past decade we've seen the development of GUI technology to supplant command lines.

Turn-Taking and Error Amplification

A big challenge in VUI design is handling issues related to turn-taking. Because there are no behavioral or visual cues accompanying the conversation, it is very common for the two parties to lose track of whose turn it is. Once a conversation goes off track, it can rapidly degenerate into chaos (see Dialog 8-1).

Dialog 8-1. Inexperienced User[3]

P (person): Umm, I'd like to know, ahhh . . .
C (computer): I'm sorr–
P: I'd like to know . . . the number of sh–
C: I'm . . .
P: I want to sell half my shares of Motorola.
C: I didn't understand.
P: I'd like to SELL . . . half my shares . . .
C: I didn–
P: . . . of Motorola
C: I didn't under–
P: I'd like to SELL . . . Motorola.
C: I didn't understand.

In this example, an inexperienced user pauses because of uncertainty about how to proceed. The computer interprets the pauses as the end of unrecognized responses and responds with an error prompt. The user pauses when the computer speaks, trying to yield his or her turn, but the computer is busy digesting the remnant of the user's previous attempt to spit out a whole sentence, which it can't recognize, and so on. The dialog spirals out of control, as the user attempts to get through to the brain-damaged computer by increasing volume and overstressing "sell," which guarantees that the computer will not recognize the speech. At this point, the only solution is to separate the combatants for a cooling-off period.

[3] From *How to Build a Speech Recognition Application,* by Bruce Balentine and David P. Morgan, pg. 99.

This is an example of *error amplification*, in which an initial small error rapidly escalates into a wholesale breakdown in communication. It can be initiated by virtually any kind of error that confuses turn-taking. This kind of error escalation is familiar to designers of real-time process control systems, and it is also known as *cascading errors*. A major goal of interface design is to structure the interface so that when anything goes astray, the interface will get the person to a state where he or she is once again oriented, in control, and knows what's coming.

To help stabilize dialogs and prevent error amplification, build *safe points* into the interface. A safe point is a known state of the application, usually a high-level menu, at which the person and computer can resynchronize turn-taking and generally get a fresh start. The tricky part about safe points is figuring out when to transition to them. If things are going badly, the person and computer may not be listening well to each other, and so normal prompting may be ignored. A last-ditch technique to stabilize a dialog is to prompt the user with a modal prompt that requires a "yes" or "no" response:

Computer: Do you want to continue adding appointments to your calendar? Say yes or no after the beep. {beep}

A "no" answer would initiate transfer to a safe point. Such a prompt can occur after a certain number of errors occur in a row.

See the end of the following section for some tips that affect both turn-taking and orientation issues.

Lost in Space

Because speech is not persistent, people must rely on their memory to know "where they are" in a conversation. Such short-term memory is fragile and can easily be disturbed by any kind of distraction: avoiding a road hazard while driving, another person entering the room, a child interrupting his or her mother, a fire truck screaming by, and so on. When these distractions occur during a conversation, a person may recover by saying something like "Where was I?" The other person reorients the distracted one, and the conversation resumes.

Feeling "lost in space" and out of control should be dealt with in a good VUI. Distraction is a common source of disorientation. Disorientation also results when the computer goes down a path the person was not anticipating. The person is still thinking about the task at hand, but the computer's prompts have become irrelevant.

A good technique for maintaining orientation is to provide auditory cues along the way, so that the person receives some feedback about where he or she is. For example, different parts of an application might use different voices. In

the SPIM example, you might use different voices for Calendar, Contact, and Appointment. *Earcons,* or auditory icons, can accentuate meaning of different types of prompts. For example, different tones can be used for confirmations, errors, and menu choices. This does not require that the conversation rely on tones for turn-taking (as in "Wait for the tone before speaking"). Bargein can be enabled so that experienced users do not have to wait, while inexperienced users will get greater contextual feedback. Music or other nonspeech audio can be used to signal that the computer is working (and hence is holding on to its turn).

A related technique is to include orientation tips in prompts generated when the person is silent or can't be understood.[4] For example, the user might receive the following prompt if they do not respond:

Computer: You are in your address book. Adding entry for Veronica Voice. What is the home phone number?

This technique should be used judiciously, in tandem with tapered prompting, because the additional orientation information increases the prompt length and may frustrate users who are making small errors but are not disoriented.

The following tips apply to good design with regard to turn-taking and orientation:

- *Do* be aware in designing all dialogs that distraction can strike at any time and that errors may be the result of a loss of orientation rather than a lack of knowledge.
- *Do* include some auditory feedback to maintain orientation and signal when it's the computer's turn.
- *Do* provide application safe points where the user can reorient him- or herself and resynchronize turn-taking.
- *Do* incorporate last-ditch error handling to avoid runaway error amplification.
- *Don't* use too many tunes, tones, or other nonspeech audio—it becomes tiring to listen to repeatedly.

[4] This is the audible equivalent of the technique used on U.S. interstate highway marker signs, which seem to have a formula for orientation. The signs show where you are (I95S), the next major destination (New York City 100mi), and the next destination (West Nowhere 2mi).

- *Don't* have the computer take drastic unilateral action (for example, transferring the user to an operator) without confirmation from the user.

- *Don't* force a lot of contextual information on the user unless he or she requests it.

Accommodating Different Experience Levels and Environments

People using your VUI will have varying levels of experience with voice interfaces in particular and with computers in general. One of the motivators for enabling Web applications with voice is to reach the user base that does not use PCs, digital networks, keyboards, and mice. So, at one end of the spectrum, novice users require an interface that is very simple, easy to use, and undemanding of the user. At the other end of the spectrum, advanced users require an interface that performs efficiently in a car, from a public phone, and in other mouse-less and keyboard-less environments.

Experience manifests itself in the following ways:

- Experienced users don't need constant cuing about what's expected of them. Too many cues can make experienced users impatient with unneeded prompts and more aggressive about barging in.

- Experienced users are more likely to retain their cool when errors crop up and to let the computer "catch up" when turn-taking errors occur.

- Experienced users require occasional help on options available in unfamiliar parts of the application, but they do not tolerate long help prompts.

- Experienced users can become disoriented as a result of distractions.

- Experienced users learn to speak to the computer so they will be understood, using constant volume, stress, and rate; enunciating fully; and so on. They are therefore capable of longer productive utterances than novices are.

It's tempting to think that the error rate decreases as user experience increases, but this may not be the case. The reason for this is that errors are a product not only of speech, but of environment as well. In fact, it's quite

possible that experienced users will only use voice when other options are not available. So, when an experienced user is in the quiet of his or her home with a good microphone and no interruptions, he or she will use a mouse and keyboard. When the same user is at an airport with flight announcements randomly drowning out speech, he or she will use voice.

Environmental characteristics that affect how well the VUI performs include the following:

- *Background noise:* Recognizers are surprisingly good at screening out constant background noise, but both people and computers have trouble dealing with "bursts" of noise (for example, airport speakers, thunder, and so on).

- *Other people talking:* If you're sitting on your couch with your parrot chattering on your shoulder and Peter Jennings blaring on the TV, the computer is going to try to recognize everybody's speech—not just yours.

- *Voice fidelity:* The computer tries to recognize speech as received at the voice gateway. Anything between the larynx and gateway that degrades voice quality degrades performance. Low-quality microphones and poor telephone connections are the usual suspects.

Good interface design attempts to make the VUI adjust to speaker and environmental variations by trading off efficiency for robustness. With an experienced speaker and a good environment, dialogs should permit (but not require) longer utterances. With a naive user and a noisy environment, dialogs should collapse to brief, structured interchanges that are tedious but reliable. Essentially, this strategy adapts the dialogs to keep the error rate down. If there are many problems communicating, slow down and take it a step at a time.

Using VoiceXML mixed-initiative dialogs is a good way to permit a user to use longer utterances. The user can respond to the initial prompt with as much or as little information as he or she wants. The dialog then switches into directed mode and elicits information a piece at a time. This can require sophisticated grammar design, so that both short and long utterances are successfully recognized (see Table 8-2).

Table 8-2. Novice and Expert Mixed-Initiative Dialog

NOVICE USER	EXPERT USER
C (computer): Add appointment. You may say help at any time. *P (person): Help.* *C: Say something like call Joe.* *P: Call Pete.* *C: What day?* *P: Friday.* *C: What time?* *P: 10 a.m.* *C: Where?* *P: Work.* *C: You have a call with Pete on Friday at ten am at work.*	*C (computer): Add appointment. You may say help at any time.* *P (person): Call Pete on Friday at 10 a.m. at work.* *C: You have a call with Pete on Friday at ten am at work.*

As mentioned previously, falling back to a yes/no interrogation gives about as much reliability as is possible: The recognizer only has to match against two possible responses. People don't tolerate a "20 questions" style of interface in general, but it may be acceptable in extreme circumstances, as shown in Dialog 8-2.

Dialog 8-2. Fallback to 20 Questions

C (computer): Do you want the shirt in red, green, or blue?
P (person): {static}
C: Do you want the shirt in red, green, or blue?
P: {static}
C: Do you want red? Say yes or no.
P: No.
C: Do you want green? Say yes or no.
P: Yes.
C: Do you want small, medium, large, or extra large?

Because of this adaptability, the interface may become more complex, and there may be many valid responses to any given prompt. So, doesn't that make it more complicated to cue the user as to what's expected? Not necessarily. The support structure of the interface—help prompts, event handlers, and so on—should be primarily geared toward the simplest way of doing things. Even if there are ten ways to do something, the basic prompting should describe one way: the simplest way. The other nine options are still available, but they are not explicitly announced or described. The process of turning novices into experts should be treated as a training issue, not a prompting issue. Possible techniques for enlightening advanced users include offline tutorials and toggling a mode switch that switches

the interface between "novice" mode and "expert" mode. Also, consider the following tips for accommodating different experience levels and environments:

- *Do* provide opportunities for expert users to use shortcuts.
- *Do* use mixed-initiative dialogs backed up with directed prompting for filling out forms.
- *Do* incorporate yes/no exchanges as the fallback when more complicated dialogs are not working.
- *Don't* clutter up basic prompts with a lot of tutorial material aimed at expert users.
- *Don't* assume that if a user encounters a lot of errors it means he or she is "slow"—it may mean that an expert user is in a tough environment.

Taking Time Out

When talking on the phone, people often engage in sidebar "mini conversations" with people around them, without explicitly interrupting the phone conversation. There are all kinds of amusing behaviors associated with these sidebars, including the following:

- Exaggerating facial expressions, mouthing words, pointing at a watch, and so on
- Pantomiming, as in the game charades
- Writing notes on paper or whiteboards
- Commenting in a sotto voce to someone nearby
- Putting a hand over the phone's mouthpiece or enabling the phone's mute feature to engage in a full-fledged conversation with someone else

While people seem to be able to converse through various types of conversational lapses, to a computer they sound like unrecognizable speech. The speech may be unrecognizable because it's disfluent, because the person's normal speech cadence is altered, because the computer hears things a person would filter out, or because a person misses timing on a response and speaks too

soon or too late. Regardless of the reason, the person can suddenly find the computer reacting to "errors" and conversational turn-taking disrupted.

The VUI should provide ways for a user to pause or suspend a dialog at any point. When the user is using a microphone, issuing a command to turn the microphone off and on works well. On a phone, the computer cannot control the microphone, so a voice command that causes the computer to ignore input for a while is required. The command should be available at any prompt, as shown in Dialog 8-3.

Dialog 8-3. Taking Time Out for Distractions

C (computer): Please say the name of the person . . .
P (person): {barging in} Go to sleep.
. . .
P: Wake up.
C: Please say the name of the person to call.
OR
C: Please say the name of the person to call.
P: Give me thirty seconds.
{after 30 seconds}
C: Are you ready to continue?
P: Give me thirty seconds.
{after 30 seconds}
C: Are you ready to continue?
P: Yes.
C: Please say the name of the person to call.

> **CAUTION** *VoiceXML does not provide an easy way to temporarily suspend the VoiceXML interpreter as described here. (In my opinion, this is a shortcoming of VoiceXML.) Implementing the design described here requires the use of proprietary features of a particular voice platform.*

Summary

Careful VUI design is not an option—it is a requirement. A poorly designed VUI can be not only frustrating to the user, but also outright insulting and provocative. This chapter elaborated three core principles of VUI design: deceptive simplicity, error accommodation, and designing for everyone. Some basic behavior and psychological characteristics of speech were also introduced, along with tips and techniques for dealing with some of the common pitfalls and challenges of VUIs.

For pragmatic, hands-on tips about implementing speech recognition applications, I recommend reading *How to Build a Speech Recognition Application,* by Bruce Balentine and David P. Morgan. For a broader view of speech interfaces in general, I recommend *Designing Effective Speech Interfaces,* by Susan Weinschenk and Dean T. Barker (the companion Web site address is `http://www.wiley.com/compbooks/weinschenk/`). Also, the June 2001 issue of *VoiceXML Review* (`http://www.voicexmlreview.org/`) covers human considerations for VoiceXML applications.