- Table of Contents

Voice User Interface Design

By Michael H. Cohen, James P. Giangola, Jennifer Balogh

Publisher : Addison Wesley
Pub Date : February 06, 2004
ISBN : 0-321-18576-5
Pages : 368
Slots : 1.0

This book is a comprehensive and authoritative guide to voice user interface (VUI) design. The VUI is perhaps the most critical factor in the success of any automated speech recognition (ASR) system, determining whether the user experience will be satisfying or frustrating, or even whether the customer will remain one. This book describes a practical methodology for creating an effective VUI design. The methodology is scientifically based on principles in linguistics, psychology, and language technology, and is illustrated here by examples drawn from the authors' work at Nuance Communications, the market leader in ASR development and deployment.

The book begins with an overview of VUI design issues and a description of the technology. The authors then introduce the major phases of their methodology. They first show how

to specify requirements and make high-level design decisions during the definition phase. They next cover, in great detail, the design phase, with clear explanations and demonstrations of each design principle and its real-world applications. Finally, they examine problems unique to VUI design in system development, testing, and tuning. Key principles are illustrated with a running sample application.

The cover photograph depicts the first ASR system, Radio Rex: a toy dog who sits in his house until the sound of his name calls him out. Produced in 1911, Rex was among the few commercial successes in earlier days of speech recognition. *Voice User Interface Design* reveals the design principles and practices that produce commercial success in an era when effective ASRs are not toys but competitive necessities.

- Table of Contents

Voice User Interface Design

By

Michael H. Cohen, James P. Giangola, Jennifer Balogh

Publisher : Addison Wesley
Pub Date : February 06, 2004
ISBN : 0-321-18576-5
Pages : 368

Slots : 1.0

Part I:

Part II:

Definition Phase: Requirements Gathering and High-Level Design

Chapter 4.

Requirements and High-Level Design Methodology

Section 4.1.

Requirements Definition

Section 4.2.

High-Level Design

Section 4.3.

Conclusion

Chapter 5.

High-Level Design Elements

Section 5.1.

Dialog Strategy and Grammar Type

Part III:

Design Phase: Detailed Design

Part IV:

Realization Phase: Development, Testing, and Tuning

Copyright

Library of Congress Cataloging-in-Publication Data

Voice user interface design / Michael H. Cohen, James P.

Giangola, and Jennifer Balogh.
p. cm.
ISBN 0-321-18576-5 (alk. paper)
1. Automatic speech recognition. 2. Human-computer interaction.
3. User interfaces (Computer systems) I. Cohen, Michael H. (Michael
Harris), 1953- II. Giangola, James P. III. Balogh, Jennifer.

TK7895.S65.V735 2004
621.39'9dc22
2003022900

Text printed on recycled paper

1 2 3 4 5 6 7 8 9 10CRS0807060504

First printing, January 2004

Dedication

For Beth, Anika, and Tobias Taylor-Cohen

and

Jim and Carol Giangola

and

Elaine Clow and Anthony Balog

Praise for *Voice User Interface Design*

"Mike Cohen is a giant in the field of speech technology. He and his co-authors bring years of valuable experience in VUI design to life in this new book. It is a must read for anyone designing user interfaces."

John Kelly
Editor-in-Chief
Speech Technology Magazine

"VUI design is a challenging combination of art, science, and process. The authors of ***Voice User Interface Design*** augment their extensive real-world experience with a thoughtful perspective on how to think about the process. And they get into the details necessary to deliver an effective voice interface. The book removes much of the mystery behind this new and critical discipline."

Bill Meisel
Publisher and Editor
Speech Recognition Update

"The strength of ***Voice User Interface Design*** over others in the area is its depth and grounding in the research literature. More importantly, it does not pretend to offer a cookbook to VUI design, but rather a set of well-informed design principles, and gives credit to the intelligence of the reader to apply these principles to specific problems."

Chris Schmandt
Principal Research Scientist
M.I.T. Media Lab

"Most current speech recognition systems can recognize a single word as well as, if not better than a human being. Yet building an effective voice system is still a significant challenge. ***Voice User Interface Design*** meets the challenge by providing the necessary background and steps to produce an effective VUI. Organized around a financial brokerage design example, the book is a comprehensive, yet approachable treatment of all that is necessary for building a successful voice application. From analyzing the differences between written and spoken language to prompt design and prosody planning, ***Voice User Interface Design*** should quickly become the standard reference for all involved in voice application design."

Harry M. Hersh
The Users Voice

About the Authors and Radio Rex

Michael Cohen cofounded Nuance Communications in 1994. He has served in a variety of roles at Nuance, including Vice President of Dialog R&D. He created the Nuance Professional Services team, which engages with customers to design and deploy applications, and led the group through the first 22 deployments of Nuance technology. He created the Dialog R&D group, which has been responsible for voice user interface research, for Nuance's natural language understanding technology (including Say Anything and Accuroute), and for designs of interfaces for products such as Voyager, the Nuance voice browser.

Before founding Nuance, Michael was at SRI International for more than 10 years. At SRI he led numerous projects in acoustic modeling and speech technology development. He led the SRI ATIS project, which combined speech recognition and natural language understanding technology to create an early spoken language understanding system.

Michael has published more than 70 papers and holds eight patents related to speech and VUI technology. He is a frequent speaker at conferences and industry trade shows. Michael is a consulting professor at Stanford University and serves on the board of directors of the Applied Voice Input/Output Society (AVIOS). He received his Ph.D. in computer science from UC Berkeley.

In his spare time, Michael composes and plays music. In the summer of 2000, his band, the Mike Cohen Sextet, performed at the Montreux Jazz Festival.

James Giangola considers himself an "industrial linguist," on a mission to apply principles of natural conversation, from prosody to the discourse level, to engineered dialogs. In addition to dialog design, his expertise includes speech synthesis, concatenation planning and production, and voice coaching. No matter the area of interface design, however, his overriding concern is to offer the user a familiar linguistic experience, and therefore one that is comfortable and comprehensible. James's design philosophy is not limited to English-language interfaces; he has also helped to create natural-sounding, persona-rich interfaces in French, Portuguese, German, and Japanese.

James holds linguistics degrees from Brown University, the Monterey Institute of International Studies, and U.C. San Diego, and he has 10 years of experience teaching languages at the high school and university level. He is also the author of *The Pronunciation of Brazilian Portuguese* (Munich: Lincom Europa, 2001).

James resides in Salvador, Brazil, where he maintains a linguistic consulting business for American and Brazilian companies needing help with interface design and brand naming.

Jennifer Balogh is a Speech Consultant at Nuance Communications, where she designs and evaluates interfaces for spoken language systems. She has worked on deployed applications for customers such as AT&T, Charles Schwab, and TD Waterhouse and has contributed to a number of Nuance products, including Nuance Call Steering, the Vocalizer TTS engine, the voice browser Voyager, and SpeechObjects. She has also conducted research on dialog design techniques that optimize user satisfaction and holds several patents.

Previously, she researched language disorders at the Aphasia Research Center at the Boston VA Hospital. She

also cofounded Phaedrus Internet Development, Inc., a Web solutions provider for corporations and medical institutions. She has presented at conferences such as CHI Conference on Human Factors in Computing Systems and CUNY Conference on Human Sentence Processing. She has also lectured for courses at Stanford University, UC San Diego, University of San Francisco, and University of San Diego. Jennifer has published several papers in journals such as *Brain and Language* and *International Journal of Speech Technology*. She received her Ph.D. in psychology from UC San Diego and holds a B.A. from Brandeis University.

Radio Rex, pictured on the cover of the book, was the first automated spoken language understanding system ever developed. Rex was first produced in 1911. Unlike many spoken language systems developed over the following eight decades, it was a commercial success! Radio Rex was a children's toy. Rex sat in his doghouse until the child said "Rex," at which point he would eagerly jump out. The technology behind Rex is briefly described at the beginning of Chapter 2.

Photo of Radio Rex courtesy of Hy Murveit. Rex himself courtesy of the private toy collection of Michael Cohen.

Preface

In the past decade, there has been an explosion in the creation and commercial deployment of voice user interfaces, especially for use over the telephone. Voice user interfaces (VUIs) use speech technology to provide callers with access to information, allow them to perform transactions, and support communications.

This proliferation is driven by a number of factors: customer dissatisfaction with touchtone interactions, a growing desire for mobile access, the need for enterprises to more effectively and inexpensively meet their customers' needs, and the development of speech technology that is finally robust and reliable enough to deliver effective and reliable interaction in well-defined domains.

At the beginning of this decade of growth (starting around 1994), the biggest obstacle that needed to be overcome was skepticism about the capabilities of the technology. Speech technology had been promised for decades and had disappointed many times. The enterprises that could potentially improve their customer service and save money, as well as the venture capital firms that could fund the early start-ups, needed proof points. Within a few years, many such proof points existed: Millions of phone calls every day were being handled successfully by speech technology. Although technology improvement will continue to play a key role in providing better experiences for end users, increased business value to enterprises, and new capabilities enabling new types of applications, it is no longer the key bottleneck to growth of the speech industry.

The biggest challenge now is the design of the user interface. There are too few practitioners who have the

knowledge and skills to create all the systems needed and to advance our understanding as new technology enables new capabilities. Current practitioners come from a wide variety of backgrounds: speech technology, user interface design, cognitive psychology, linguistics, and software development. All these fields have contributed to our current level of understanding of voice user interface design. In fact, the field has benefited substantially from the diversity of influences. However, the need to pull together information from diverse fields has made it difficult to codify and teach the rationale for design.

In this book we aim to offer in one place much of the background information needed for practitioners to design specific applications and to contribute to the advance of the field. We try to take a principled approach to deriving best practices, with the hope that designers will then have a basis for approaching new design situations and new technologies.

Organization of The Book

Given that our primary focus is to teach VUI design, we chose to organize it according to the design methodology we recommend. The parts are as follows:

Part I, introduction: Chapters 13 provide the necessary introductory material, including an overview of voice user interfaces and design issues, a description of the technology, and a high-level view of the design methodology we detail throughout the remainder of the book.

Part II, definition phase: Chapters 47 cover the definition phase of a project: discovering the requirements and making high-level design decisions that will guide the detailed design.

Part III, design phase: Chapters 814 cover the detailed design phase. Design principles are covered in detail, with many examples of how to apply them to real applications.

Part IV, realization phase: Chapters 1518 cover the realization phase: development, testing, and tuning. A number of issues that are unique to voice

user interface design, such as grammar development, are covered in detail.

Each part begins with a chapter covering the methodological details for that phase of design. Following that are a few chapters describing the design principles and approaches relevant to that phase. The final chapter of each section presents a design example.

Audience

This book is meant to address a variety of audiences:

- Practitioners: The primary audience is practitioners or those intending to become practitioners. We try to lay the groundwork so that beginners can understand all the material. The book should provide value to both experienced and inexperienced designers. Practitioners can benefit from reading all the chapters.

- Students of humancomputer interfaces: Students studying humancomputer interface design will find that VUIs have many things in common with other types of user interfaces. On the other hand, a number of issues and design approaches are unique to voice user interfaces. The entire book is useful to students, although Chapters 14, 6, 813, and 1516 stand out as most important.

- Business managers: Business managers making decisions about how speech technology can best meet the needs of their organizations will benefit most from Chapters 14 and 6.

- Project managers: Project managers who must understand the steps in designing and deploying an application will benefit most from Chapters 14, 68, 1415, and 18.

Web Site

Throughout the book we stress the value of *listening* to your interface rather than only looking at a written specification. We would be remiss if we did not provide a means by which readers could listen to the many examples presented. We have created a Web site (http://www.VUIDesign.org) that tracks the book and provides audio versions of the examples. We recommend following the Web site as you read.

You can listen to all examples marked with this icon in the left margin.

Acknowledgments

The accumulated knowledge in this book is the result of years of collaboration between the authors, Nuance Professional Services, the Nuance Dialog R&D team, the Nuance Speech R&D team, and Nuance Speech University. Many people have contributed to the accumulation of knowledge represented here, including Rajeev Agarwal, Sandra Ahlen, Gorm Amand, Kerry Bartlett, Mar Bergeron, Carol Bleyle, Wally Brill, Kathy Brown, Dan Burnett, Candace Cardinal, Ari Chanen, Philippe Charest, Hong Chen, Mike Coskey, Carol Curt, Jenny DeGroot, Greg Dehaan, Melissa Dougherty, Benoit Dumoulin, Kevin Erler, Lisa Falkson, Daniel Ferrero, Malan Gandhi, Alvaro Garcia, Frank Geck, Debajit Ghosh, Rebecca Nowlin Green, Leo Haasbroek, Mike Hochberg, Lauren Hodgson, Judee Humberg, Vitaly Iourtchenko, Eric Jackson, David James, Charles Jankowski, Kathy Karey, Karen Kaushansky, Brian Krause, Vijaya Kurada, Remi Kwan, Ruth Lang, Will Lavery, Dominic Lavoie, Nicole Leduc, Sophia Lee, Melanie Levasseur, Hank Liao, Ari Limon, Kristine Ma, Mike Maben, Marcelo Marsano, Muneeb Master, Jane Mather, Ian Menzies, Art Morgan, Bob Morgen, Idil Moskateel, Ali Mouline, Leslie Myers, Hy Murveit, Christine Nakatani, Claudio Pateras, Cathy Pearl, Doug Peters, Marco Petroni, Tony Rajakumar, Vijay Raman, Padma Ramesh, Lys Reddy, Dave Resnick, Keith Rolle, Eliot Rubinov, Marikka Rypa, Martha Senturia, Ben Shahshahani, Tony Sheeder, Shamitha Somashekar, Krishnan Srinivasan, Elizabeth Strand, Jared Strawderman, Brian Strope, Ann Thyme-Gobbel, Real Tremblay, Amy Ulug, Margaret Urban, Linda Waldon, Nino Walker, Trace Wax, Jim White, Todd Yampol, and Torsten Zeppenfeld.

A number of others at Nuance have provided encouragement for this project, including Steve Erlich, Ron

Croen, Chuck Berger, Matt Lennig, Bruce Dougherty, Paul Scott, and Doug Sharp. Tony Sheeder and Rebecca Nowlin Green gave talks at VWorld 2001 in which they presented a sample application that formed the basis for the sample application presented here. Harry Hersh, Jennifer Lai, Roger Chapman, Chris Schmandt, Nathaniel Borenstein, and a number of anonymous reviewers provided extensive feedback on an early draft that resulted in significant improvements.

The team at Addison-Wesley has been tremendously helpful and supportive in guiding us through this project, including Peter Gordon, Curt Johnson, Heather Mullane, John Fuller, Bernard Gaffney, Betsy Hardinger, Amy Fleischer, and Jacquelyn Doucette.

Finally, we thank our families for love, support, and tolerance. Beth Taylor (Mike Cohen's wife) provided significant editing help that made the manuscript far clearer and more readable.

Part I: Introduction

Chapter 1. Introduction to Voice User Interfaces

The following is a true account of the first experience of one of the authors (MC) deploying a commercial spoken language system:

> *The first application we ever deployed at Nuance was a voice-driven stock quote system for Charles Schwab & Company. The months leading up to the deployment were very exciting for me personally. I had spent more than ten years in the research labs at SRI developing spoken language technology. Finally, we had started a company to commercialize our technology. We had signed our first customer and were almost ready to deploy our first system. We were about to prove to the world that this technology had real valuethat it was ready for "prime time."*
>
> *I observed the early usability tests. I was behind the one-way mirror as subjects came into the test room. I watched them pick up the phone and use the system to get stock quotes and set up watch lists and watched as the experimenter questioned them about their experience.*
>
> *The first subject arrived. He was an 83-year-old man from San Francisco. He came in the door, hesitated, and said, "Oh no, I forgot my hearing aid!" Five seconds into the first real user test, and we were already hitting problems we had never anticipated in ten years of research!*

The test proceeded. The subject had a great deal of trouble using the system. He had difficulty hearing the prompts, was confused about what he could say, and got lost in the application. By the end of the experiment, he had never succeeded in setting up a watch list and never received a quote on a company he requested.

At that point, the experimenter asked the subject about his reactions to the application and whether he thought he would use such a system. He loved it! He described the experience as the first time in years he had been able to have a conversation with anyone so patient, so willing to repeat themselves, and so willing to talk with him and never get frustrated. He promised he would be calling this system every day!

This story illustrates the two key themes of this book:

1. Understanding basic human capabilities is key to the design of effective user interfaces.

2. Understanding the user's needs and goals, in the context of the business goals, is key to the design of successful applications.

The two themes differ in that the first addresses general human capabilities: understanding what is easy and what is hard for all (or most) people and how to exploit this knowledge to optimize design choices. (Here we are concerned with human cognitive capabilities and linguistic behavior rather than individual physical capabilities such as hearing, as suggested by the story.)

In contrast, the second theme specifically addresses the application at hand: what you need to understand about the intended users, the task, and the business goals in

order to focus the design on meeting user and business needs simultaneously. In the story, we could have considered the system test a success if merely providing an electronic conversational partner were an acceptable result. The test result did not, however, meet the business goal of providing value with respect to the handling of brokerage accounts.

In *The Humane Interface* (2000), Jef Raskin draws a distinction between user-centered design and human-centered design. He describes **user-centered design** as a design process focused on studying the task-related needs of the intended users of a specific application. He then goes on to emphasize the importance of **human-centered design**, which is focused on "making sure that the interface design accords with universal psychological facts." He argues that it is more important to understand the cognitive capabilities and limitations of humans in general and to apply that understanding to interface design problems.

In this book, we argue that both human-centered and user-centered design are key to the creation of successful user interfaces. We spend substantial time explaining the core principles of design based on an understanding of human cognitive capabilities and human linguistic behavior. However, we also examine in detail the methodologies by which you can integrate an understanding of the intended users, the application tasks, and the business goals of the system to create effective interfaces that provide successful and satisfying user experiences.

Our focus is entirely on *voice* user interfaces. Moreover, given the ubiquity of the telephone and the large number of spoken language systems currently being deployed for over-the-telephone use, we focus specifically on voice user interfaces designed for the phone.

1.1 What Is a Voice User Interface?

A **voice user interface** (or **VUI**) is what a person interacts with when communicating with a spoken language application. The elements of a VUI include prompts, grammars, and dialog logic (also referred to as call flow). The **prompts**, or **system messages**, are all the recordings or synthesized speech played to the user during the dialog. **Grammars** define the possible things callers can say in response to each prompt. The system can understand only those words, sentences, or phrases that are included in the grammar. The **dialog logic** defines the actions taken by the systemfor example, responding to what the caller has just said or reading out information retrieved from a database.[1]

[1] Note that when we use the word "dialog" throughout the book, we mean a spoken exchange of words and not "dialog" in the sense it is used by some software designers to refer to a box containing written words.

The following example is an interaction between a caller and a flight information application:

(1)

SYSTEM: Hello, and thanks for calling BlueSky Airlines. Our new automated system lets you *ask* for the flight information you need. For anything else, including reservations, just say "more options." Now do you know what the flight number is?

CALLER: No, I don't.

SYSTEM: No problem. What's the departure city?

CALLER: San Francisco.

. . .

In this application, a voice actor has prerecorded everything the system says. Following the prompt "Now do you know what the flight number is?" the system listens, using a grammar that accommodates inputs from the caller such as "No," "No, I don't," "Yes," "Yeah, it's flight two twenty seven," and so on. The dialog logic then decides what to do next, depending on the answerin this case, to prompt the

caller for the departure city. If the dialog succeeds, the system ultimately provides the desired flight information.

The methodologies and design principles for VUI design overlap substantially with those used for other types of user interface design. However, there are a number of characteristics of voice user interfaces that pose unique design challenges and opportunities. Two primary characteristics stand out: the modality is *auditory,* and the interaction is through *spoken language.*

1.1.1 Auditory Interfaces

An **auditory interface** is one that interacts with the user purely through soundtypically speech input from the user and speech output and nonspeech output from the system. **Nonspeech output** (often referred to as **nonverbal audio**, or **NVA**[2]) may include **earcons** (auditory icons, or sounds designed to communicate a specific meaning), background music, and environmental or other background sounds.

[2] A term introduced by Wally Brill.

Auditory interfaces present unique design challenges in that they depend on the ability to communicate with a user through **transient**, or **nonpersistent**, messages. The user hears something, and then it is gone. There is no screen to display information, instructions, or commands, as is the case with a visual Web interface, where items can be accessed over time at will. Users do not have the opportunity to review the system's output or state their wishes at their own pace. Rather, the pacing is largely controlled by the system.

The ephemeral nature of output in auditory interfaces places significant cognitive demands on the user. There are a number of guidelines you can use to make sure your interface designs do not overload the user, do not unduly challenge short-term memory or learning ability, and provide a means for the user to influence the pacing of the interaction. We cover these guidelines in Chapter 9.

Multimodal interfaces that combine speech with other modalities have the potential to mitigate some of these problems. Even a small screen, when combined effectively with a speech interface, can significantly reduce the cognitive demands on the user, thereby changing some of the design criteria and trade-offs.[3] However, given the immature state of the multimodal device industry and the large number of spoken language systems that are currently being designed and deployed for use in traditional telephony networks, this book focuses on purely auditory interfaces. Many of the same design principles, with appropriate refinement, can be applied to multimodal design. Consideration of multimodal interfaces that include speech will be left for a future volume.

[3] See Oviatt 1996 for a review of studies showing the complementary power of speech and other modalities.

Despite the challenges, auditory interfaces offer a number of unique design opportunities. People rely on their auditory systems for many levels of communication. Listeners derive semantic and other information not only from word choice and sentence structure but also from the way a message is delivered: from **prosody** (intonation, stress, and rhythm), voice quality, and other characteristics. By carefully choosing a voice actor (for recording the prompts the system will play to the user) and effectively coaching the way the prompts are spoken, you can help create a consistent system **persona**, or personality. This offers

opportunities for branding and for creating a user experience appropriate to the application and user population.[4] We discuss the crafting of a persona in Chapter 6.

[4] Clearly, other features such as the wording of prompts also play a role in persona creation.

Auditory interfaces offer an additional opportunity based on effective use of nonverbal audio. You can use earcons to deliver information (e.g., a sound indicating "voice mail has arrived") without interrupting the flow of the application. Distinctive sounds can be used to landmark different parts of an application, thus making it easier to navigate. Additionally, nonverbal audio such as background music and natural sounds can create an **auditory environment** for the user, thereby creating a unique **sound and feel** associated with a particular business or message. Designers, through effective use of nonverbal audio, can solve user interface problems in new ways as well as exploit opportunities for added value.

1.1.2 Spoken Language Interfaces

Voice user interfaces are unique in that they are based on spoken language. Spoken communication plays a big role in everyday life. From an early age, we spend a substantial portion of our waking hours engaged in conversations. An understanding of human-to-human conversation can be brought to bear to improve the conversations we design between humans and machines.

Humans share many conversational conventions, assumptions, and expectations that support spoken communication; some are universal, and others apply in specific language communities. These conventions,

assumptions, and expectations operate at many levels, from the pronunciation, meaning, and use of words to expectations about such things as turn-taking in conversations. Some expectations people bring to conversation are conscious, but many operate outside our awareness. Although largely unconscious, these shared expectations are key to effective communication.

An understanding of these shared expectations is essential to the design of a successful spoken language interface. Violation of expectations leads to interfaces that feel less comfortable, flow less easily, are more difficult to comprehend, and are more prone to induce errors. Effective leverage of shared expectations can lead to richer communication and streamlined interaction. In Chapters 10 and 11 we cover many of the expectations speakers bring to conversations and show you how to leverage that understanding in the design of VUIs.

Two other realities of spoken language have major impacts on VUI design choices and design methodology. First, humans learn spoken language implicitly at a very young age rather than through explicit education. In contrast, most other user interfaces depend on specific learned actions designed to accomplish the task at hand (e.g., choosing operations from a toolbar, dragging and dropping icons). Therefore, the VUI designer must work on the user's terms, with an understanding of the user's conversational conventions. As designers, we don't get to create the underlying elements of conversation.

Second, communication through language is largely an unconscious activity in the sense that speakers usually do not explicitly think about word choice, pronunciation, sentence structure, or even turn-taking. Their conscious attention is instead on the meaning of the message they wish to communicate. Therefore, despite the fact that we

all engage in conversation, designers are at risk for violating the user's conversational expectations unless they have some explicit knowledge of conversational structure, as discussed in Chapters 10 and 11. Furthermore, design approaches that make explicit the role of each prompt *in the conversational context in which it occurs* will maximize the ability of designers to bring to bear their own unconscious conversational skills as they design dialogs. The detailed design methodology discussed in Chapter 8 will show how this can be done.

1.2 Why Speech?

Given all the challenges we have cited, why bother with speech interfaces?

To answer that question, let's begin by looking at the Schwab deployment (the application referred to in the opening story). The Schwab application was a big success, from the points of view of the business and the end users. Schwab's original motivation for deploying a speech system for stock quotes was that a large percentage of callers to Schwab's touchtone system immediately pressed 0 on their touchtone keypad to get a live agent. Those callers were the primary target for the new speech system.

Most of those callers ended up very happy using an automated speech application, despite their reluctance to use the touchtone system. The primary reason was that they no longer had to enter stock symbols using complex two-keystroke sequences for each letter, as they did with the touchtone system. Now, they simply said the company name and got their quote. Many customers actually expressed a preference for talking to the speech system rather than a live agent. They were no longer hesitant to ask for a large number of quotes, or even to ask for the same company twice in the same call if they had forgotten the result or were curious to see whether there had been movement in the price.

Charles Schwab & Company was also pleased with the results. The firm saved a lot of money and was able to reassign live agents to deal with more complex user requests, and caller satisfaction was high. Schwab reinforced its reputation as a leader in providing innovations that bring value to their customers.

Since the Schwab deployment, there have been thousands more.[5] You can now talk to automated systems to trade stocks, check airline schedules and book flights, access your bank account, track packages you have shipped, rent a car, check on weather and traffic conditions, get driving directions, check bus schedules, find out what movies are playing nearby, and make restaurant reservations. You can have your own automated personal agent (discussed shortly) to handle your phone calls, calendar, e-mail, and voice mail. You can access **voice portals** that package a variety of voice services and, in some cases, personalize them to your needs. The list goes on and on, with new ways of providing information and transactions and facilitating human-to-human communications appearing every day.

[5] Counting systems from all the major vendors.

The companies deploying these systems are motivated by a number of factors:

- **Saving money:** Speech systems typically save companies significant amounts of money, usually paying for themselves within a few months. When companies replace touchtone services, abandonment rates typically go down substantially, and automation rates often rise dramatically. Decreased call durations have also been a factor, saving companies toll call fees.

- **Improving reach:** Companies want to be available to their customers everywhere (at home and when mobile) at all times (24x7x365). In some cases, a system is deployed to complement the self-service already provided over the Web. In this way, companies can reach customers who either don't have Web access or desire the same self-service access when they are away from their desk or in their car.

- **Extending a brand:** When you engage in spoken interaction, you get more than "just the facts." Speech communicates at many levels. As we listen to someone speak, we make judgments about numerous attributes: the speaker's emotional state, social class, trustworthiness, friendliness, and so on (Soukup 2000; Giles and Powesland 1975). If we design a speech application with these things in mind, we can connect with customers in new ways. The system becomes an extension of the company's brand and image. In effect, we can design the "ideal employee"with the desired voice, the desired personality traits, the desired mood, and the desired way of handling customer needs and problems. We can present that image reliably, phone call after phone call.

- **Solving new problems:** Many kinds of problems can be solved, or services offered, by speech applications that were simply impossible in the past. For examples, see the sidebar.

- **Increasing customer satisfaction:** Numerous surveys and deployment studies have shown high user satisfaction with speech systems.

From the point of view of the end user, there are many appealing advantages of using speech systems compared with other access modes:

- **Intuitive and efficient:** Spoken language systems draw on the user's innate language skills. Many tasks can be made simpler and more efficient than with touchtones. For example, in a travel application a caller may say things such as, "I wanna leave on June fifth" rather than enter an awkward and unintuitive

touchtone sequence (such as 0605) after hearing a long-winded instruction. Even simple things such as saying the name of your destination city, rather than spelling it on a touchtone keypad, bring significant convenience to end users.

- **Ubiquitous:** Telephones are ubiquitous. Many people carry cell phones. Spoken access makes systems available from anywhere, even when the user is far from the desktop.

- **Enjoyable:** A well-designed speech system can provide a user experience that is engaging and enjoyable, at the same time efficiently meeting the user's needs.

- **Hands-free, eyes-free:** Activities such as driving occupy the user's hands and eyes. Speech is an ideal solution for accessing services while engaged in such tasks. Furthermore, mobile devices generally have very poor user interfaces. Entering complex information is awkward with keypads, pointing devices, or handwriting recognition. Speech is the ideal solution.

New Ways to Solve Old Problems

Speech applications let organizations solve problems in new ways. One example is call routing. Many companies want to combine their various access phone numbers into one convenient point of entry to serve all the needs of their customers. But the resulting set of items from which a caller must choose (such as the names of the various available services) is often huge. Furthermore, the set of choices often does not lend itself to an intuitive

menu hierarchy that callers can easily navigate. As a result, it has not been possible to create such systems with touchtone technology (or even with earlier generations of speech technology). Recent advances in **natural language understanding** (technology that figures out the intended meaning of a string of words) have made it possible to let callers state their needs in their own words and then be automatically forwarded to the appropriate service. For example, a system recently deployed by Sprint PCS, with 40 different "routes" or services, allows callers to say things such as, "I just had a call dropped, and I'd like to get a credit for it," "Yes, I want that store in Rockville, on Rockville Pike," and "I'm having a problem with my voice mailit's locked up on my phone, and I can't get into it to get my messages." (These were transcribed from actual calls.)

Another new application of voice technology is the personal agent. Most of us are not rich enough to hire someone to sit beside us in our car, office, and home and accompany us as we walk down the street, all the while handling our phone calls, reading us our e-mail, managing our calendar, and carrying the entire world's white pages and yellow pages. A number of companies now offer automated **personal agents**, which use speech technology to provide such capabilities on the user's phone.

1.3 Where Do We Go from Here?

The primary goal of this book is to teach a way of thinking about VUI design. We focus on the underpinnings of the craft, basic design principles, and design methodology. Along the way, we present many examples and show you how to apply the principles in specific instances to address specific design issues. In the end, you should have a toolbox of techniques you can apply to many design situations.

By itself, however, a toolbox can be dangerous. You cannot create an effective design that engages callers, meets their needs, and satisfies business goals by blindly applying rules of thumb. As you will see, there are two recurring issues that arise with every design decision at every step of the design process: consideration of context and understanding of the end user. By **context** we mean everything from high-level constraints such as business goals to low-level considerations such as how the wording of a prompt works in the context of the immediately preceding prompt and response. Understanding your end users includes assessing how they will react, what their needs are, and what their preconceptions are at each point in time. These considerations must play into every design decision you make, whether it is a decision to apply a principle, use a specific technique, modify it, or invent something new.

Furthermore, technology is developing rapidly, enabling new, sometimes unanticipated possibilities. Leveraging new technological capabilities will create new challenges for the VUI designer. You must understand the fundamentals and design principles when you approach new situations. As a case in point, recent technology advances have made it possible to implement call routing applications of the type

mentioned earlier. To achieve success with these systems, we had to learn new approaches to crafting prompts and new ways of creating the appropriate mental model for the caller. A **mental model** is the caller's internal model of how to interact with the system (Norman 2002; Preece, Rogers, and Sharp 2002). (The details of call router design are discussed in Chapter 12.)

As you will see, a number of fields of knowledge contribute to the fundamentals of VUI design. In this book, four key areas emerge. Three of them we have already touched on:

- **Human cognitive capabilities:** Understanding the design challenges you face when dealing with purely auditory interfaces
- **Human conversational language use:** Uncovering the unconscious expectations and assumptions that humans bring to all conversations
- **Methodology:** Understanding the methodological principles and goals (such as user-centered design) and applying that understanding to the development of a detailed methodology that enables us to design effective systems that meet user needs and business goals

The fourth area is technology. To make effective design decisions, you need some understanding of speech technology and must bring it to bear. Our description of technology gives you the necessary background to understand how to leverage the possibilities of the technology and compensate for its limitations. You will see the need for technology understanding when we discuss basic dialog strategy choices as well as design decisions for

optimizing accuracy and recovering from errors, developing grammars, and tuning performance.

This book is organized according to the design methodology we recommend. We and many others have applied this methodology to hundreds of successful deployments.

Each major part of the book discusses one major phase of design. The first chapter in each part describes the relevant steps of the methodology. We follow that with a number of chapters discussing design principles and issues related to that phase of design. Each part ends with a chapter containing a detailed example, showing concretely how to apply the process steps and principles for that part.

Before we dive into the methodology, a little more introductory material is needed. Chapter 2 provides an overview of speech technology and introduces some of the technical issues that figure into VUI design decisions. Chapter 3 introduces the general methodology that is covered in detail in the remainder of the book.

Chapter 2. Overview of Spoken Language Technology

To make effective design decisions, you need to understand the strengths, weaknesses, and possibilities of the underlying technology. This chapter introduces speech technology at the level of understanding required by VUI designers and discusses how this understanding should come into play when you make design decisions.

Even though speech recognition technology is only now becoming commonplace, it has been with us for almost a century. The first success story was actually a children's toy, produced around 1911, called Radio Rex (pictured on the book cover). This toy was a small celluloid dog sitting in a tin-roofed doghouse. Inside the doghouse, an electromagnet held a spring-loaded lever. There was a metal bridge in the circuit that supplied power to the electromagnet. The bridge was tuned to have a resonance around 500 cycles per second, which coincides with a resonance in the vowel sound for the word "Rex." When the dog's name was called, the energy in the vowel caused the bridge to vibrate, breaking the circuit. When the magnet shut off, the spring-loaded lever was released and sent the dog flying out of the house in response to his owner's call.

Rex was a commercial success despite its simplicity and one-word vocabulary. (That was lucky, given that more capable technology was not available for another 40 years!) Today's commercial systems require a bit more sophistication, as described in this chapter.

2.1 Architecture of a Spoken Language System

We begin with a high-level overview of the elements of a spoken language system and then look in more detail at what happens inside the recognizer. Finally, we discuss two speech technologies that are sometimes used in speech applications: text-to-speech synthesis and speaker verification.

2.1.1 Elements of a Spoken Language System

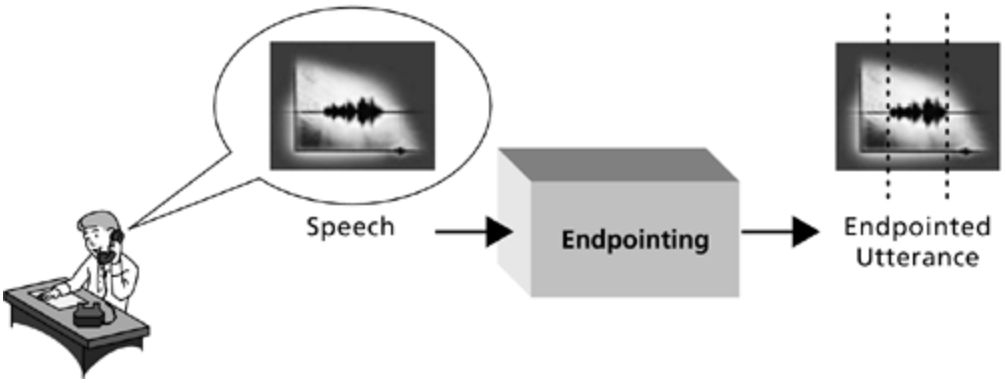
When you design a voice user interface, you are, in effect, defining a set of potential conversations between a person and a machine. Each of those conversations consists of a series of interchanges, with the machine and the human taking turns speaking. To meet the needs of the human user, the machine must "understand" what the user says, perform any necessary computations or transactions, and respond in a way that moves the conversation forward and meets the caller's goals.

Figure 2-1 shows the basic architecture of a spoken language system. It consists of a series of processing modules designed to take speech input (the user's **utterance**), understand it, perform any necessary computations and transactions, and respond appropriately. Following the response, the system waits for the next utterance from the user and repeats the sequence until the call has ended. In this section, we describe the activities of the various processing modules.

Figure 2-1. The architecture of a spoken language understanding system.

Figure 2-2 shows the first step, **endpointing**, which means detecting the beginning and end of speech. The system listens for the caller's input. The endpointer determines when the **waveform**, representing the vibrations of the caller's spoken utterance, has begun and then listens for a sufficiently long silence to indicate that the caller has finished speaking. The waveform is packaged and sent to the next processing module, which performs feature extraction.

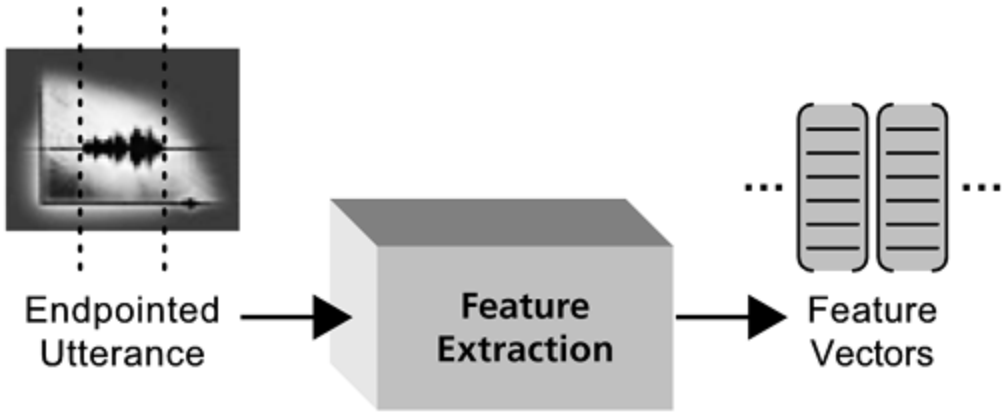
Figure 2-2. The endpointer determines where the speech waveform begins and ends.



The **feature extraction** module (Figure 2-3) transforms the endpointed utterance into a sequence of feature vectors. A **feature vector** is a list of numbers representing measurable characteristics of the speech that are useful for recognition. The numbers typically represent characteristics of the speech related to the amount of energy at various

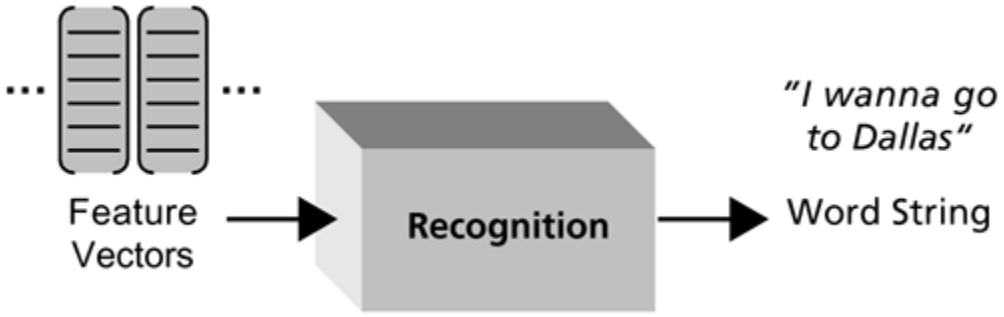
frequencies. Typical systems divide the endpointed waveform into a sequence of feature vectors, with one vector for each small time period (e.g., one feature vector for each successive 10-millisecond segment of the speech).

Figure 2-3. The feature extractor transforms the endpointed utterance into a sequence of feature vectors, which represent features of the utterance as it occurs through time.



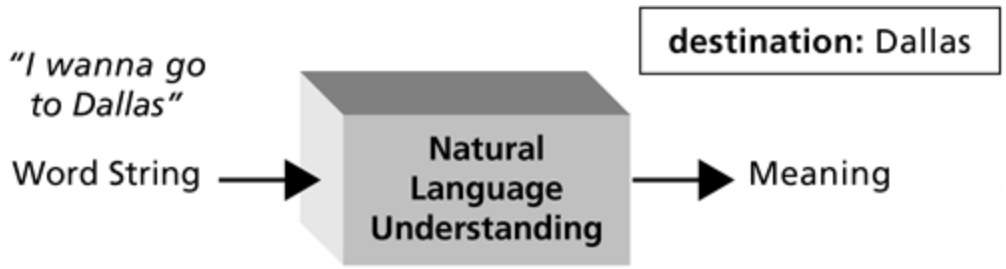
The **recognizer** (Figure 2-4) uses the sequence of feature vectors to determine the words that were spoken by the caller. This process is described in more detail later in this chapter.

Figure 2-4. The recognizer uses the string of feature vectors to determine what words were spoken by the caller.



Following recognition, natural language understanding occurs. The job of **natural language understanding** is to assign a meaning to the words that were spoken. There are a number of ways to represent the meaning. A common representation is as a set of slots with values. A **slot** is defined for each item of information that is relevant to the application. For example, relevant information for an air travel application might include the caller's origin city, destination city, date of travel, and preferred departure time. The natural language understanding system analyzes the word string passed from the recognizer and assigns **values** to the appropriate slots. For example, in Figure 2-5, the caller said, "I wanna go to Dallas," communicating that the destination city is Dallas. The natural language understanding system sets the value of the <destination> slot to "Dallas."

Figure 2-5. The natural language understanding module assigns values to slots to represent the meaning of the word string. Here, the <destination> slot was assigned the value "Dallas."



After the meaning of the caller's input has been determined, the dialog manager (Figure 2-6) takes over. The **dialog manager** determines what the system does next. There are many possibilities. The system may take an action, such as accessing a database (e.g., of available flights), play back information to the caller (e.g., list the flights that fulfill the caller's needs), perform a transaction (e.g., book a flight), or play a prompt requesting more information from the caller (e.g., "OK, what day do you want to leave?"). In current commercial systems, dialog management is the result of an explicit program written to control the flow of the application (often with special tools provided by platform vendors, or in special-purpose languages such as VoiceXML). Some research systems (Jurafsky and Martin 2000) provide a generic dialog management module that can be configured for a particular application.

Figure 2-6. The dialog manager determines what the system should do next.

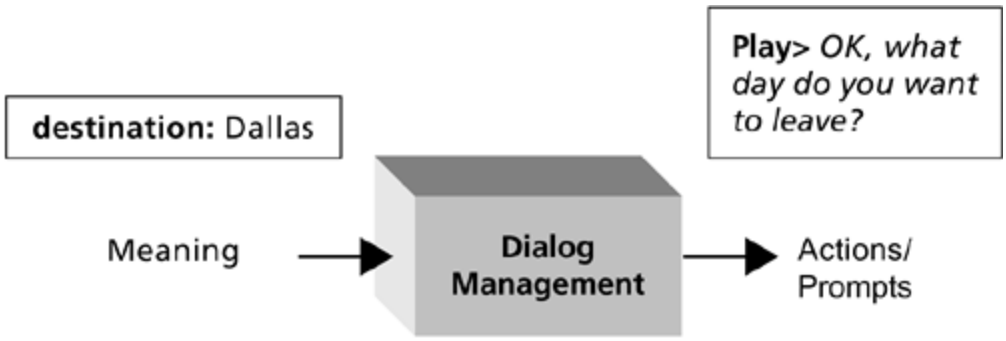
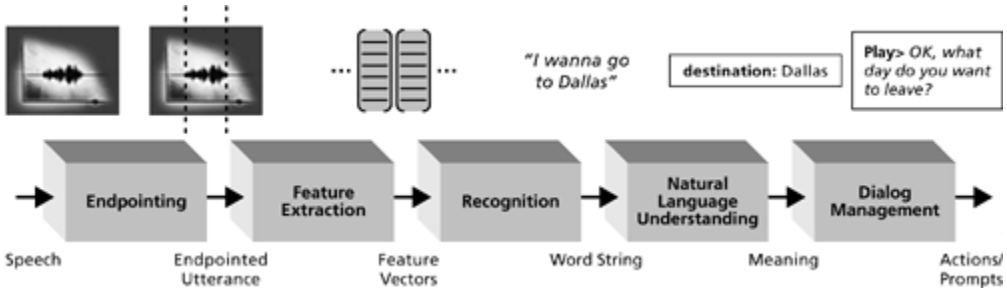


Figure 2-7 illustrates the entire process, showing the inputs and outputs for each module. The entire processing sequence runs for each input utterance from the caller until the call ends.

Figure 2-7. The processing sequence for handling one spoken input from a caller.

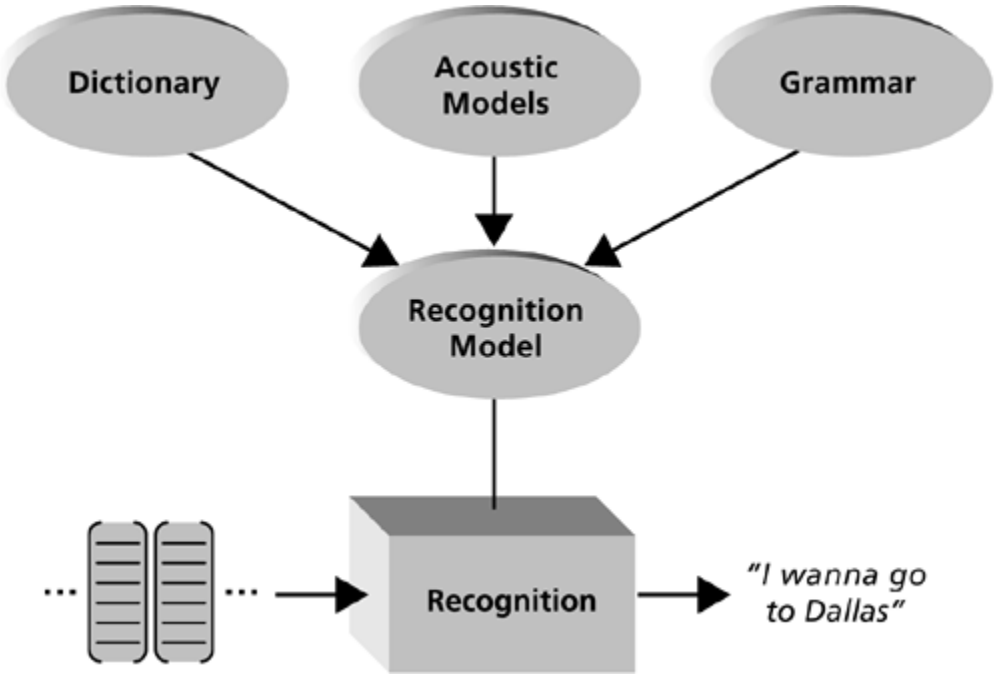


2.1.2 Recognition

Now let's look in more detail at what happens inside the recognizer (see Figure 2-8). Remember, the job of the recognizer is to figure out the string of words that was spoken, given the sequence of feature vectors. It does this by searching the **recognition model**, which represents all the word strings the caller can say, along with their possible pronunciations. The recognizer searches all those possibilities to see which one best matches the sequence of feature vectors. Then it outputs the best-matching word string. To create the recognition model, three things are needed: acoustic models, a dictionary, and a grammar.

Figure 2-8. The recognizer searches the recognition model to find the best-matching word string. The recognition model is built

from the acoustic models, dictionary, and grammar.



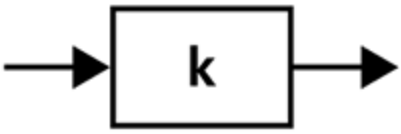
Acoustic Models

An **acoustic model** is the recognizer's internal representation of the pronunciation of each possible **phoneme**, or basic sound,[1] in the language. For example, in English, one acoustic model may represent the sound commonly associated with the letter *K* (see Figure 2-9). Acoustic models for most current systems are created by a **training** process. Many examples of spoken sentences and phrases, labeled with the word string actually spoken, are fed to the system. Based on the set of examples, a statistical model for each phoneme is created, representing the variety of ways it may be pronounced.[2] The features that are modeled are the same as those in the feature vectors created by the feature extraction module.

[1] More precisely, phonemes are abstract classes capturing the minimal distinctive sounds in a languagethat is, the sounds that can differentiate words.

[2] Modern systems actually create models for many versions of *K* and all the other phonemes in order to capture the effects of the different contexts in which they may occur. However, VUI designers don't need to understand those details.

Figure 2-9. *Acoustic model for the sound of* K.



Note that in this book the individual phonemes are indicated by a set of symbols called the Computer Phonetic Alphabet (CPA). There is one CPA symbol for each phoneme. Appendix A defines the entire set of CPA symbols for English.

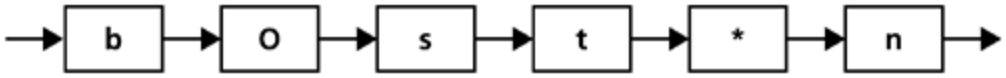
Dictionary

A **dictionary** is a list of words and their pronunciations. The pronunciation indicates to the recognizer which acoustic models to string together to create a word model. Figure 2-10 shows a dictionary for a system with a two-word vocabulary: "Dallas" and "Boston." Figure 2-11 shows the word model created for the word "Boston."

Figure 2-10. Dictionary showing the pronunciation of "Dallas" and "Boston" in CPA.

Dallas	d a l * s
Boston	b O s t * n

Figure 2-11. Word model for "Boston," consisting of a string of acoustic models, one for each basic phoneme making up the word, as defined by the dictionary.



Many words have more than one possible pronunciation because of regional accents, stylistic variations, rate of speech (how fast the speaker is talking), and soon. The dictionary can contain multiple entries for a word to handle different pronunciations. Figure 2-12 shows two dictionary entries to handle the two common pronunciations of "economics."

Figure 2-12. Dictionary showing two possible pronunciations for "economics."

economics	E k * n A m I k s
economics	i k * n A m I k s

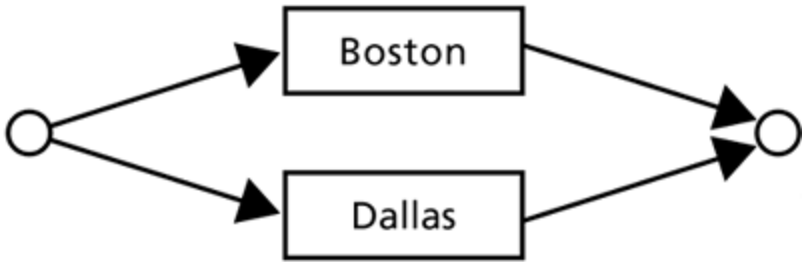
Grammar

The **grammar** is the definition of all the things the caller can say to the system and be understood. It includes a definition of all possible strings of words the recognizer can handle, along with the rules for associating a meaning with those strings (e.g., by filling slots). Different grammars may be **active** at different times during the conversation (the active grammar is the one currently being used for recognition and natural language understanding). In this

section, we are concerned only with the recognition grammarthat part of the grammar that defines all the possible word strings.

Figure 2-13 shows a simple grammar that can recognize the words "Boston" and "Dallas." A more realistic grammar would of course include many more items; a real application would have a longer list of cities. In addition, callers typically include many **filler words** and **filler phrases** when they speak. A caller is likely to say things such as, "I want to go to Boston" rather than simply "Boston." All the filler words and phrases must be included in the grammar. However, to simplify our discussion, for now we consider a system that recognizes only "Boston" or "Dallas," with no fillers.

Figure 2-13. A simple grammar. This would allow recognition of only two possible inputs from the caller: "Boston" or "Dallas."



It is important to distinguish two types of grammars. A **rule-based grammar** is created by writing a set of explicit rules that completely define the grammar. Alternatively, a **statistical language model** (**SLM**) is a statistical grammar that is created automatically from examples. To develop a statistical language model, you collect a lot of speech from callers and transcribe what they said (notate the exact word strings spoken). You then feed the data to a

system that creates the grammar by computing the probability of words occurring in a given context.

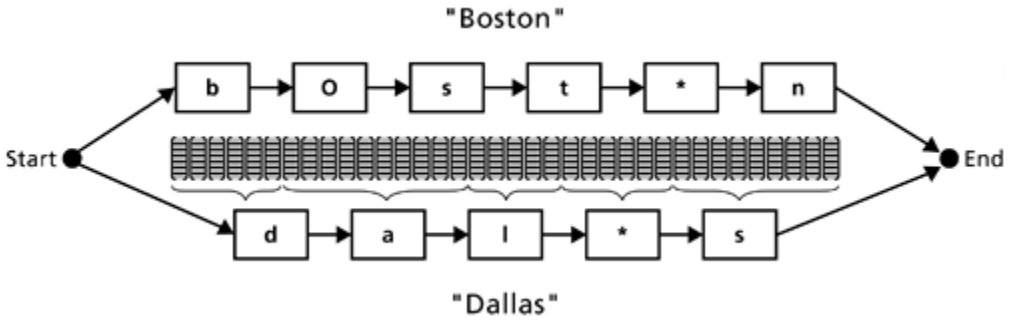
For example, the system assigns the probability of a particular word occurring next, given the word (or the last two words) just spoken. A statistical language model would result in a more complex version of the diagram shown in Figure 2-13, with more possible word combinations, and would define probabilities associated with the transitions from one word to the next.

As you will see later, the choice of using a rule-based grammar or a statistical language model is critical. The advantage of an SLM is that it generally allows callers far greater flexibility in what they can saythe words they choose and how they put them together. Statistical language models typically are used when you want to allow callers more "natural" language or "free form" speech (such as, "Um, I really need to get to Dallas next Tuesday, and I need to arrive by about three p.m."). The choice of grammar type has vast implications for VUI design. It affects every aspect of your design, from the wording of prompts to dialog strategy, from call flow to the organization of the complete application. Both types of grammars can be included in different parts of the same application. The choice of grammar type is discussed in detail in Chapter 5.

Recognition Search

Figure 2-14 shows what you get when you put it all together. For each word in the grammar, the appropriate word model is inserted, thereby creating the recognition model shown earlier in Figure 2-8.

Figure 2-14. Recognition model that can recognize the words "Dallas" and "Boston." The feature vectors are shown aligned with their matching acoustic models along the best-matching path.



The result is a representation of the entire set of word strings that can be recognized, as defined in the grammar. Each word incorporates a word model, as defined in the dictionary, consisting of the appropriate sequence of acoustic models. This representation, then, includes all the possible word strings and all their possible pronunciations. This is the representation through which the recognizer makes its search.

Recognition consists of comparing the possible paths through the recognition model with the sequence of feature vectors and finding the best match. The recognizer returns the path that, given the model, is most likely to have generated the feature vectors observed in the caller's utterance. This **best-matching path** is associated with a particular word or string of words; that is what is recognized.

In Figure 2-14, imagine that the endpointed waveform is 0.36 seconds long. Given that each feature vector is for a 10-millisecond segment of speech, the entire input utterance is represented as a sequence of 36 feature

vectors. Figure 2-14 shows a possible match of feature vectors to acoustic models along the best-matching path. If the closeness of the match of feature vectors to the acoustic models is best along this path, the recognizer returns the result "Dallas."

Three other elements of recognition have an impact on VUI design decisions: confidence measures, N-best processing, and barge-in.

Confidence Measures

Most commercial recognition systems, in addition to returning the **recognition hypothesis** (the best-matching path found in the recognition search), also return a confidence measure. The **confidence measure** is some type of quantitative measure of how confident the recognizer is that it came up with the right answer. It is based on a measure of closeness between the feature vectors representing the input signal (the caller's speech) and the best-matching path.

VUI designers can use confidence measures in a number of ways. For example, if confidence is low, you can immediately conduct an explicit confirmation of the recognition result (e.g., "You want to fly to Dallasis that correct?"). Various ways of using confidence measures are discussed at a number of points during the detailed design process described in Part III (Chapters 814).

N-Best Processing

Most commercial recognizers can also run in **N-best** mode. Rather than return a single, best-match result, the system

returns a number of results (the N best-matching paths) along with the confidence measure for each. Given an N-best list of possible results, the system can bring to bear other knowledge to make a choice. For example, if the two best-matching recognition results were "I wanna go to Boston" and "I wanna go to Austin," but the caller disconfirmed Boston earlier in the dialog, you can design the system to pass over Boston and select Austin, thereby not repeating the mistake. There are numerous ways to take advantage of N-best processing, and they are also covered in Part III.

Barge-in

Barge-in is a feature that allows callers to interrupt a prompt and provide their response before the prompt has finished playing. When barge-in is enabled, the recognizer starts listening as soon as the prompt begins rather than when it ends. If the caller begins to speak while the prompt is still playing, the prompt is cut off and the recognizer processes the input.

There are many excellent references you can use if you want to learn more about spoken language technology. For signal processing and feature extraction, see Gold and Morgan (2000). For speech recognition, see Jelinek (1997), Rabiner (1989), Rabiner and Juang (1993), Weintraub et al. (1989), Cohen (1991), and Cohen, Rivlin, and Bratt (1995). For natural language understanding, see Allen (1995), Manning and Schutze (1999), and Jackson et al. (1991). For dialog management, see Chu-Carroll and Nickerson (2000), Chu-Carroll and Brown (1998), Rudnicky and Xu (1999), and Seneff and Polifroni (2000).

2.1.3 Other Speech Technologies

Two other speech technologies may prove useful for some applications: text-to-speech synthesis and speaker verification.

Text-to-Speech Synthesis

Text-to-speech (**TTS**) technology synthesizes speech from text. Although TTS does not yet replicate the quality of recorded human speech, it has improved a great deal in recent years. We typically use recorded human speech to play prompts and messages to callers. However, certain applications, such as e-mail readers and news readers, have very dynamic data to which callers wish to listen. In those cases, given that the text of the messages cannot be predicted, you can use TTS technology to create the output speech.

The primary measures of the quality of synthesized speech are as follows:

- **Intelligibility:** How well the listener can understand what is said
- **Naturalness:** How much the synthesized speech sounds like real human speech
- **Accuracy:** The correctness of what is synthesized (e.g., making the correct choice between "doctor" and "drive" when the input text includes the string "Dr.")
- **Listenability:** How well users tolerate extended listening without fatigue

In recent years, there have been tremendous advances in the naturalness of synthesized speech, largely because of the refinement of an approach called **concatenative synthesis**. A concatenative synthesizer uses a large database of segments of recorded speech. The output signal is created by concatenating a sequence of these prerecorded segments. Signal processing is applied to achieve the appropriate timing and intonation contour and to smooth out the boundaries between segments so that the concatenation splices are not audible.

Chapter 11 reviews guidelines for using TTS and for combining TTS with recorded speech. It discusses a number of approaches for optimizing the quality of the output. To gain a deeper understanding of TTS technology, you can consult a number of excellent sources, including Edgington et al. (1998a and 1998b), Page and Breen (1998), Dutoit (1997), and van Santen et al. (1997).

Speaker Verification

Speaker verification technology is used to verify that a caller is the person he or she claims to be. It has been deployed for a variety of applications, sometimes as part of the login process for a spoken language application. In some applications, speaker verification has been used to replace personal identification numbers (PINs) so that customers no longer need to remember them. In other cases, it has been used to provide secure access to account and credit card information. One application applies speaker verification to home incarceration, verifying that home parolees are, indeed, at home.

Before callers can be verified, they must be **enrolled** in the system. Enrollment involves the collection of a small amount of the caller's speech, which is used to build a

model of the person's voice (sometimes referred to as a **voiceprint, voice template**, or **voice model**). On future calls, callers first make an identity claim by, for example, entering an account number. The voice is then compared to both the stored model and an **imposter model** (a model created from a combination of other speakers). A decision to accept or reject the caller is made based on how well the input speech matches each of those models.

There are a number of good overviews of speaker verification technology, including Reynolds and Heck (2001), Campbell (1997), and Furui (1996).

2.2 The Impact of Speech Technology on Design Decisions

Why have we taken the time to describe speech technology in such detail? As you will see, this understanding influences design decisions throughout the process of creating a voice user interface. In this section, we review a few of the concrete ways you can apply this knowledge to design.

Three primary areas stand out:

1. **Performance challenges:** If you understand what the technology can handle easily and what affects performance, you can best leverage the technology's strengths and design around its weaknesses.

2. **Problem solving:** Problems (such as recognition errors) can arise while a caller is interacting with an application. An understanding of the nature of potential problems, and how to detect them, will help you design dialog strategies for rapid and graceful recovery, as well as help you tune the system to avoid problems.

3. **Definition files:** The designer must create or modify a number of definition files (e.g., grammars, dictionaries) that are used by the recognition and natural language understanding modules. Your knowledge of how these definitions figure into the recognition and understanding process will guide you in creating them.

2.2.1 Performance Challenges

Three of the biggest challenges for recognition performance are ambiguity, limited acoustic information, and noise.

Ambiguity

As you observed in Figure 2-14, the recognizer's ability to correctly determine the spoken word string depends on finding a better match (between feature vectors and acoustic models) along the correct path than any other path. The recognizer's biggest enemy is similar-sounding paths because they can easily be confused.

A classic example is the pair of sentences "Wreck a nice beach" and "Recognize speech." Although the word strings look very different, when spoken quickly they can sound the same. A more practical example of a difficult recognition task is the alphabet. Many letters (such as *B* and *D*) rhyme with each other. Add to this the fact that the duration of the initial consonant when we say "bee" or "dee" is quite short (i.e., most of the feature vectors are for the vowel, which does not provide help distinguishing the two words), and you have a challenging recognition problem. Chapter 13 covers design guidelines for domains that have recognition ambiguities.

In general, as the vocabulary and grammar get larger, the potential for ambiguity increases. However, you cannot simply keep your grammar small or delete ambiguous items. The grammar must cover the things callers are expected to say. What's more, the worst ambiguities are often not obvious: You can judge only by actual system performance on real data. Chapter 16 provides guidelines for creating and tuning grammars.

Limited Acoustic Information

In general, shorter words and phrases are harder to recognize than longer ones. Longer words and phrases provide more acoustic information that can help in differentiating paths through the recognition model.

Let's consider an example from nationwide directory assistance. Such applications cover an extremely long list of cities from all across the country. To ease the task of recognizing cities, these applications typically begin by asking for "city and state" rather than asking only for the city name. As a result, the system gets more acoustic information to help it differentiate paths through the recognition model associated with different cities.

To see why this is helpful, consider the problem of differentiating the spoken city names "Boston" and "Austin." The only acoustic information that can distinguish between these two cities is the *B* in Boston. In general, the realization of *B*'s are quite short. If we imagine the utterance was 0.75 seconds long (therefore including 75 feature vectors[3]), it is quite likely that the *B*, if it was there, was represented by 3 or 4 of the 75 vectors; in other words, it played a small role in the overall score. Alternatively, if the caller said "Austin," any slight distortion at the beginning of the word (such as a lip smack) could easily have matched the *B* model. Clearly, differentiating "Boston, Massachusetts" from "Austin, Texas" is far easier than differentiating "Boston" from "Austin."

[3] Assuming one feature vector for every 10-millisecond segment of speech.

Noise

Noise and distortion can come from numerous sources, including environmental noise and distortions created by the phone line. Noise makes recognition harder. It adds a

random factor to the feature vectors so that they no longer represent the caller's actual speech as accurately. They may not then match as closely the acoustic models along the correct path. The noise may also mask important features for matching. In general, anything that changes the feature vectors so that they are less like the data used to train the acoustic models will make recognition less accurate.

In short, all the recognition challenges we have raisedambiguity, limited acoustic information, and noisecan be understood with reference to Figure 2-14. The better you can differentiate between the correct path and all possible others, the better the recognition performance. The more feature vectors that match well to acoustic models along the correct path, the easier an utterance is to recognize. You can apply this basic concept to help understand and ameliorate recognition challenges in new situations.

2.2.2 Problem Solving

Recognition does not always succeed. When it fails, there are a number of messages the recognizer may return to the application. For example, a **reject** indicates that the recognizer did not find any path that matched the input well; even the best path had a very low confidence measure. A reject may have a number of causes, such as noise or the caller saying something not included in the grammar.

A **no-speech timeout** indicates that the endpointer did not detect speech. This could indicate that the caller said nothing, or it may be caused by poorly tuned endpointer parameters. For example, maybe it was set to listen for only one second before giving up, and the caller hesitated longer than that before beginning to speak.

To design effective dialog strategies to recover from problems, you must understand all the possible failure messages from the recognizer and what they may indicate about the problem. (These messages are covered in Chapter 13.) During the tuning stage, you must be able to track down the underlying cause of observed problems so that you find the right solutions (e.g., tune the endpointing parameters versus reprompt the caller).

2.2.3 Definition Files

Figure 2-8 shows three files used by the recognition model: the acoustic models, the dictionary, and the grammar. As you have seen, these three files play crucial roles in defining how the recognizer works. In addition, there is usually some type of configuration file that sets various parameters of the process (e.g., how long the endpointer waits for a sound before it gives up). In this section, we discuss the role of the VUI designer with respect to each of these definition files.

Grammar

Much of what has been written in the past on VUI design mainly treats the *output* side of the interface: everything that controls what the system will say to the caller (e.g., prompts and call flow). However, when you deploy a system, you must also design the *input* side, defining everything that the caller can say to the system (the grammar). These are the two sides of the conversationinput and output, caller speech and system speech.

Grammar is perhaps the place where VUI design and technology are most intimately entwined. The output

cannot be designed in isolation from the input. In fact, there is a close relationship between what a prompt says and what the caller ends up saying to the system. Many people have noted the correlation between the wording of a prompt and the words chosen by the caller in response (Baber 1997). Even when designers of prompts and call flow are lucky enough to have someone else write the grammar, they still must understand the issues of grammar: how prompt wording and dialog strategy determine grammar needs, how grammar possibilities may constrain the design of prompts, and so on. As discussed earlier, even the choice of grammar type (rule-based versus a statistical language model) will have a significant impact on VUI design decisions. You can make that choice only by combining insights about your end users, application, and business needs with an understanding of the technology.

Chapter 5 discusses the choice of grammar type, and Chapter 12 examines the implications of that choice on the details of prompt design. Chapter 16 discusses the development and tuning of grammars.

Dictionary

Most speech technology vendors supply large dictionaries with their products, covering the vast majority of words and pronunciations in the languages they handle. For most applications, the designer need not touch the dictionary. Chances are, the words in your grammar will already be there, with correct pronunciations.

In a few cases, however, it is appropriate to supplement the dictionary. Some applications may have unusual words that must be recognized. For example, a system that provides driving directions may need to recognize all the street

names in a big city. Some street names are unusual words and may not be in the default dictionary. They must be added. (Some technology vendors provide tools to automatically determine the pronunciation of new words based on their spelling.) In other cases, callers may use an unexpected pronunciation for some words, warranting the addition of pronunciations to the dictionary. These pronunciation additions should be made only during the tuning stage, based on observations of real usage patterns. Chapter 15 discusses how to tune dictionary entries.

Acoustic Models

It is almost never necessary to alter the acoustic models supplied by the vendor. In general, the vendor has trained a set of acoustic models on a large data set, covering many domains. The default acoustic models should work well out of the box. In fact, vendors do not usually provide a means for designers or developers to alter (retrain) acoustic models. That is a good thing. Done wrongly, altering acoustic models is more likely to hurt than help performance.

Some vendors supply a mechanism for **acoustic model adaptation**, which automatically refines the acoustic models for the application and domain without the need for intervention by the designer or developer of the application. If offered, that is the best way to ensure optimal performance.

Configuration Files

There may be other definition files that are used to control the recognition and understanding process. For example,

you may choose the confidence level at which the recognizer will reject the input rather than return an answer. You may also choose parameters for the endpointer (e.g., how long it should listen before timing out). In general, default values are supplied for all these parameters, and they need to be changed only in special cases. If, for example, the system is about to execute a stock transaction, you want to be very sure the caller has confirmed it. In that case, you may want to set the reject rate higher.

2.3 Conclusion

We have looked at the basics of modern-day speech technology and have seen examples of how an understanding of the technology will enhance VUI design decisions. Many details of those decisions and design issues will come up throughout the remainder of the book. However, the most important goal for this chapter is to give you a level of understanding that will provide a basis for thinking about new design situations: those that may be missed in this book, and those challenges that may arise in the future.

We are now ready to introduce the design methodology that will be discussed in detail throughout the remainder of the book.

Chapter 3. Overview of the Methodology

Voice user interface design is both an art and a science. It is an art in the sense that it is a design activity requiring the application of careful craftsmanship and aesthetic judgment. It is a science in the sense that it calls on a scientific understanding of human cognitive capabilities and linguistic behaviors. It is also an engineering practice in which you can apply appropriate metrics and testing in all phases of design.

We begin by looking at the key underlying methodological principles that drive many of the details we cover throughout the book. Following that, we lay out the primary steps of the methodology. Finally, we discuss issues that arise when you apply the methodology to real-world projects (with real-world constraints).

3.1 Methodological Principles

Five key principles underlie much of the methodology. An understanding of these principles will help you more effectively apply the methodology to projects with real-world constraints. Furthermore, the principles will give you a basis for extending the methodology as new technologies create new opportunities and challenges.

The five key principles are as follows:

1. **End-user input:** Inform design decisions with end-user input.
2. **Integrated business and user needs:** Find solutions that combine business goals and user goals.
3. **Thorough early work:** Avoid expensive downstream changes by focusing on thorough work in the early definition and design stages.
4. **Conversational design:** Move the design experience close to the user experience so that the designer can experience design elements in their appropriate conversational context.
5. **Context:** Make all design decisions with appropriate consideration of context.

Let's discuss each principle in detail.

3.1.1 User Input

The designer of a user interface is, in some ways, the worst person to evaluate its usability. Designers know what they intended when crafting the interface. In the case of VUIs, designers know exactly the intended message behind every prompt and system action and tend not to see any ambiguities that may exist. They know how to use the interface, however difficult it may be for others to learn.

This is a problem in the design of all user interfaces. With VUIs, the medium of communication with the user is spoken language. Language functions in context, and effective communication depends on the user's life experience and world knowledge as well as innate and learned linguistic skills. As a result, VUI designers are at great risk that they will not even notice the confusions and complexities that must be faced by users of the interfaces they design

The only solution is to gather input from users to validate and refine design decisions. This idea is the key concept underlying the user-centered design paradigm, which has become the cornerstone of modern user interface design (see Norman and Draper 1986, Rubin 1994, and Cohen 2001). The type of user input you need varies as you progress through the phases of design. For example, early in the process you may focus on understanding users' mental model when performing tasks of the type you intend to enable. In the midst of detailed design, you may run user tests of specific elements to see how easy they are to use.

With each phase of the methodology we present, we discuss the ways of gathering user input appropriate to that phase. As you move from phase to phase, your questions will change, and the techniques for collecting user input will change.

3.1.2 Integrated Business and User Needs

The opening story in Chapter 1, about the first test subject for the Schwab system, showed an extreme example of a system that met a user need without meeting business goals. Delighted though the subject was in finding a patient, conversational partner, no brokerage business was conducted! Although designers must be advocates for end-user needs, they must always work with a clear understanding of the business context.

Many steps in our methodology, especially in the early phases, are focused on achieving that understanding. Many dimensions of the business must be clarified, including, for example, understanding how the application will fit with all the business's other means of touching its customers, its branding and image needs, and so on. Many design decisions are influenced by the business context. For example, business goals influence design trade-offs, branding needs influence creation of a system persona or voice, and so on. The job of the designer is to meet the needs of the business and its users simultaneously.

3.1.3 Thorough Early Work

In the software industry, many years of experience have taught developers and project managers that the most efficient way to deploy successful software is to follow a process that includes thorough analysis of requirements and high-level functional design before developers dive into the details of specific software modules. Skipping the early steps has led to many software disasters (McConnell 1996).

Thorough early work is equally important for VUI design, for many of the same reasons. If you have a comprehensive understanding of the applicationfrom both the user and the business point of viewbefore you dive into the design details, you greatly reduce the risk of making poor design decisions. It is far more expensive to make design changes later in the design process or, worse yet, after implementation is complete. A detailed understanding, up-front, of end users' needs will help you make effective design choices. Early user testing can provide early design guidance and avoid the risk of extensive changes late in the process.

Additionally, making high-level design decisions up-front facilitates consistency throughout the design, which is one of the keys to usability. For example, if you choose the persona or character of the application early, you can use it to guide the crafting of prompts. In this way, you will create a consistent character that is reinforced throughout the application.

The methodology we describe has a significant focus on the early stages (see Part II, Definition Phase). Our experience has proven that thorough early work greatly reduces the risk of poor usability and the need for reworking designs.

3.1.4 Conversational Design

Imagine painting a picture without ever seeing it. For example, suppose you are using software that lets you type commands such as "Place a three-inch red square in the upper-left corner" but never displays the image you are creating. Even an experienced painter would not have a good result using this approach. Whether you are creating a work of art, a consumer product, or a user interface, to

create an effective design you need to envision the end product as you shape the elements.

Often, VUI designers get buried in the details of the mechanics of specific dialog states and lose sight of the larger context of their design decisions (the conversations). [1] The elements of the dialog state (e.g., initial prompt, prompt to play in case of recognition reject, prompt to play in case of timeout) are often crafted one after another, without regard to the actual conversational flow in which they would occur. The result is language that sounds stilted, unnatural, and less comprehensible than the designer expects.

[1] A **dialog state** is roughly defined as a single interchange between the caller and the system (i.e., a prompt followed by a response), along with all the special handling that supports the interchange. Examples of special handling include what the system says to get the caller back on track if there is a problem such as a recognition reject, a timeout, a caller request for help, and so on. Chapter 8 has a more detailed definition.

Chapter 8 describes a number of design approaches that help you consider design elements in their conversational context, thus bringing your experience closer to that of the user. The result should be more naturally flowing and more comprehensible dialog. We all have tremendous skill and lots of practice engaging in conversation, even if we are not consciously aware of the underlying structural elements involved. Given that much "knowledge" of conversational structure is unconscious, techniques that help you imagine conversations as you design will allow you to bring to bear more of your unconscious knowledge of conversation as you craft prompts and make other design decisions.

3.1.5 Context

All design decisions must be considered in context. The relevant context ranges from high-level considerations,

such as business goals and user needs, to low-level details such as the role of a particular prompt in moving the conversation forward.

Let's look at the contextual considerations involved in a single design decision. Consider a travel application that is in the pilot stage and is experiencing lots of recognition rejects in the `GetDate` dialog state, where the system asks the caller the date of travel. Imagine that the pilot data show lots of unexpected fillers (e.g., "Um, well I need to get there Tuesday for a Wednesday meeting") as well as unexpected formats for specifying the date (e.g., "I wanna leave a week from Tuesday"). We decide to use the following new prompt when a recognition reject occurs in the `GetDate` dialog state: "Sorry, I didn't get that. Please say the date of travel. For example, you could say 'March tenth' or 'June sixth.'" The contextual considerations of this prompt include the following:

- **Application need context:** We know at that point that there is a recognition reject, so we need to reprompt for the date.

- **User context:** The data show that the users are using fillers and date formats that the grammar cannot handle. Although we can add more variations to the grammar, we see enough variation that we decide it would hard to achieve high grammar coverage, so we decide to provide more guidance to the user.

- **Language-use context:** We know from experience and published work that callers tend to pattern their utterances after the prompt, especially if examples are provided.

- **Discourse context:** Let's consider the role of this new prompt in the conversation. The discourse goal has moved from information gathering (soliciting the trip details such as origin and destination) to a need to *re*solicit information. Not only is this a new role in the discourse, but also it is an unexpected shift. The caller probably expects to continue his or her travel plan, but the system needs to backtrack. We want to signal the change to the caller, both to avoid confusion and to encourage help. "Sorry, I didn't get that" signals the caller that we are changing mode and explains why.

- **Persona (branding) context:** Assuming that the company image calls for a friendly, helpful persona, we want to signal the new discourse goal and be very clear about how the user is to respond. If we were seeking to create a different persona, we might have chosen different words. For example, a more formal persona might have started with "I'm sorry, I didn't understand."

3.2 Steps of the Methodology

This section describes a six-step methodology for designing and deploying voice applications. Our emphasis is on those steps typically performed by a VUI designer, although we also cover other steps in deployment.

As a designer, even if you are lucky enough to be part of a team in which others are responsible for some phases such as software development, you must have an understanding of what is involved in the entire process. Such an understanding will facilitate teamwork and avoid design decisions that cause problems for those involved in other parts of the project.

The six steps are as follows:

1. Requirements definition
2. High-level design
3. Detailed design
4. Development
5. Testing
6. Tuning

3.2.1 Requirements Definition

The purpose of requirements definition is to achieve a detailed understanding of the application. The goal goes beyond understanding the features and desired

functionality. You must also understand the target users of the application: What motivates them to use the system? What expectations do they bring? What are the typical usage scenarios? For example, will callers usually use the system while driving? What are the most common reasons for calling?

You also must understand the business context: the primary goal for deploying the system, how the system functions in concert with other ways that the company communicates with its customers, the image and branding goals, and so on. You also must understand the full application context, including the other systems it integrates with and all possible failure modes.

A thorough understanding of requirements not only helps you understand the features you will design but also helps you make appropriate trade-offs, design the system persona, and create a usable system for the target audience. One output of the requirements process is a set of **success metrics**: performance attributes that can be measured to assess success and identify areas for tuning and improvement.

3.2.2 High-Level Design

A high-level design encapsulates what has been learned from the requirements phase in a concrete form that creates the framework for the detailed design. Additionally, by making high-level design decisions up-front before diving into the multitude of application details, you can achieve a consistency and unity of structure that would never emerge from the details without planning.

A high-level design includes a number of decisions. You determine the basic dialog structure (e.g., simple menu

versus complex natural language), grammar needs (rule-based versus SLM), use of nonverbal audio, and system persona. Chapter 4 covers the details and approaches to both requirements definition and high-level design.

3.2.3 Detailed Design

For the VUI designer, the detailed design is the phase that receives the bulk of the effort. The detailed design includes the complete specification of the call flow and all the prompts. You specify every detail of every application feature, and you design dialogs to handle all possible scenarios and problems (e.g., recognition errors, rejects, system problems, and caller requests to speak to an operator). You define ways to offer callers help and instruction when needed, to confirm recognition results when necessary, and to support callers when they change their minds and need to start over or redo tasks. You must test your design decisions on subjects who are representative of users, typically with iterative usability studies, beginning early in the detailed design process. If you're planning to use nonverbal audio, you must design the specific sounds.

Much of this book (Chapters 814) covers the detailed design process. We present a set of fundamental design principles and show how you can apply them in concrete ways. Additionally, we present examples of designs for many of the basic types of actions that happen in dialogs (e.g., confirmations and recovery from rejects).

3.2.4 Development

The development phase includes implementation of the call flow as software, specification of grammars, voice recording and audio production, and creation of interfaces to backend databases, Web services, and other software systems that interact with the application. In some cases, you implement the software in proprietary languages that run on particular **interactive voice response** (**IVR**) platforms, often using tools provided by the IVR vendor. Increasingly, systems are implemented using VoiceXML, a markup language in the same vein as HTML (the language typically used to design Web pages). VoiceXML is quickly becoming a standard. Chapter 15 discusses VoiceXML in detail.

You develop a grammar by specifying expected inputs from callers: all the words, phrases, and sentences callers are commonly expected to say. Your approach will depend on whether you're creating a rule-based or a statistical grammar. Chapter 16 discusses grammar development in detail.

3.2.5 Testing

The testing we refer to here happens after development is complete and before deployment or pilot rollout. You run a number of types of tests at this stage, each with a different goal. Some tests are run to make sure the implementation faithfully follows the design specification. Other tests help find good initial values for recognition parameters and make sure that recognition performance is reasonable before the system is exposed to the public. Also, you typically run **evaluative usability tests** to validate that the system meets basic usability goals. Evaluative usability testing may also help find problems that could not be detected before the availability of a fully implemented

system (e.g., timing problems due to poorly chosen endpointing parameters or latencies).

3.2.6 Tuning

Before full system rollout, you typically run pilot tests using the fully implemented system. In pilot mode, the system is deployed to a subset of the caller population. This is your first opportunity to measure the system's performance as it handles real callers using the system to perform the real tasks for which it was designed.

You collect the pilot data (including all the audio data from calls to the system) along with data on system behavior. Then you use the data to measure recognition performance, tune recognition parameters, improve the grammar's coverage (add the things spoken by real callers that were missed in the original grammar), and tune the performance of the VUI. The latter includes finding dialog states that have problems such as high hang-up rates, and tracking down and fixing the problems. For example, it may be necessary to improve the wording of a prompt so that callers speak "out of grammar" less often. If you're using a statistical grammar, you will add the pilot data to the grammar training set.

You often continue data collection and tuning after full system rollout. This is especially true if you're using a statistical grammar, because larger and more task-specific training data are likely to improve performance.

3.3 Applying the Methodology to Real-World Applications

This six-step process can be considered the "ideal" case. There are a number of important considerations to keep in mind when you apply this process to projects in the real world. First, phases often partially overlap, and you need to iterate not only within a phase but also between phases. Second, real-world projects always have both budget constraints and tight deadlines driven by business needs, requiring you to plan carefully, set priorities, and execute efficiently.

3.3.1 Coordination of Phases

As we've mentioned, within each of the six project phases, a certain amount of iteration is to be expected. For example, during detailed design, usability testing may reveal issues that lead to design changes that, in turn, lead to further usability testing. Similarly, there can be iteration between the phases as well. For example, measurements made during the tuning phase may lead to changes in the wording of a prompt or additions to the grammar.

The six phases can be categorized into three major goals: definition, design, and realization.

- **Definition** (including requirements definition and high-level design): The goal is to understand all application needs and create a design framework that encapsulates those needs.

- **Design** (including detailed design): The goal is to flesh out and specify all design details.

- **Realization** (including development, testing, and tuning): The goal is to turn the design into a working, deployed system.

The organization of the remainder of the book reflects these three goals. Part II covers definition, Part III covers design, and Part IV covers realization. Each part describes the relevant parts of the methodology, covers details on the related design issues and design principles, and provides a sample application to show the methodology in action.

3.3.2 Dealing with Real-World Budget and Time Constraints

All projects have budget constraints, and all projects have a schedule that presses teams to accelerate all phases. Much of that pressure falls on activities such as requirements gathering and usability testing, given that many managers don't see the importance of those activities for achieving the desired application goals. Part of the designer's job is to convince managers and companies deploying systems of the importance of thorough early work and a focus on user needs. Another part of your job is to structure those activities so that they are efficient and appropriate in scope to the real needs of the project.

Throughout the book, we discuss a variety of ways to achieve the goals of any particular task and present criteria for deciding which approach to use. For example, when you're designing an interface for a company trying to

achieve a strong, differentiated brand, it may be worth a significant effort to define a persona that exemplifies the brand. In other cases, a quick and simple persona definition may be adequate, just to ensure consistency in prompting style throughout the application.

This chapter emphasizes the need to understand underlying principles and motivations behind the methodology. That understanding will help you adjust the various tasks as needed, given project realities, and help you make effective trade-offs. By the same token, an understanding of the value of each task will help you argue for time and resources when the investment is warranted.

As the designer, you are responsible for the quality of the design and for achieving the business and user goals. Your knowledge about the best-case methodology is invaluable in evaluating the likely consequences of decisions regarding the allocation of time and resources. On the one hand, you are an advocate for the users, doing what it takes to meet their needs. On the other hand, you must bear in mind that the budget and time constraints are driven by business realities.

3.4 Conclusion

This brings us to the end of the introductory material. We are now ready to dive into the details, stepping through the entire process of creating a voice application and understanding core design principles and how to apply them.

Part II: Definition Phase: Requirements Gathering and High-Level Design

Chapter 4. Requirements and High-Level Design Methodology

Together, the requirements gathering and high-level design phases create a concrete definition of the application and set the stage for the detailed design. If the requirements you define aren't clear, it significantly raises the risk that the application will fail or that you will waste resources and time making major changes after development or even after deployment. This chapter lays out the steps to achieve a thorough definition of requirements and describes how to turn that into a concrete, high-level definition of the application.

4.1 Requirements Definition

Designers and developers often think of requirements as a detailed list of the application's features and functionality. Although that is one goal of requirements definition, it is not the only one. You must understand and document other things in order to create a successful application. The goals of requirements definition include the following:

- Understanding the business goals and context
- Understanding the users
- Understanding the application

Once you understand these elements, you can define a set of metrics to measure application success and to provide guidance during design, development, and tuning.

4.1.1 Understanding the Business

A number of business issues must be understood:

- **The business model:** What is the primary business motivation for developing the system (e.g., to reduce costs, create new services)? What are the secondary motivations? Is the system replacing an existing system (e.g., touchtone)? If so, why?
- **The corporate context:** How does this product fit with other products the company offers? Are there

multiple ways its customers can accomplish the same goals (e.g., on the Web as well as over the phone)? If so, in what way is this system supposed to complement the other systems (e.g., extend the organization's reach to mobile)? How does this application fit with all of the organization's other customer touchpoints?

- **The corporate image:** What image is the company trying to project? What are its brand attributes? How does the company portray itself in other media (e.g., Web, TV ads, touchtone systems)? How does this application figure into its overall branding efforts? Keep in mind that many companies are not aware of the branding and image-creating opportunity speech systems provide. You should be prepared to educate them. Make sure you have contact with those who are responsible for brand and image.

- **The competitive environment:** What products will this system compete with? What other methods do users have to accomplish the same goals?

- **Schedule and rollout plans:** How critical is time to market? (This may provide guidance on the size of the feature set.) How will customers be introduced to the system? Will they receive instructions in the mail? Will there be special incentives for them to start using the service? Will the service be rolled out in a series of phases with increasing functionality?

Answers to these questions can provide significant guidance during design and can help you make wise design trade-offs. For example, if the main goal of the business is to improve customer satisfaction, you might provide more information in the dialog explaining how to get to a live

agent than if the business's main goal is to increase automation rates.

Once the business goals are understood, metrics can be defined that can be used to judge overall success. Such metrics can guide efforts during tuning and clarify priorities that will influence trade-offs during design.

Metrics may include task completion (how many callers successfully complete their task), user satisfaction (how happy callers are with their experience), and accuracy. Many other measures are possible, depending on application goals. For example, you might measure call durations, the number of dialog steps to complete tasks, or longitudinal results such as the proportion of customers who repeatedly call into the system over the following few months.

It is important to avoid the mistake of delaying the start of pilot testing on real customers pending the achievement of ambitious metrics. Analysis of pilot data is invaluable in achieving high performance.

The primary means of establishing the business requirements is through meetings with the appropriate people from the marketing group at the deploying company. However, you must supplement that with other sources of information. In the following sections, we describe techniques you can use to answer the business questions.

Evaluating Other Company Systems and Customer Touchpoints

Most companies touch their customers in a variety of ways. Some are one-way (not interactive)for example, television,

radio, and newspaper advertising. Others may provide customers with means of accessing services, such as Web sites, touchtone systems, or live agents who provide information, transactions, or technical support.

Before meeting with company representatives, it will be worthwhile to familiarize yourself with the variety of the company's customer touchpoints. Explore its Web site. Call its touchtone system. Find an excuse to talk to a customer service representative. Pay attention not only to the functionality offered in each case but also to the look and feel. How does the organization reinforce its brand with each type of media? Additionally, look at its TV, radio, and newspaper ads. What are the image and message it is trying to communicate?

If you investigate these touchpoints before meeting with company personnel, you will be well equipped with questions and information that can make your meetings more productive and informative.

Meeting with Company Personnel

Meeting with marketing and other personnel at the deploying company will be a key means of understanding the business goals and context. To make sure that these meetings are productive and that you are getting to meet with the most appropriate people, send a list of questions and goals in advance. Make sure you meet not only with the people (probably in marketing) who are responsible for defining functionality but also with people involved in branding efforts. It can be useful to meet with someone responsible for training customer service representatives to understand the attitude and style the company tries to instill in its live agents.

Discuss all the issues mentioned in the earlier list. Additionally, ask for a brief description of the brand attributes. Then step through all the media used to reach customers, and find out how those brand attributes are manifested in each one.

Evaluating Competitive Systems

If other companies have deployed systems similar to the targeted application, you should evaluate them. At the very least, call them and exercise the systems thoroughly. It may even be worth conducting usability studies of competitive systems in order to improve on them and learn from what others have done. (Of course, time and money are always important factors in deciding the extent of such research.)

You should also evaluate other ways users can accomplish the same tasks. For example, if the target application is a fully automated directory assistance application, evaluate existing directory assistance applications that use live operators.

4.1.2 Understanding the Users

The population of expected callers to the system must be understood from two perspectives. First, you must understand caller profiles: Who will call the system? What are their characteristics and needs? Second, you must understand the usage profile: How will people use the system?

Caller profiles include the following elements:

- **The set of target customers:** Who are the target customers? Is a particular demographic group targeted? What is known about that group that is relevant to the application? Is the target customer set segmented? Are different classes of service planned for different segments?

- **Dialect or jargon:** Will these callers be from one industry and use industry-specific jargon? Are they from a specific dialect (geographic) region? Is the application focused on a particular age group or social group that has its own lingo?

- **Level of technological sophistication:** How comfortable are the callers with technology? Will most of the callers be Internet users? Will most of them have experience talking to automated systems?

- **Caller's state of mind:** Will this system be used in time-critical or mission-critical environments? Are callers seeking to manage their money? Looking for information? Looking for entertainment?

- **Caller's self-image:** Does this system target callers who have a particular self-image (e.g., tech-savvy, busy businessperson, on-the-go, relaxed, young and hip)?

- **Caller's mental model:** Part of the designer's job is to create an appropriate mental model in the mind of the caller. When callers initially interact with a system, they start with a mental model driven by past experience with similar systems and the ways they have performed the same or similar tasks in the past (including nonautomated ways). The goal, at this stage, is to understand the mental model callers bring with them

when they first experience this system. That will help you craft a mental model that most effectively takes advantage of the caller's existing mind-set.

- **Current user view of the brand:** How do users currently view the company's brand and image?

Often, a business has multiple user populations. A tricky design challenge is to create an application that is suitable for all types of callers. Because of the cost involved in creating different applications for each user population, you usually must create one design that caters to multiple groups. If possible, try to gather statistics on how many of the callers will fall into each of the subgroups so that you have the necessary information to help you make decisions about design trade-offs.

Knowing who is going to use the system is only part of the story. You also must understand the when, where, why, and how. Usage profiles include the following elements:

- **Primary versus secondary usage scenarios:** What is the primary reason that people will use the system? Understanding this will help you optimize menus, error messages, help messages, and so on. How often will each feature will be used?

- **One-time versus repeated use:** Will this system be used repeatedly by the same callers, or should each caller be treated as if it is his or her first call?

- **Level of attention:** Will the caller's primary attention be elsewhere while using the system (e.g., is this a system typically used while driving)? If so, you must give careful consideration to cognitive load issues (see

Chapter 9) and to features that accommodate sporadic lapses in attention.

- **Channel and environment:** Is the application focused primarily on mobile users? Will it usually be accessed by cell phone? Will it be accessed in noisy environments (e.g., hands-free in the car)?

- **Voluntary versus involuntary use:** Are the callers expecting to interact with a live agent and involuntarily talking to an automated system? Have they knowingly subscribed to an automated service?

- **Integration with other systems:** Will users of this system be expected to use other systems in concert with it, such as a Web interface to set preferences or call lists for a voice-activated dialer? Will users of this system sometimes do the same thing with other systems (e.g., use the Web when at their desktop, call this system when traveling)?

Clearly, discussions with the marketing department of the deploying company will be one important way to understand its users. However, it is important to also gather user information in other ways. The company may not see its users totally objectively, and it won't necessarily have all the answers to the kinds of questions and issues you must understand in order to optimize a VUI design. There are a number of approaches available. Some of these (e.g., observational studies) can be quick yet informative. Others (e.g., surveys and focus groups) can take more time but are warranted in certain cases.

Meeting with Company Personnel

As we suggested earlier, prepare for the meeting with company personnel by providing questions in advance, and make sure that the appropriate people will be there. The marketing group should be able to provide you with demographic profiles, information about industry-specific terminology (if, for example, it is a system for employees), and expectations of usage scenarios. Marketers also should be able to provide statistical data to predict the frequency of usage of the various features (e.g., from usage statistics of other systems or live agent calls or from statistics about the customer base). Many companies have accumulated a substantial amount of data about their customers, including their buying habits, lifestyles, and so on. All such data that they are willing to share can be of value to you.

Observational Studies

An **observational study** involves observing users performing the task that will be facilitated by the proposed application (Nielsen 1993; Preece et al. 2002). For example, if the application is for booking airline flights, an observational study might involve listening in on calls between travelers and travel agents. This approach can provide great insight into the mental model people bring to such tasks. You can observe their concerns, issues, and the ways they typically perform the task. For example, do travelers most often describe their trip based on departure time or arrival time? Do they prefer to get a long list of possible flights, or to consider a single flight and then ask questions about particular flight parameters (e.g., "Are there any later flights?")? Are their biggest concerns pricing, schedule, airline, type of airplane, or something else?

Observational studies create an early opportunity to get real data about motivated callers truly engaged in

performing the task. As you will see, many of the approaches to testing during the design process use subjects who are assigned specific tasks to performfor example, "Call this system and book a flight from New York to Boston for next Tuesday." Although there are many things to learn from such studies, they do suffer from the lack of realism: The caller is not actually planning a flight. The next opportunity to get real data is not until the system is designed, built, and ready for pilot testing with real callers.

Interviewing Customer Service Representatives

Another approach is to interview customer service representativesin the example just mentioned, the travel agents. Service reps can often tell you a great deal about what customers ask about, what problems come up, typical confusions and misunderstandings, and so on. This is not a substitute for observing real calls. There is nothing as valuable as direct observation of real data, removing the filters that others put on it. However, it can be worthwhile to complement observational studies with interviews of customer service agents.

There are a number of approaches to gathering data directly from users, including focus groups, individual interviews, and surveys, covered in the next three sections. These three approaches have the advantage of direct user contact. However, they all depend on the ability of users to report accurately on their behaviors, reactions, and needs, and users are often imprecise or even wrong when reporting on their own behavior and needs. It is useful to combine one of these techniques with other approaches that directly observe or measure user behavior.

Focus Groups

A focus group is a moderated discussion with groups of four to ten participants (Nielsen 1993; McClelland and Brigham, 1990). Successful moderation of a focus group takes considerable skill. The moderator must keep the discussion on track, focusing on a predetermined set of questions and issues. At the same time, the moderator must keep the discussion flowing relatively freely, allowing for new ideas and directions to emerge. He or she must be skilled enough to identify opportunities for pursuing issues in greater detail and must be able to elicit input from all participants rather than let a few people dominate the session. It is a good idea to start with open-ended questions and then delve into greater detail. A skilled moderator crafts questions carefully to avoid biasing participants' answers. If possible, it is good to run sessions with several different groups.

One advantage of a focus group is that some participants will be encouraged to speak about their own usage patterns and preferences in new ways in response to what others in the group say. A disadvantage of a focus group is "groupthink": Some participants may be less than candid or may be influenced by what others say.

The most typical way to run a focus group is to gather in a room the participants, moderator, and possibly observers (e.g., designers and developers of the system). However, a focus group can also be run over the telephone. Despite the advantages of a live session (e.g., the ability to read body language), an over-the-telephone session is cheaper and makes it easier to achieve a representative and balanced sample of participants. Moderating a focus group over the phone takes special skill and works best with no more than four participants.

Individual Interviews

An alternative to focus groups is a series of individual interviews. The goals may be similar to those of a focus group. The advantage is that the interviewee is not influenced by opinions expressed by others. It may also be easier to delve into greater depth on particular issues that emerge.

Interviews can be conducted either over the telephone or live. In general, they are not much harder to run over the phone than live. Running them over the phone has the advantages of reduced expense and improved ease in reaching a geographically dispersed group of people.

Surveys

A survey consists of a set of questions typically sent through the mail or e-mail. The question format may be multiple choice, check boxes, or brief fill-ins. A survey can be an inexpensive and effective way to collect data from a large number of subjects, although issues are typically pursued in far less depth than with other approaches. The answers solicited must be very brief. There is no way to elicit elaboration on any issues, and no way to be sure that participants clearly understand all the questions.

4.1.3 Understanding the Application

The most obvious part of requirements definition is fleshing out all the application details. All tasks and features must be defined completely. Specific elements include the following:

- **Tasks and subtasks:** Describe in detail all the tasks and subtasks of the application. This includes a complete description of all information supplied by the caller to the system and all information supplied by the system to the caller.

- **Task complexity:** Analyze the complexity of each task from the point of view of the caller. How much does the caller need to learn in order to use the application? Will a large set of commands be necessary? How much will the caller need to learn just to start using the system? What is the caller already assumed to know? Answers to these questions will help you determine which approaches to use for teaching and instructing users.

- **Recognition challenges:** Identify those tasks that are likely to be challenging in terms of recognition accuracy. Examples of challenging recognition problems include spelling and alphabetics, very long lists of items (many thousands), long number strings (because the whole thing is wrong if a single digit is wrong), and so on. For those tasks with challenging recognition, look for sources of leverage that can help. For example, if a long alphanumeric account number must be recognized, find out whether there is a special structure to the number (such as a checksum digit) or whether it is feasible to test the N-best recognition results against a database of existing account numbers. Chapter 13 covers these and other approaches to optimizing accuracy.

- **Application environment:** The environment in which the application operates includes hardware and software platforms, computer telephony integrations, and databases. It may also include other software systems such as fax-back or credit card verification.

The interfaces to these systems may place constraints on the VUI design. For example, if the application must access a database in order to return information to the caller, find out the expected **database latency** (delay in returning data). If latencies are likely to be long, the application must handle it. For example, you might change the order of tasks so that the system can do useful work while waiting for a database response, or you might play a latency sound or message. Furthermore, you must understand all the failure modes for each of the systems so that you can design appropriate dialogs for each one.

- **Other user interface technologies:** What technologies, in addition to speech understanding and prompt playback, will be part of the user interface? Will the system use speaker verification, text-to-speech synthesis, and the like? How will these impact the design? For example, if speaker verification is used, will this system be responsible for gathering voiceprints? Will it be combined with other means of user verification such as PINs?

Evaluating Other Company Systems

If the system will replace or complement an existing system, it is a good idea to evaluate the existing system to help define all tasks. For example, speech systems are often implemented to replace touchtone systems. Analysis of the touchtone system can provide a detailed definition of all tasks and features. Keep in mind that the goal of this analysis is not to duplicate the user interface (an effective speech interface is typically very different from a touchtone interface with the same functionality) but rather to understand all the tasks and features.

Meeting with Company Personnel

Marketing personnel should be able to provide you with detailed information on the system's functionality. In addition, meet with developers and integrators, or look at the appropriate documentation, to get a thorough definition of all the behaviors and failure modes of systems with which the application integrates.

4.2 High-Level Design

Once the requirements are understood, there is one more step before detailed design can begin: high-level design. The **high-level design** is often brief, but it plays a crucial role in achieving the goals of the application while creating a consistent and effective user experience and a unity of structure in the design. The goal is to encapsulate the requirements in a concrete way that can guide design and to make decisions about dialog strategies and dialog elements that permeate the design, thereby achieving consistency.

The elements of a high-level design include definitions of the following:

- Key design criteria
- Dialog strategy and grammar type
- Pervasive dialog elements
- Recurring terminology
- Metaphor
- Persona
- Nonverbal audio

4.2.1 Key Design Criteria

The first element of the high-level design is a list of the key design criteria for the application. You must keep these criteria in mind throughout high-level and detailed design. This list will provide guidance when it comes to making difficult trade-offs.

The list of key criteria should be very short, generally one to three items. Naturally, there is a long list of attributes that every system tries to achieve. However, the purpose of the list of key criteria is to guide design trade-offs and the major focus of the design, so a very short list of the highest-priority criteria is best. By the time requirements definition is complete, the key criteria should be pretty obvious. They should be covered by the success metrics defined during the requirements process.

4.2.2 Dialog Strategy and Grammar Type

A **dialog strategy** is a basic scheme for structuring the dialog of an application or task. For example, if you are designing a travel application, you might choose to go step-by-step, as if filling in a form, asking the caller for one piece of information at a time. For example, you would first ask for origin city, then the destination, then the travel date, and so on. Alternatively, you might choose to accommodate greater flexibility in how callers can describe their travel plan. You might start with a prompt such as, "What are your travel plans?" and handle a wide range of inputs such as, "I wanna go from New York to San Francisco next Tuesday," "I need a flight to San Francisco," and "I need to be in San Francisco by three p.m. Tuesday."

In the latter case, the application must choose its next dialog step based on the caller's input. Furthermore, the

grammar must be flexible enough to handle the great variety of possible caller utterances. In this case, you would typically use a statistical language model for the recognition grammar.

In general, the choice of dialog strategy will strongly influence the choice between rule-based and statistical grammars. Chapter 5 covers the various dialog strategies and discusses how to decide which grammar type is best for your application.

4.2.3 Pervasive Dialog Elements

A number of operations happen frequently throughout a VUI. For example, in every dialog state, you must provide handling for typical problems that may occur, such as recognition rejects. Making decisions about the approach to such operations up-front, before detailed design begins, will help ensure consistency in how these elements are designed throughout the system.

Pervasive elements include error handling (e.g., handling of recognition rejects and timeouts) and universals (i.e., commands always available to callers in every dialog state, such as "Help"). Particular applications may have other elements that either happen repeatedly or influence many dialog states and therefore should also be included in the high-level design. One such element is the login strategythe means by which callers are identified. Chapter 5 describes the design choices for error handling, universals, and login.

4.2.4 Recurring Terminology

It is important to use terminology consistently throughout an application. Violating this rule will lead to confusion for callers. Furthermore, if the application is used in concert with other systems, such as Web sites, it is important to ensure consistency in terminology between them. Deciding on terminology up-front will help ensure consistency.

The topics of the next three sectionsmetaphor, persona, and nonverbal audioare sometimes referred to as the **consistent character** of the application because they present a layer that plays a key role in creating the look and feel of the system. Keep in mind, however, that these elements are there not only on the surface, for look and feel, but also to play key roles in achieving your usability goals.

4.2.5 Metaphor

Metaphors are used in many user interfaces to help create a useful mental model for the user. A **metaphor** uses knowledge about one concept to understand or comment on a second concept. In a user interface, metaphors can be used as a tangible analogy for the abstract organizational scheme of an application.

Familiar examples of metaphors in user interfaces include the desktop metaphor in Microsoft Windows and the shopping cart metaphor at Amazon.com. Metaphors can be overarching, influencing the entire application (much like the desktop metaphor), or they can be specific to a subset of operations (like the shopping cart). The metaphor for a voice system can be as simple as a definition of the role of the system with respect to the caller (e.g., a personal assistant). If you use an overarching metaphor, you should choose it in advance of detailed design, because it may influence many aspects of the design. At the very least,

some thought should be given to the role the system plays for the caller.

4.2.6 Persona

When people engage in conversations, even over the telephone and even if they know they are conversing with a machine, they make many inferences about the kind of "person" they are speaking with. These inferences are driven by numerous characteristics, including voice quality, the words spoken, how the words are spoken, and so on. This phenomenon provides an opportunity for a company to create a specific image and extend its brand by carefully designing the persona behind its speech system. If you define the persona before detailed design, you will be able to consistently apply it as you craft the interface.

Many companies invest significantly in creating and extending their brands. When you work with such a company, it may be worth a significant effort to design a persona that supports its brand and corresponds to how it presents itself in other media (e.g., television ads). You may even create more than one persona (with voice recordings of sample dialogs) and compare them in focus groups.

Other companies may have little interest in the branding opportunity offered by a speech system. Even in those cases, it is worth at least a small effort to describe a persona before detailed design begins. Doing so will result in far greater consistency throughout the design. Furthermore, it will provide guidance in choosing a voice actor to record the prompts and will help the actor achieve a consistent delivery.

Some technology and platform vendors offer prepackaged or "standard" personas. These are a set of carefully

designed personas, with voices already chosen, and a set of sample dialogs you can listen to in order to understand the look and feel of the persona. Although using a standard persona does not provide differentiation in terms of brand, it does provide an inexpensive way for a company to take advantage of a well-designed persona. Additionally, some vendors create TTS voices using the same voice actors who voice their standard personas. You can take advantage of this when designing systems that combine recorded voice with TTS, thereby achieving a smoother integration between recorded and TTS segments of prompts.

Some practitioners claim that the reason for designing a persona is to fool the caller into thinking the system is human. We advise against that. Design choices should never mislead the caller about the capabilities of the system. The reason for designing a persona is to better meet user needs by creating a more engaging and familiar experience and a more usable system, and to better meet business needs by extending the company brand and creating a favorable and appealing image.

The issues surrounding persona design are complicated, especially considering particular user and business goals, so we devote all of Chapter 6 to this topic. Furthermore, because deploying companies need to be aware of the value of explicit persona design for their applications, Chapter 6 also includes information on a number of studies that support our contentions about the role a persona can play in the perceptions callers have of a system.

4.2.7 Nonverbal Audio

Nonverbal audio (NVA) includes all sounds, other than speech, that you design as part of your system. There are

three primary goals for using nonverbal audio in applications:

- To help create a particular look and feel
- To solve usability problems
- To communicate particular types of information

Look and Feel

The look and feel of an application can be enhanced with careful use of nonverbal sounds. To achieve a uniformity of look and feel, the set of sounds should be carefully designed. There are three types of NVA specifically designed for look and feel: background music, environmental sounds, and branding sounds.

When you use background music, you should carefully choose it for relevance to the application or piece of the application it is associated with. Background music has been used most effectively in voice browsers, with different background music associated with different "voice sites" (e.g., sports news, restaurant guide). Keep in mind that extra sounds over the telephone, in addition to speech, can lead to distortion and make the speech difficult to understand. When you use background music, it is often most effective to have it play for only a few seconds to set the mood, and end before the spoken interaction begins.

Environmental sounds are used in a similar way. For example, a visit to the restaurant guide might begin with a few seconds of the noises you typically hear in a busy restaurant. The same precautions mentioned for

background music apply to the use of environmental sounds.

Branding sounds are earcons specifically designed to identify a particular company or service. Many companies already have readily identified branding sounds. The typical place to use these is right at the beginning of a call, as part of a welcome message.

Usability

You can solve or mitigate a number of usability challenges by using NVA. One such issue is latency. Some applications access other systems, such as backend databases, with unpredictable amounts of latency before connecting or returning a result. Complete silence may be frustrating for callers. They may wonder whether the system has disconnected or is still working on their problem. To fill the space, you can use latency sounds, such as music or a specifically designed repeating sound. In addition to creating more interest than silence, the sound indicates to callers that they are still connected to the system and it is working.

Another usability issue that can be aided with nonverbal audio is **landmarking**. Systems that move between different services (e.g., voice browsers) can be confusing to the caller as the context switches. You can landmark each service with a sound that identifies it. These sounds can be background music, environmental sounds, or specifically designed identifying earcons. As noted earlier, such sounds should last only a few seconds and should not overlap with spoken interaction.

Communication

Nonverbal audio can be used to communicate specific messages. For example, you can design a specific earcon to indicate a specific meaning. We are familiar with many such sounds in our daily lives, the most common being the sound of a telephone ringing.

Earcons are sometimes used to communicate the occurrence of an asynchronous eventfor example, a tone that indicates that voice mail has arrived while you are in the middle of a telephone conversation. Earcons have also been used to indicate voice hyperlinks in some voice browsers.

General Considerations

You should not assume that all your designs should use nonverbal audio. Many designs work quite well without it. When using NVA, you should keep in mind a number of general considerations.

The most important guideline is to be sparse. If you use too many sounds, they lose their effectiveness and may lead to confusion. If you decide to use multiple sounds, design them carefully to work well together. For example, if you use a number of earcons with different meanings, make sure that each sound is distinctive. Although trained musicians may be sensitive to subtle differences, most callers will differentiate only those sounds that have significant contrast. Furthermore, think about the complete set of sounds and design each of them with some sense of how they will work together to create a particular look and feel.

Always test NVA over the telephone. The sound over the phone will be very different from the sound over high-quality speakers in the recording studio. Some examples

that sound great in the studio may not work at all over the phone.

During high-level design, decide whether you will use any nonverbal audio and, if so, where. This is important because other design decisions will depend on placements of NVA. The specific designs can be left for the detailed design phase, and final audio production left to the development phase. In general, the design of NVA involves a very different set of skills than the other VUI elements, so it is often best to bring in someone with appropriate expertise in sound design.

For more information on the design of NVA, see Raman (1997) and Kramer (1994).

4.3 Conclusion

This brings us to the end of our description of the application definition process: requirements and high-level design. This definition should provide you with the groundwork needed to create a design that is structurally consistent, engaging, and effective at meeting the needs of users as well as the business.

A number of the methodological principles first discussed in Chapter 3 have been applied in this chapter. Approaches have been described for soliciting the kind of *user input* needed for this phase, including focus groups, interviews, surveys, and observational studies. *Thorough early work* sets the stage for the detailed design, taking into account important considerations of *context*. The process stresses both the *business and user goals*, so that your design can successfully satisfy both.

In the remainder of this part, we first discuss the issues of dialog strategy choice (Chapter 5). Then we examine persona design in detail (Chapter 6) and apply the definition process to a sample brokerage application (Chapter 7).

Chapter 5. High-Level Design Elements

A number of design choices will have ramifications throughout the design process and a major influence on the details of the resulting design. These choices cover the dialog strategies and grammar type as well as the approaches to pervasive elements of the dialog, including error recovery and universal commands. This chapter covers the common techniques and strategies and equips you with approaches to help decide what is best for your application.

5.1 Dialog Strategy and Grammar Type

The basic choice of dialog strategy and grammar type will have significant effects on all other design decisions. In general, you consider dialog strategy and grammar type together, because particular choices of dialog strategy may demand a particular type of grammar (e.g., rule-based versus statistical).

One common way to classify dialog strategies is to separate them into directed-dialog (or system initiative) versus mixed-initiative types. In a **directed dialog**, the system asks the caller very specific questions and expects very specific answers. In effect, the system initiates, and closely directs, all interaction. Most systems deployed thus far have used a directed-dialog strategy. A directed dialog for travel planning might sound like this:

(1)

SYSTEM: What's the departure city?

CALLER: Um, San Francisco.

SYSTEM: And the arrival city?

CALLER: I wanna go to New York.

SYSTEM: OK, what day are you leaving?

CALLER: Next Tuesday.

SYSTEM: Great. And what time do you want to go?

CALLER: Sometime after ten a.m.

This directed dialog is an example of **form-filling**. The caller is asked a series of directed questions as if the caller were filling out a form.

This is one of the two most common types of directed dialog structures. The other is a **menu hierarchy**. In menu hierarchies, the caller is presented with a number of options in a menu (e.g., "Which would you like to do: get flight information, make a reservation, or hear about our special offers?"). Once the caller makes a selection, the system offers another menu until the application settles on the one item or action the caller wants. Typical applications use form-filling and menu hierarchies at different points, depending on the nature of the information and the caller's mental model.

With a **mixed-initiative** dialog strategy, the same travel dialog might allow callers more flexibility in what they can say. First, the initiative comes from the caller. Depending on

the caller's response, the system may then take initiative and prompt for missing information:

(2)

SYSTEM: What are your travel plans?

CALLER: I wanna go to New York next Tuesday morning.

SYSTEM: OK, and what's the departure city?

In this second example, the caller provides several pieces of information for the trip, and then the system takes the initiative and prompts for the rest. All mixed-initiative dialogs need to include back-off strategies to capture missing pieces of information.

There are a many shades of gray between directed-dialog and mixed-initiative approaches. In the following example, the system makes a direct request, but the recognition grammar is flexible, allowing the caller to provide additional information.

(3)

SYSTEM: What's the arrival city?

CALLER: I want to go to New York next Tuesday.

SYSTEM: OK. What time do you want to leave?

In this piece of dialog, the system makes a directed request for the traveler's destination. However, the caller replies with both the arrival city and the day of travel. The system is capable of handling this input and proceeds to prompt for the departure timethe only remaining piece of information needed. Although this approach is convenient for the caller, it makes the recognition task a bit more challenging because more variation is covered in the grammar, something that increases the size of the recognition model. This kind of strategy makes sense when many callers include additional information in their responses. Some applications support portions of mixed initiative and directed dialog, again depending on the nature of the information and the caller's mental model.

Call routing, as discussed in Chapter 1, is the most commonly deployed type of mixed-initiative system. Call routing is popular because it fulfills a strong business need. Many companies provide telephone access to a wide array of services, transactions, and information sources, often

numbering in the hundreds. But previous automated approaches to help callers connect with the appropriate service have been problematic. Some companies have tried to use touchtone systems with long menu hierarchies. These have proven extremely difficult to use because of the difficulty of mapping the large number of diverse services onto an intuitive menu structure. Both the hang-up rates and the misrouting rates have been very high in these systems. Other companies have a pool of 800 numbers, but many callers do not know which number is most appropriate for their issue and end up calling the wrong service. This costs the business money because of transfers, and it wastes the caller's time. Speech-based applications that use statistical grammars and advanced natural language understanding have proven to be very effective at solving this important business problem.

Mixed-initiative systems of the type shown in dialog 2 are relatively rare but are likely to become more common. Although such an approach can greatly improve efficiency and ease of use, the business drivers for deploying such systems are not as strong as for call routers because a directed dialog can often solve callers' problems, even if less efficiently.

Choosing a mixed-initiative approach usually implies the need for a statistical grammar (SLM). The amount of variation that can be expected from callers is simply too great to capture with a handcrafted rule-based grammar. It is hard to predict all the variations, and the grammar would grow huge and complex.

The following simple guidelines will help you decide whether to use a statistical or rule-based grammar. Use a rule-based grammar in the following situations:

- When caller input can be constrained

- When a handcrafted grammar can achieve high coverage
- When free-form speech is not expected

Use a statistical grammar in these cases:

- When the preceding guidelines for rule-based grammars do not hold
- When an open prompt is required rather than a prompt that explicitly describes what to say (e.g., for call routing)
- When a grammar with trained probabilities can perform better because of contextual information (e.g., name spelling, given that certain letter combinations are far more likely than others)
- When efficiency is highly important and you wish to minimize the number of steps to complete a task

When you use a mixed-initiative/SLM approach, it cannot be stressed enough how important it is to use appropriate VUI design strategies. A number of aspects of design are significantly different for mixed initiative than for directed dialogs, including prompting, error recovery, and other strategies. The mental model that must be communicated to the caller is different for an SLM-based system than for a directed dialog. With a directed dialog, you can direct the caller quite explicitly. With a mixed-initiative system, the range of possibilities is simply too large. You must find other ways of creating the model of how callers should speak to the system. When there are problems (e.g., a

recognition reject), you must carefully design error prompts to get callers back on track and clarify how they should speak. Chapter 12 presents specific guidelines for mixed-initiative prompting.

SLM-based grammars have one other advantage that many people don't recognize. When using an SLM and analyzing data during the tuning stage, you often learn about things callers are asking for that you never would have discovered with a touchtone or directed-dialog speech system. We have added routes to call routing systems based on lessons we learned from the data, often teaching the deploying company things it never knew before about the needs of its customers.

Some practitioners claim that the main purpose of using advanced natural language understanding (mixed-initiative) technology is to make callers believe they are talking to a live human being. That is absolutely not the case! We strongly advise against any attempts to fool callers into thinking they are interacting with a person rather than a machine, for the same reasons stated in the discussion of persona design in Chapter 4. Such an incorrect mental model is likely to lead to problems, given that even the most advanced technology is far less intelligent and capable than a human. The purpose of using natural language technology is to better meet business needs and user needs in those cases when it is the best solution for the application.

As with any design decision, when you are selecting between a directed-dialog and a mixed-initiative strategy, consult all the information gathered in the requirements phase about the needs of the business, user, and application, and keep in mind the key design criteria for the application.

5.2 Pervasive Dialog Elements

In addition to choosing a dialog strategy, in the high-level design phase you need to make decisions about how to handle several pervasive dialog elements. We will discuss the choice of error recovery strategy and universals, which are relevant to all applications. We will also discuss login strategies, a high-level design choice that must be made for some applications.

5.2.1 Error Recovery Strategies

There are two reasons error recovery is needed: The technology is not perfect, and neither are humans. The recognizer may not be very confident that it understood the caller, it may not have been ready to accept the caller's utterance, it may not have heard the caller, or it may be taking too long to generate a recognition hypothesis. As for humans, users often misspeak their responses, slur their words, stay silent, provide irrelevant information, and push buttons on the keypad at times when the system is expecting speech.

Applications must be armed with the logic to handle these situations so that callers can get back on track, no matter what the problem was, and succeed in completing their task. The challenge for the designer is to understand the nature of the caller's problem and provide the information needed to recover.

The error recovery strategies discussed here involve designing how the system should behave when the recognizer does not return a recognition hypothesis. The two most common messages the recognizer returns when it

cannot find a result are **reject** (no good match was found) and **no-speech timeout** (no speech was heard). In this chapter we cover the choice of strategy for handling such error messages. The actual prompts used in response to error messages are highly dependent on the dialog state and are worked out as part of the detailed design (see Chapter 13). How you detect and handle recognition mistakes (when the recognizer returns a wrong hypothesis) depends on the dialog state context and is also discussed in Chapter 13.

To define the error recovery strategy for your application, you must consider both the type of error and the number of errors the caller has encountered. First, let's look at the most common error type: rejections.

Escalating Detail

A common approach to reprompting following a rejection is to give more detailed instruction or examples of what the caller should say (or both). With successive rejects, even more detail is provided. This approach has been variously referred to as **escalating detail** or **progressive prompting** (Yankelovich, Levow, and Marx 1995; Weinschenk and Barker 2000). Here's an example from a travel planning application:

(4)

SYSTEM: When would you like to leave?

CALLER: Well, um, I need to be in New York in time for the first World Series game.

SYSTEM: <reject>. Sorry, I didn't get that. Please say the month and day you'd like to leave.

CALLER: I wanna go on October fifteenth.

Instead of asking "when" the caller wants to leave again, the error prompt explicitly tells the caller to say the month and day.

Another aspect of dialog 4 is the phrase, "Sorry, I didn't get that." This is a common technique in which the application provides feedback on the exact nature of the problem: The system did not understand the caller. This technique is often combined with escalating detail.

The first error prompt the caller hears in a dialog state is sometimes referred to as the **first-level error prompt**. When the caller encounters a second error in a row, the second-level error prompts play. Upon the second error, some designers like to begin with a phrase such as, "Sorry, I *still* didn't get that," followed by either a more detailed instruction or a suggestion to try an alternative approach

(e.g., "Please key in your account number"). In general, during detailed design, you should craft the specific error recovery prompts for each state. In this way, you will maximize the chance of recovery; the best approach will vary depending on the state.

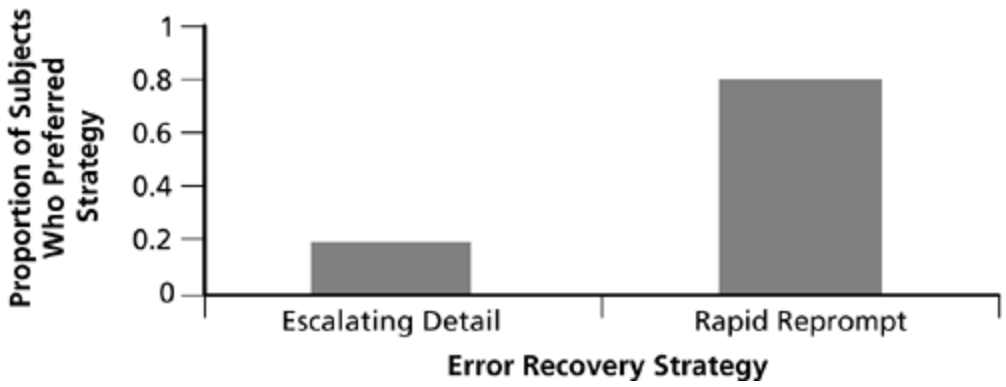
One disadvantage of using the escalating detail approach is that often the caller knows exactly what to say to the system and only needs another chance. As a result, callers sometimes get impatient with detailed error messages. The rapid reprompt approach, discussed next, is an alternative strategy that is often effective.

Rapid Reprompt

The **rapid reprompt** approach does not provide detailed information right away. Instead, the system replies to the error with a short prompt such as, "I'm sorry?" or "What was that?" This is similar to the kinds of statements people often use in conversation to indicate that they did not understand the speaker. If another error occurs, the behavior usually follows the escalating detail strategy by providing detailed information about what to say. Table 5-1 compares the two strategies.

A number of experiments at Nuance compared the two strategies. When all variables are controlled and when only the prompt wording is varied, the rapid reprompt strategy is unquestionably preferred by users. Figure 5-1 shows the results of comparing the two error recovery strategies.

Figure 5-1. Preference for error recovery strategies.



The major disadvantage of rapid reprompt, however, is that it does not provide the caller with detailed information right away. As a result, callers who are unsure of what to say often take an extra step to recover from an error, because they must wait for the next error prompt to get explicit instructions. On a positive note, callers do not seem to mind this extra step because they do not perceive it as an error.

The best place to use the rapid reprompt strategy is in directed dialogs whose structures are obvious or with repeat users of a system. It is not recommended for cases with open-ended prompts, such as, "How may I help you?"

Table 5-1. Escalating Detail Versus Rapid Reprompt

EVENT	ESCALATING DETAIL	RAPID REPROMPT
Initial prompt	What's your account number?	What's your account number?

EVENT	ESCALATING DETAIL	RAPID REPROMPT
First error	Sorry, I didn't understand. Please say your 10-digit account number.	I'm sorry?
Second error	Sorry, I still didn't understand. Your 10-digit account number statement at the top right corner. Please say your account number now, or for more information, say, "Help."	Sorry, I didn't understand. Please say your 10-digit appears on your monthly account number.
Third error	Sorry, I still didn't understand. Please key in your account number, or say, "I don't know it," and I'll connect you to someone who can help you.	Sorry, I still didn't understand. Your 10-digit account number appears on your monthly statement at the top right corner. Please say your account number now, or for more information, say, "Help."

Variation on Rapid Reprompt

Some designers prefer a variation of the rapid reprompt strategy that quickly restates what the system is expecting

but does not go into detail or provide examples. Compare the following sample dialogs.

Here's a sample dialog with rapid reprompt:

(5)

SYSTEM: What's your account number?

CALLER: Uh, it's five five five … wait.

SYSTEM: I'm sorry?

Here's a sample dialog with the rapid reprompt variation:

(6)

SYSTEM: What's your account number?

CALLER: Uh, it's five five five … wait.

SYSTEM: Sorry, what was that account number?

Rapid reprompt and escalating detail are the two most common error recovery strategies used for rejections. Which one you use will depend on the application and the specific task. Even if you choose rapid reprompt as the general strategy, you may find that you need to override this behavior in certain dialog states.

No-Speech Timeouts

No-speech timeouts are the second most common type of error. Although much of the error recovery behavior for rejections can be adopted for no-speech timeouts, it is a good idea to treat them separately. For example, if the reject prompt begins, "Sorry, I didn't understand …" you might substitute "Sorry, I didn't hear you" even if the remainder of the message is the same. Furthermore, the causes of no-speech timeouts are likely to be different from the causes of rejects, so you should consider alternative strategies. Given that callers often remain silent when they are not sure what to say, a rapid reprompt strategy may not be as effective for timeouts as it is for rejects.

State-Specific and Global Error Counts

After two or three successive errors in a single dialog state (regardless of the type of error), it is unlikely the caller is going to recover. In fact, hang-up rates are very high after a few rejects. For this reason, every application should have defined behavior for what to do when a maximum number of errors has been reached. For example, if live agents are available, the application might transfer callers to an agent after they have experienced three successive rejects. In the high-level design phase you should establish three thresholds: the maximum number of successive errors in a single dialog state (the state-specific error count), the maximum number of errors counted globally, and the maximum number of disconfirmations (responses of "No" in confirmation states).

A typical number for the state-specific error count is three. Designers differ on whether the three errors should be of the same type. We recommend that any error count toward the maximum error threshold. An exception might be in an initial dialog state for a newly deployed system. Sometimes callers are caught off-guard by the new system, especially one that is expecting the caller to talk to it. In these situations it is sometimes better to customize the threshold and corresponding behavior for no-speech timeout errors.

For disconfirmations, a default limit of two is recommended (this is based on hang-up rates from real deployments). When the threshold is reached, it is a good idea to send callers directly to an agent if one is available.

5.2.2 Universals

Universals are commands that are always available in an application. The most common universal is "Help." Most applications allow the caller to say "Help" at any time, in any dialog state. In response to "Help," the application

provides the caller with detailed instructions specific to the current dialog state.

There is a set of six universals that we recommend for all applications: help, repeat, main-menu/start-over, go-back, operator, and good-bye. Chapter 9 covers the motivation for this standard set of universals and explains how to use them. During high-level design, your task is to decide whether to use these standard universals and whether to supplement them with any universals that have particular relevance for the application. For example, a stock trading application may use "broker" as a universal so that the caller can always get directly to a live broker.

5.2.3 Login

Although not as pervasive as error handling and universals, login strategies can have an effect on many parts of certain applications. (**Login** is the part of some applications that gathers information such as account numbers and PINs from callers.) The decision to be made is whether to log in callers at the beginning of the interaction or wait until they want to do something that requires secure information (often called **delayed login**). There are pros and cons to each approach.

The benefit of up-front login is that the designer can customize the system's behavior for that individual or class of caller. For example, if the caller has not set up automatic payments, the system does not have to offer automatic payment options related to managing an existing account. The application can also be intelligent about the help strategy and can even adjust the prompt detail depending on whether the caller is a novice or expert user.

A disadvantage of this type of login is that callers must provide personal information even if they want to do something generally available, such as getting a stock quote. This takes time. Often, users forget their PINs and have a difficult time getting into their accounts. This can be frustrating if all they want to do is a simple task that does not require account information.

The opposite is true for delayed login. Although callers are not bothered with remembering account numbers, PINs, and passcodes, they may have to wade through irrelevant options in menus, and often experts must use dialogs designed for novices.

Your choice of login approach depends greatly on the application's key design criteria and how easy or difficult it is for users to remember or access the required login information.

5.3 Conclusion

Understanding high-level design elements is critical when you're creating a framework for the detailed design. Many of the design choices you make at this stage present trade-offs. Which is more important: the clarity of directed dialog, or the flexibility of mixed initiative? The efficiency and naturalness of rapid reprompt, or the informative nature of escalating detail? The universal access of information inherent in delayed login, or the personalization afforded by up-front login? You need to weigh each of these choices before you make any definite decisions.

Chapter 6 discusses the design of a persona for your applicationone part of the high-level design that is unique to voice user interfaces.

Chapter 6. Creating Persona, by Design

In helping customers design their VUIs, we ask them many questions about their corporate identity and brand and about typical users of the system (see Chapter 4). The intent of such questions is to help the designer arrive at the most appropriate "character" or "personality" for the interface, which the user is bound to infer in even a single interaction. In response to these questions, however, we sometimes hear, "Oh, this app isn't going to have a personality." A substantive body of research, though, shows that we humans cannot help inferring personality traits and social information from the voices we hear, even if we encounter them as brief, recorded samples.

We infer not only the speaker's gender but also age, ethnicity, socioeconomic status, geographic background, level of education, and emotional state. We can also infer personal qualities, perceiving the person to be, for example, trustworthy, punctual, generous, hospitable, even-tempered, or romantic. In U.S. culture, breathiness in male voices is perceived as young, but in female voices as sexy. In other words, the entire field of research that sociolinguists call **speech evaluation** strongly suggests that there is no such thing as a voice user interface with no personality.

Many speech evaluation studies center on the social cues that are transmitted through a speaker's accent. These studies show the significant extent to which people rely on certain linguistic cues in other people's speech to infer nonlinguistic attributes and to make value judgments about the speakers as individuals. These attributes and judgments are based on preconceived notions about the

social group speakers are associated with. For example, if we think of people from Brooklyn, New York, as being tough, and then we hear someone speaking with a Brooklyn accent, then we judge the individual as being tough.

In an article titled "Y'all Come Back Now, Y'Hear?" Soukup (2000) explores language attitudes in the United States toward Southern accents. In this study, some 300 college students from New England and Tennessee were asked to evaluate four speakers' samples: two with a Southern accent and two with a "neutral" accent, with male and female voices represented for each. The setting for this study was a job interview situation in sales. Statistical analysis of the data strongly confirmed that having a Southern accent is a strike against job applicants. Southerners categorically lost on "competence" (e.g., intelligence, education, determination), a quality the informants deemed most important for job performance. There were some positive associations with this accent: Notably, the Southern female voice took the lead in social attractiveness (e.g., friendliness, good humor) but not enough to endorse the speaker as a desirable employee. Language attitudes toward Southern American English were rather negative overall.

This study corroborates the findings of Giles and Smith (1979), in which subjects rated Cockney-accented speakers lower than standard-accented speakers, and Labov (1966), where listeners tended to downgrade speakers with a New York City accent in terms of their job suitability.

Not all speech evaluation studies focus on regional differences. In an older study (Giles and Powesland 1975) whose findings are now relevant to the design of commercially deployed VUIs, student teachers were asked to assess eight hypothetical students with respect to intelligence, enthusiasm, self-confidence, gentleness, and

"being privileged." These fictional students were defined by (1) a photograph, (2) a taped sample of speech, and (3) school work. The research question posed in this study was, What would happen if information from one of the three sources gave a favorable impression, but information from another source gave an unfavorable one? The findings of this study consistently point to the central role that speech plays in our evaluation of others. Favorable impressions of the speech sample overrode unfavorable impressions from the photograph and the work sample. Unfavorable impressions from the speech samples overrode favorable impressions from the other sources.

These findings provide one of the most compelling arguments for incorporating explicit persona design into the larger design and production effort of a VUI, especially in the case of commercially deployed applications. Simply put, speech is more powerful than the written word or a visual image in making a good impression.

Voices can even suggest physical characteristics. Most people have had the experience of meeting in person someone who until that point had only existed, so to speak, as a voice on the telephone. When finally we meet the person face-to-face, either we feel validated that our mental image of the person was on target, or we are taken aback by the mismatch. ("He seemed *taller* over the phone.") Either way, we cannot help imagining what people must *look* like based on what they *sound* like. The relevance for VUI design is that the people who call your application are likely to follow the universally human impulse to evaluate the voice they hear and draw on it for nonlinguistic information.

The research on speech evaluation contradicts assertions that a particular voice user interface is "neutral"that it "doesn't have a personality" or that "there's no persona."

The choice not to deal with personality nevertheless is likely to result in some personality being perceived by the user. The voice of such a system might be perceived as an announcer reading awkwardly worded messages at the user, or perhaps a robot who speaks "computerese." In fact, we have found that people sometimes mistake prerecorded audio for an artificial, synthesized voice, especially in applications whose prompts are poorly concatenated or awkwardly worded. In any case, it is not advisable to leave such perceptions to chance, especially because branding and image are at stake. The creation of persona should not be haphazard but the object of an explicit design effort.

To fully leverage the power of the spoken word, voices alone are not enough. Both favorable and unfavorable mental images that we infer from speech signals also depend on what the voice is saying and how it is said (see Chapters 10 and 11, respectively). Taking all these elements together, we can think of speech application design as creating for users a total language experience in which they interact with an ideal employeeif we get the details right. This is the world of persona.

6.1 What Is Persona?

"Persona" is defined as the role that we assume to display our conscious intentions to ourselves or other people. In the world of voice user interfaces, the term "persona" is used as a rough equivalent of "character," as in a character in a book or film. A more satisfying technical definition of "persona" is the standardized mental image of a personality or character that users infer from the application's voice and language choices. For the purposes of the VUI industry, **persona** is a vehicle by which companies can brand a service or project a particular corporate image via speech.

6.2 Where Does Persona Come From?

Interaction with the speech application should suggest an underlying stable, coherent, and consistent personality. Specific personality traits of the desired persona will likely include helpfulness, competence, efficiency, friendliness, and likeability. Getting the right persona and getting the persona right depend on attention to a lot of language-related details in order for such traits to be inferred by the user.

It is a common misconception that persona design begins and ends by hiring someone with a pleasant voice, with little attention paid to what the user will actually hear in the context of an extended interaction. Prompts that sound artificial and mechanistic and dialogs that clunk along instead of flow will fail to evoke a well-designed, coherent persona. Instead, they will evoke the verbal equivalent of a company representative who dresses poorly or projects an awkward, undesirable image.

A prerequisite for projecting favorable personality traits has to do with the way messages are worded and dialogs are structured. Prompts and larger stretches of dialog must be modeled after the communication system that users are most familiar with, the system they began learning in infancy: everyday conversation. The alternative would be to subject VUI users to a less familiar system, such as an over-the-phone reading of questions and informational messages that typify the formal features and structures of written discourse. This is what we often find in traditional touchtone systems and is a legacy inherited by many VUIs. As we discuss in Chapter 10, however, written discourse is best suited for reading and is not ideal for ensuring *listening* comprehension, for which conversational

discourse is ideal. Chapter 10 also explains how to apply basic characteristics of conversational language to the task of crafting prompts. Many of the ideas presented in that chapter are fundamental to the design of a voice persona because they are relevant to spoken language in general.

Linguistic distinctions among personae emerge from application-specific details, such as the age group, professional affiliation, or regional background of the typical user. They also derive from decisions about "who the persona is" vis-à-vis the user in terms of social rolesfor example, an assistant talking to a boss, a librarian to a researcher, a tour guide to a visitor, a teacher to a student, and so on. Such differences in social dynamics imply differences in word choice and sentence structure, as discussed in section 10.5.

Another way to differentiate one persona from another is by using different dialog strategies. For example, whereas one persona might rely on implicit confirmations (e.g., "How many shares *of IBM* do you want to buy?"), another might instead employ explicit confirmations ("You want to buy IBMis that correct?"). The following section reviews basic guidelines that will assist you in making such decisions.

Prosodythe intonation, stress, grouping, and rhythm of stretches of speechis also an essential and systematic component of conversational language. Just as a persona cannot be supported or instantiated by prompts whose wording typifies written text, neither can it be supported or instantiated by messages that do not conform to basic principles of prosody, as is frequently the case with concatenated prompts. Prosody in VUI design is the theme of Chapter 11.

Finally, the perception of the persona you design is also affected by certain decisions relating to the design of the call flow and even by your design methodology. For

example, in Chapter 8 you will see a case in which the perception of basic linguistic intelligence depends on reentering a particular dialog state with a novel prompt ("Sorry about that, what's the *correct* amount?"). If a different design is used, the caller, upon reentry to the state in question, might hear exactly the same prompt that was played the first time through ("How *much* would you like to pay?"). In the context of an actual interaction with the system, this design choice yields a conversationally inept persona. From the point of view of the caller, the wording of the message upon reentry suggests that the system is not keeping track of the conversation, at least not as a human would, or is experiencing short-term memory loss. Chapter 8 presents a prompt design methodology that will help you avoid such pitfalls.

6.3 A Checklist for Persona Design

This section deals with key considerations that will help you to zero in on a persona that will best suit the purpose, goals, and functionality of your application. As discussed in Chapter 4, you should decide on the persona before you design the system prompts in detail. Reversing this order will entail a major rewriting task to retrofit the prompts to the persona.

To identify an appropriate persona for your application, consider the following issues.

6.3.1 Metaphor and Role

What is the role of the system with respect to the user? For example, is it a familiar-sounding personal assistant who offers lots of tips and help? Or is the role somewhat more distantfor example, a knowledgeable, professional tour guide or stockbroker? Or should the character be still more impersonal, like a bank clerk or a telephone operator, simply asking questions and providing instructions? Consider whether the system is to serve as a communication aid, a provider of transactions, or a source of information.

6.3.2 Brand and Image

Many companies spend large sums on branding and building a corporate image through advertising and other marketing efforts. Brand and corporate image are also projected via a persona, whether or not the persona has

been explicitly designed. The persona chosen for a particular application should at least be compatible with brand and corporate identity, if not a chief conveyor of these ideals. As discussed in Chapter 4, the persona must be coordinated with any existing brand the company has, as well as with other customer touchpoints.

6.3.3 End Users

A successful, appropriate persona strikes the user as familiar, both linguistically and socially. Persona design must therefore consider who these users will be and in what contexts they will use the application.

Target Audience

To develop a compelling persona, consider the demographics, attitudes, and lifestyle of the target audience. We can easily imagine how a voice portal for teenagers requires a different persona than one for adults. In addition, when localizing a system and porting it to a different language, you must reconsider the design of the persona. A persona that works well in one culture may be inappropriate in another.

Frequency of System Use

Is the system accessed frequently by the same users? Is it a subscription-based service? Is the user identified by name? If so, the experience should be designed to build rapport or camaraderie with the user.

Mind-Set of the User

The system's persona should be compatible with the users' likely frame of mind. You should use one kind of persona if they are looking for entertainment, for example, and a different one if they are trying to land an airplane or selling stocks under volatile market conditions.

6.3.4 Application

One reason for designing a persona to "front" a speech interface is to facilitate task completion and to usher the user effortlessly from one recognition state to the next, again, by making the experience seem familiar, comfortable, and consistent. Persona appropriateness therefore depends on certain details specific to the application.

Content

The system's persona should be consistent with the application's content. The persona of an astrology voice site is unlikely to have much in common with the persona of a portal for mobile executives. Different personae are suitable for different tasks. Caricatured, over-the-top voices based on popular cartoons may be ideal for presenting main menu options in an entertainment-finding application for kids but not for checking the status of stock market trade orders.

Task-Related Issues

Consider how the choice of persona might facilitate certain key tasks. If the interface is complex and presents many functions couched in a less-than-straightforward mental model, it may benefit users if the persona is cast in the role of a helper, perhaps a teacher, who gives occasional reminders or tips.

The persona should be designed with usability issues in mind. Appropriate design can help solve usability problems, as in the example just mentioned. In contrast, usability can be hampered by the wrong choice of personafor example, a chatty persona for a voice-activated dialer. Dialing a phone number is a quick and simple task. If voice dialing is to bring value, it certainly must be efficient. People will not want to deal with a chatty personality every time they use a voice-activated dialer.

6.4 Persona Definition

With some idea of the kind of persona that is suitable for your application, you need to define a specific point of reference to anchor the persona in the minds of all those who have a hand in creating the **language experience** of the application.[1] This includes dialog designers, prompt writers, voice coaches or directors, the voice actor, and the marketing department of the company deploying the system. This common point of reference is a **persona definition document**. This resource includes at least the following components:

[1] Thanks to Wally Brill and Melissa Dougherty for assisting with the content of this section.

- **A biographical sketch:** This is a fictional background for the persona. It pays special attention to factors that might determine or reflect how a particular individual would speak: for example, geographical background, education, gender, ethnicity, affinity groups, and hobbies.

- **Vocal attributes:** These are the qualities the voice should suggestfor example, peppy and chirpy, or calm and mellow. Parameters such as these can serve as a guide in the search for a voice actor as well as an aid for voice actors themselves.

A photograph can also be a useful visual tool to help bring the persona into focus.

Once a potential voice actor is chosen, audio samples of hypothetical dialogs between a typical user and the application should be recorded. These samples will serve to

demonstrate how the persona will perform in the context of an interaction between the user and the system and should answer any questions as to the persona's suitability. Such recordings are particularly useful to make sure that the various stakeholders in a project have the same assumptions about what the final system will sound like.

One of the key benefits of starting with a persona definition has to do with ensuring a consistent user experience. By serving as a sort of "style guide," persona definition helps ensure a consistent linguistic experience throughout the interaction, especially where the wording of prompts is concerned. Imagine, for example, a banking agent in his early twenties from Flagstaff, Arizona. He might ask for a caller's patience by saying, "Hold on while I look that up" or might request repetition of an account type by saying, "Sorry, what account was that?" It would be unrealistic, however, for such a person, especially in the same conversation, to pose a question as grammatically self-conscious and as socially distant as, "To which account do you wish to transfer funds?" (as opposed to a more natural, context-dependent wording such as, "And the account you want to transfer *to*?") These prompts differ sharply in **register**, a topic that is covered in Chapter 10. For the present, however, it is enough to note that these prompts clash in the social cues they project. In effect, their incompatibility undermines the coherence of the personality that users are bound to infer.

In general, people prefer dealing with personalities that are identifiable and consistent over those that are confusingly unpredictable and inconsistent. Social psychologists Reeves and Nass (1996) assert, "People like identifiable personalities… . Quick assessments are valued, and undiluted personalities are more quickly and accurately considered. Complicated people are just that: 'Who knows

what they'll do next?'" Consistency of the user's experience with the interface depends in part on a consistent persona.

There is another benefit to defining a particular persona for your interface. It provides the prompt writer or team of writers with a point-of-reference tool for keeping the language experience familiar. By "familiar" we do not necessarily mean informal. Rather, we mean familiar in the sense of the communication system that users are most intimately acquainted with: everyday conversation. In other words, VUI designers should take full advantage of what is already familiar to users. Prompts that do not reflect conversational normssuch as, "An error message has been generated"do not flow from a sociolinguistically familiar persona. It is easier to capture familiar, everyday spoken language if you stay focused on what a socially and linguistically familiar persona might say.

6.5 Conclusion

VUI users know that the prompts they are hearing are only recordings. Some users even know something about speech recognition software, IVR platforms and networks, and backend integration. But even these tech-savvy users are "programmed" to infer mental images from the voices and linguistic choices they encounter. Designers therefore need to be "in control of the image that their application projects so that users will favorably assess the company's brand, values, and support of them as customers."[2] Whether or not you have planned for it, your application's persona is an expression or statement of an image. Don't leave users' perceptions of your application, and therefore of the company, to chance.

[2] Margaret Urban, personal communication.

Chapter 7. Sample Application: Requirements and High-Level Design

Albert Einstein made the comment, "Example isn't another way to teach, it is the only way to teach." Indeed, when learning VUI design, you can read lists of procedural steps and guidelines, but, until you see the process in action, you can never fully comprehend the details, subtleties, and constraints imposed by real-world problems.

This chapter begins our presentation of a sample application. Throughout the book, we will carry on this example, stepping through all the details of definition, design, development, and deployment. You will have a chance to see the methodology and design principles in action. Keep in mind that our purpose is not necessarily to advocate a particular solution or set of design choices for the application at hand, but rather to show the process in action. We have fashioned our design decisions and test results for this example to demonstrate particular points. They do not necessarily represent actual test results on a real deployment.

The sample application is for our imaginary client Lexington Brokerage. The application it wants to build will allow callers to get stock information, place trades, and manage their accounts. We begin with a brief introduction to Lexington Brokerage and its goals. Then we dive into the details of requirements definition and high-level design.[1]

[1] The material in these chapters on the Lexington Brokerage application is based partially on presentations by Tony Sheeder and by Rebecca Nowlin at Nuance V-World: "VUI Design Under the Microscope," V-World 2001.

7.1 Lexington Brokerage

Lexington Brokerage is a small firm, not terribly well known, but aggressive and eager to become a major player in its industry. In a competitive analysis, Lexington executives have discovered that other brokerages have been offering speech recognition, instead of standard touchtone systems, in their call centers. They have met with industry analysts, who suggested that not only have the systems provided significant cost savings, but also brokerage customers are very satisfied with them. The analysts reported that customers prefer speech to touchtone systems and that they appreciate the easy access to their accounts and up-to-the-minute information on the performance of their portfolios. One analyst described a survey he performed in which customers of brokerages with speech offerings rated their brokerages higher on customer service.

Lexington has a touchtone system that provides stock quotes and basic account information. The executives have decided to create a speech system in order to expand the firm's automated offering to include trading and portfolio information, and to provide customers with an easier-to-use, more intuitive interface. They expect the automation of telephone trading to lead to significant cost savings.

We have just signed a contract to design, develop, and deploy the system. Lexington wants to deploy quickly, although it is concerned with achieving a high-quality result and making the deployment process smooth and predictable. We enter the first phase of our project: requirements definition.

7.2 Requirements Definition

Our goal for this phase is to understand the business, the end users, and the application details. We also want to come up with a set of metrics that will measure our success and point us to important issues that need attention. We begin by looking at the business.

7.2.1 Understanding the Business Goals and Context

Before our first meeting with Lexington, we do some quick homework. First, we explore the Web site. Luckily, a colleague has an account, so we can try some of the services offered. We note the variety of services offered on the site as well as the look and feel. The feel of the site is "down to business," with very little in the way of extra graphics. When you request a quote on a given company's stock, you also get additional company information such as trading volume and high and low prices for the year. Another click can get you details on company financial performance or headlines.

Next, we call the Lexington touchtone system. We notice an earcon at the very beginning, just before the welcome prompt. It sounds familiar, and we remember hearing it on a recent radio ad. We use the touchtone system to get some quotes, get account information, and transfer some money back and forth between two accounts. The system uses a male voice that is fairly emotionless and dry.

Now we send Lexington a proposed agenda for our meeting, and we call to make sure the right people will be

there to answer the variety of questions we have.

At the beginning of the meeting, we ask the Lexington personnel why they want to deploy a speech application. Jack Farmer, the marketing representative, tells us that Lexington wants to expand its service. "We want our customers to be able to get up-to-date information on their portfolios twenty-four by seven," he tells us. Lexington also wants to offer automated trading over the phone for customers who are often away from their desks and trade frequently. This service has never been available in the touchtone system.

We learn from Jack that customer service is extremely important to Lexington; it wants a system that is efficient and easy to use and makes it easy to connect to a live broker if desired. Another goal is to attract and retain customers. The firm has some concern about losing customers to competitors, so it would like to make its service more "sticky" (one that hangs on to customers) by using personalized features.

Next, we meet with Jane Brody, who coordinates advertising and is responsible for branding. "What do people think of when they hear your company name?" we ask her. "Value, dependability, reliability," Jane says, "The professional, helping hand." We ask what else they should think after using the system. "Efficiency and helpfulness," she says. "We help them get the job done."

We ask about the voice and tone of the touchtone system. Jane explains that the goal was to sound professional and authoritative. However, Lexington marketers are not happy with the result. They would like our help finding a better voice and defining an improved look and feel. She also explains that all the Lexington systems will begin with the branding earcon we heard on the touchtone system.

In the afternoon, we meet again with Jack and his team. We learn a number of things. The speech application will be one of many services that are available to Lexington customers. Traditional person-to-person services will always be offered. The Web site offers a superset of the functionality that will be offered by the speech system. The touchtone system presents a subset of the functions. A growing number of customers use the Web site, and many of them conduct their trades on the Web. These facts will influence design decisions. For example, if callers are accustomed to specific terminology on the Web site and even a certain organization of information, they will be more likely to generalize and apply these concepts when interacting with the spoken language system.

We request a copy of the specification for the touchtone system. Our goal is not to map touchtones onto speech. Speech systems should be designed with speech in mind. What we do want to glean from the touchtone spec is the type of language that is used and the functionality available, because callers will be familiar with these things from previous interactions.

We ask Jack's group how they foresee getting their customers to use the new system. Lexington has found that mailers included with statements are not very effective. They are considering advertising the service by using a postcard that describes some of the basic commands (although they are relying on the application itself to teach callers all they need to know about how to use the service). They will also advertise the speech system on the Web site, emphasizing that now customers can have complete access even when away from their desks.

Our discussions with Lexington have been fruitful. We now have an understanding of the business goals and context. The motivation for deploying the speech system is to

expand the firm's automated service offerings (thereby cutting costs), increase customer satisfaction, and retain customers. It wants to promote its corporate image of value, dependability, and facilitation. All the functionality of the touchtone system will need to be supported by the speech system, but the goal is to make the interaction more user-friendly. Finally, the application should not rely on any outside instruction to teach users how to interact with it. Callers must be able to learn the system simply by using it.

We agree with Lexington on a number of success metrics that correspond to the business goals. It is important to keep in mind that, during the tuning phase, we will measure many other characteristics of system performance in order to improve the system. However, the metrics we agree on now are the highest-priority indicators of success.

We select the following metrics:

- **Increased automation:** The primary means of expanding services and reducing costs will be the increased automation of trading, an operation that currently often utilizes a broker and is not available in the touchtone system.

- **Customer satisfaction:** Lexington Brokerage already has a process whereby customers provide feedback on services through periodic surveys. The data it has gathered for the touchtone system will serve as a benchmark for the speech system. The goal is to achieve better scores on subjective customer satisfaction than those for the touchtone system. The two focal areas covered in the customer satisfaction survey are simplicity of use and efficiency. A third area, accuracy, is added, because perception of accuracy will

play an important role in convincing callers to use a voice system, especially for trading.

- **Customer retention:** The ultimate measures of customer retention will await longitudinal analysis of customer statistics. We will try to get some early indicators by conducting a survey at the end of the pilot that includes questions about the user's intention to reuse the system. We will also measure usage of features designed to make the service sticky, such as setting up personal stock watch lists.

7.2.2 Understanding the Caller

We want to learn as much about the customers (our potential callers) as we can. What is their demographic group? Are they older, more mature customers or young go-getters? Do they use any jargon? Is a particular dialect prevalent? What about technological sophistication? Are they comfortable with computers? How do they feel about new technology such as speech recognition? What would they like from a system such as this? What are the typical usage patterns we should expect?

We sit down with the Lexington team members and ask them some questions about their customers. They tell us that their customers are mainstream traders in the moderate- to high-income range. They have customers of all different ages, but most tend to be in their 30s and 40s. There is an even mix of males and females. We're given a document with statistical information about the customers. It covers demographics as well as usage statistics for both the Web site and the touchtone system: how often customers call or log in, frequency of use of each feature, and so on.

In addition to Lexington's own information, we decide to talk directly to the users. We want to understand their state of mind and image of themselves as traders. Lexington's statistics can get us only so far in probing the internal thoughts of the customers. We decide to explore the needs of the customers by convening a focus group, and Lexington provides us with contact information of people who might be willing to spend some time telling us their opinions. With the help of a small monetary incentive, we find eight willing volunteers.

In a group discussion headed by one of our usability specialists, we learn that these folks are generally very receptive to automated services. They like the fact that they can get information and conduct transactions without waiting on hold. Saving time is important to everyone there, so the convenience of automated services is a definite plus.

Here's what three of the participantsRachel, Gordon, and Mikehad to say:

(1)

RACHEL: "I like taking care of things when I want to."

GORDON: "Yeah, convenience is definitely nice. I had a regular broker for a while, but I didn't get much service because I'm not a high roller or anything."

MIKE: "That's definitely true. I figure if I'm not getting much attention, then why not do it myself?"

GORDON: "And save on the commission."

What we learn from these comments is that it's not only the convenience that is important to these folks but also the fact that the service is an enabler. It allows them to be self-reliant, which has been a growing trend in financial decision making ever since the dawn of online trading.

The participants also tell us that they are concerned with accuracy. After all, they are managing their money. They want to be sure that if they get information from the service, such as stock quotes, this information is up-to-date. Security risks and accuracy make them a bit cautious about telephone services. When using the Web, they can see the information right in front of them and confirm it. For some reason, just hearing the information over the phone makes them a bit wary. What if they hear the information wrong and mistakenly confirm it? It's important to note these concerns and address them in later design decisions.

The focus group has given us some insight into the mind-set of the customer: We have discovered that they enjoy convenience and like being in control of their money. They

want to be sure that the decisions they make are based on correct information and that transactions are executed accurately.

We now probe deeper into their thoughts about the service. How do they use online brokerage Web sites today? How might they use something like an automated telephone system that they can talk to? We want to learn what customers use services for, how often, and from where (are they in their car, at home, at work?). Here's another snippet from the focus group:

(2)

LEADER: "I'd like you to talk about your experience with brokerage services. This includes brokers, Web sites, and touchtone systems. How often do you use these services, and what for?"

JUDY: "I usually call my broker every once in a while to make trades … um … I'll sometimes get a quote off a Web site, too."

LEADER: "How often would you say?"

JUDY: "Maybe … maybe once every two weeks."

BOB: "Oh, not me. I like to get quotes several times a day sometimes, and I check the value of my portfolio on a regular basis … especially these days when the markets are so volatile. I generally go online in the morning at work to see whether things are up or down and keep tabs on the market all day. If I could do it more easily, I might call more often, too, I don't know. It'd be nice to be able to call in on my way to work just as the markets are opening …"

GORDON: "I used to call my broker all the time for stock quotes, but I'm finding that it's just cheaper and easier to use the Web and trade online at home. I don't do a lot of trades, but maybe once a week when I feel I need to do something, I'll just do it over the Web. It's just quicker and easier for me."

LEADER: "So have you completely given up on your regular broker?"

GORDON: "Yeah, I haven't called him in a long time. Pretty much everything I used to do with my broker I can do online now. So, I can definitely just do it over the Web. I really like doing research over the Web. You can get news on a bunch of companies and not feel like you're paying for anybody's time."

We learn several things from this conversation. These users care about easy access to get information and make trades (facilitation). They are not looking for advice from a broker. They are repeat users, although some would use the system far more frequently than others. Some would call from their cars, others from home.

We now have important information about who the customers are and what they want to see (or hear) in a service. We will do two things with this information: Feed it into the high-level design and use it to help make design tradeoffs during detailed design.

To help us further understand the customer, we ask the client whether we can hear some live interactions between brokers and customers. We want to listen to the language that is being used and the order in which information is shared between the two. We also want to get a sense of how knowledgeable these traders are and how much help the brokers provide them.

In the course of a day, we work with four brokers and listen in on eight to twelve calls with each one. We also ask them questions to get their view of the needs and concerns of their callers. A few trends emerge: Callers always ask for a few stock quotes before placing trades. Often, they ask not only about the company they are trading but also about other companies in the same industry. Before placing a buy order, they usually check the cash balance in their account to help decide how many shares to buy. The brokers always carefully confirm the details of a trade before placing it. If the caller sounds at all unsure (e.g., "Um, yeah, I guess that'll do"), the broker asks for a more definitive confirmation, if necessary repeating the details of the trade. Occasionally, a caller does not know his or her account number. In that case, the broker solicits some personal information, such as name, address, and mother's maiden name, to access the account.

7.2.3 Understanding the Application

Given everything we know about the end users and everything we know about the client's motivations, we work with Lexington on a complete definition of the functionality. It seems clear from customers that gathering and monitoring information are key. They want to get up-to-the-minute stock quotes, updates on their portfolios, account information, and even financial news. Lexington

emphasizes the need to incorporate personalization features into the application to make it sticky. We agree on three personalization features: personal watch lists (lists of companies callers can get quotes for in a single request), a portfolio performance summary (account value and quotes for companies callers own), and account settings such as default accounts for balance inquiries.

Trading of stocks and mutual funds is clearly needed. Other features that go hand-in-hand with placing trades are learning about executed orders, reviewing open orders, and canceling and changing orders. A number of other features allow the customer to manage money and get account information. They include transferring money between accounts, withdrawing money, and requesting copies of monthly statements.

We compile a list of all the features (see Table 7-1) and consider what information the caller will need to supply to the application and the type of information it will provide to the caller.

The client agrees on this list of features. We discuss a feature that allows the caller to request news headlines and reports for companies. Given the complexity of this feature and the desire to deploy quickly, we agree to delay it until a later version. Similarly, we decide to wait on speaker verification (as a replacement for PINs) until a later version. Any other requests for new features will also be considered for a later version.

Table 7-1. Functionality of the Lexington Speech System

FUNCTIONALITY	INFORMATION SUPPLIED BY CALLER	INFORMATION PROVIDED BY SYSTEM
Log caller in	Account number and PIN	Acknowledgment of successful login
Get quotes for stocks, indices, and funds	Company, index, or fund name (some popular ticker symbols)	Company name, price, up/down amount (details will be considered in design phases)
Create personal watch list	Company, fund, or index name	Confirmation that name was added to list
Check personal watch list	Watch list command	Same as quote information
Check personal portfolio	Commands such as quotes (on all portfolio companies), balance, holdings, net worth, and available cash	Amount of balance, company or fund name, number of shares held and current value, amount of net worth, amount of available cash

FUNCTIONALITY	INFORMATION SUPPLIED BY CALLER	INFORMATION PROVIDED BY SYSTEM
Trade stocks and funds	Company or fund name, number of shares to trade, type of order (buy/sell), price, expiration, all or none, price for stop orders	Price at which the stock or fund is currently trading, confirmation of order, order number, instructions on how to change or cancel order
Learn about executed orders	Executed order command	Company or fund name, buy/sell, number of shares, amount
Review open orders	Open order command	Company or fund name, buy/sell, number of shares, amount
Cancel orders	Cancel order command, name of company or fund, identification of correct order	Confirmation that order was canceled

FUNCTIONALITY	INFORMATION SUPPLIED BY CALLER	INFORMATION PROVIDED BY SYSTEM
Change orders	Change order command, name of company or fund, identification of correct order, new order information	Confirmation that order was changed
Get account balance	Name of account	Cash balance in account
Transfer to a broker	Transfer request	Confirm request before transfer
...		

Before our next meeting with Lexington, we review each feature for recognition complexity. The only recognition challenge we identify is account numbers, which are 12-digit strings. We want extremely high recognition accuracy for account numbers, because that is the first item requested by the system when callers log in. We want that interaction to be seamless. This issue is put on the agenda for our next meeting.

At this point, we bring in the technical members of the team at Lexington and talk about the application environment. The system will need to connect to data feeds

that supply quote information. It will also require access to the database that stores user account information and to the backend transaction services. Additionally, there will be an interface with the CTI (computer telephony integration) software to pass caller information to a broker when a call is transferred.

We discuss the recognition of account numbers. Lexington informs us that the last digit of the account number is a checksum. We agree to use N-best recognition. The application will test each item on the N-best list for a valid checksum and, if it is valid, will check the account database to see whether it is an existing account number. The account number chosen will be the one having the highest recognition confidence that has a valid checksum and exists in the database. (See Chapter 13 for more details on this and other approaches to solving tough recognition problems.)

The tech team gives us copies of the specifications for all the data feeds and the CTI system. This documentation details all the inputs and outputs for the transactions and all the failure modes (all failure messages that may be returned to the application). We review these together to make sure we have all the details we need. For each integration, they provide figures on average latency.

A summary of everything we have learned from the requirements process, including an exhaustive feature list and the success metrics, is written up in a formal requirements definition document. Lexington quickly signs off on the requirements definition. Given how closely we have been working, there are no surprises.

It is now time to move to the next design phase, high-level design. We take a step back to reflect on what we have learned from the requirements phase. We now have a thorough understanding of who the callers are, how they

view themselves, and what features they want in a service. We also know about the business's perspective: what its motivation is, the key features that will help the business case, the corporate image it wants to promote, and its rollout plans. We have mapped out the service's functionality, and we have prepared a list of the databases and data feeds that must be integrated with the system. Although it has taken a bit of time to collect all this information, the time has been well spent: Our design decisions going forward will be well grounded, and we will be able to look back at our finished product and accurately assess whether the goals we have set out to achieve have been met.

7.3 High-Level Design

It is now time to roll up our sleeves and dig into the design of the application. However, we want to start at a high level before drilling down into the details. The goal is to create a framework for the detailed dialog design that is based on the requirements. We also want to make some design choices up-front so that we can apply them consistently throughout the application.

7.3.1 Key Design Criteria

To come up with key design criteria, we review what we have learned about the users and about the business. The target customers want convenience, flexibility to do the things they normally would need a broker to do, efficiency, and the knowledge that any information they receive or transactions they perform are accurate. The business wants to automate calls and provide a system that is easy and satisfying for their customers to use. The key design criteria are as follows: simplicity, accuracy, efficiency, and flexibility. We will refer to these design guidelines as we make design decisions.

7.3.2 Dialog Strategy and Grammar Type

At this point, we can start to determine which high-level dialog strategies we should use in order to meet the guidelines and requirements we have defined. Given the priority of efficiency and the expectation that many users

will call frequently, we want to make sure that experienced callers can rapidly and flexibly state their requests. We decide on a mixed-initiative dialog structure, allowing experienced callers to access system functionality directly from the main menu and make complex requests (e.g., "Buy one hundred shares of Cisco at the market").

Allowing the caller to control the initiative at the beginning of the conversation is not without its drawbacks. An extremely open-ended prompt caters to expert users. Novice users will need more guidance. We decide to track callers across sessions via their account numbers. On the first call into the system, we provide the instruction needed for the novice user, meeting our goal of simplicity. Then, after callers have become familiar with the system, we will put the initiative into their hands (promoting efficiency and flexibility). Whenever callers experience difficulties, we will back off to a more directed dialog. If callers remain silent, we will offer additional instruction to remind them what they can do. In this way, the initiative shifts between the caller and the system depending on how smoothly things are advancing.

One impact of this design decision is that we will use a statistical language model, as well as rule-based grammars in modules that perform more constrained tasks.

7.3.3 Pervasive Dialog Elements

At this point we make a few other high-level design decisions related to universals, error recovery, confirmation strategies (making sure recognition is correct), and login strategy. With regard to universals, we will incorporate the six standard ones: help, operator, repeat, main menu, go back, and good-bye. We add one more universal: At any time, the user can ask to speak to a broker. This helps

ensure that we will not lose callers, especially if they want to make a trade but are not yet willing to do so with the automated system. We will make sure to remind callers of the availability of brokers in some of the help and error messages.

Next, we define our error recovery strategy. At the initial, open-ended prompt, if we do not hear a response within a few seconds, we will explicitly list the choices available. Similarly, for other errors that occur in the main menu state, we will respond by listing some general commands that the caller can say; these commands correspond to the major functional capabilities of the application. Once the caller is in a more constrained interaction (e.g., the system has just asked "How *many* shares do you want to buy?"), a rapid reprompt will be used (e.g., a quick prompt such as "I'm sorry?") at the first error, followed by more detailed error prompting on subsequent errors. Additionally, error and help prompts will let a caller know how to return to the main menu and, at select times, how to connect directly to a broker.

The application will also have several mechanisms for helping those callers who are experiencing multiple errors and are at risk of failing to complete their task. We prefer to send these callers to someone who can help them rather than allow them to become frustrated and hang up. We will remind them about the ability to transfer to a representative if they are having trouble entering account numbers or PINs. In addition, on the third error in any given dialog state, we will ask callers if they want to transfer. In addition to dialog state error limits, we will count the number of times the caller negatively confirms information (for example, when placing a trade). If the caller states three times that the information is incorrect, the system will offer to transfer to a broker.

Confirmation will be an important element in the application because we want to ensure accuracy without slowing down the dialog. Trades, of course, will need to be explicitly confirmed, as will other transactions such as canceling an order. However, when the caller is simply receiving information such as a stock quote, the system will offer only implicit confirmation (e.g., "*Apple* is selling at …").

Another pervasive dialog element is login. There are two approaches to consider: (1) to require login at the beginning of the application for all users, even those who want only to get stock quotes, or (2) to log in users only when they reach a secure area of the application that requires account information, such as placing a trade. This design decision needs to be made before any detailed work begins because the context is quite different between the two scenarios, resulting in different call flows and different prompting. We decide on the first option: logging users in up-front. This will allow us to adjust the prompting depending on the callers' usage patterns. The approach will meet the design criteria of simplicity for novice users and efficiency for expert users.

7.3.4 Recurring Terminology

We want to be sure that there is a seamless coordination of information between the Web site and this system, so we use the general components of the Web site as a starting point for the main components of the speech system. The functional areas include *quotes, trading, portfolio*, and *account information*. These will become terms callers can use to navigate to the different areas of the application. In this way, callers will be able to take the mental model they

have developed for the Web site and generalize it to the speech system using consistent terminology.

In addition to the main functional areas, we create a list of terms to promote consistency. For example, we decide to call the list of stocks maintained by the caller a "personal watch list." By establishing these phrases up-front, we reduce the risk of introducing inconsistencies in the detailed design.

7.3.5 Metaphor

We begin thinking about the metaphor by considering the style and character of the application. First, we define the service. It is a resource for customers of a brokerage firm that provides up-to-the-minute information and the ability to perform transactions from callers' personal accounts. The relationship between the caller and the service is similar to that of a trader interacting with a broker. In fact, if we recall from the focus group, services like these are beginning to replace interactions with brokers for some customers. Considering this, we choose the overarching metaphor to be a conversation with a live stockbroker.

In choosing this metaphor, we do not want to fool the caller into thinking that the system is a live person. We want only to take advantage of the caller's existing mental model of how to interact with a broker. For example, we would like to assume that the caller has a model of what information to provide in order to place a trade.

7.3.6 Persona

Next, we design the system persona. We think about what this stockbroker should be like. The persona needs to embody the service and also appeal to users, so we must take many elements into consideration. To help us, we use the persona-design checklist from Chapter 6.

Metaphor and Role

The persona should convey the essence of a professional broker.

Brand and Image

We want the persona to reflect the corporate image of the business: knowledgeable, dependable, helpful.

End Users

- **Target audience:** We decide we want the persona to sound about the same age as most customers (mid-30s) and reflect the language use of an individual of that age with a middle to high income.
- **Frequency of system use:** Typically, customers will be talking to the system repeatedly, so we want to make sure that the personality is engaging and retains its appeal over time.
- **Mind-set of the user:** The mind-set of the caller will be on managing money quickly and accurately. Creating a chatty or extremely laid-back persona would be a misguided design decision given what we know

about the people interacting with the service. The persona should be quick, to the point, and helpful.

Application

- **Content:** In the application, callers will be asking for information about stocks, making financial transactions, and managing their accounts. The persona should convey an individual who is knowledgeable about the stock market as well as professional and capable. We need to inspire confidence that the information callers are receiving is accurate and that the transactions are in good hands.

- **Task-related issues:** Because callers may be asking for numerous stock quotes in a row, we must be sure that the persona is efficient but clear. However, he or she must be ready to slow down and give careful instruction if the caller is having a problem.

Based on these guidelines, we create the persona definition shown next.

Frank Delano

Frank Delano is an ambitious, 35-year-old New York stockbroker. Raised on Long Island, he became interested in the market in high school. He likes investing and learning about new markets. After majoring in business at Fordham University, he joined a prestigious brokerage house, where he now specializes in Internet and technology stocks. When not working, Frank is a mountain biker and skier.

Characteristics:

Knowledgeable (an expert in the field)

Capable (smart, competent, and accurate)

Dependable (follows through)

Helpful (patient, not abrupt; offers help but does not give advice)

Honest (credible and trustworthy)

Efficient (works quickly but is not curt; keeps the conversation moving)

Energetic (very positive and determined)

Vocal Attributes:

No detectable regional accent

Diction is precise and professional but at ease

Pace is slightly quicker than average, although he slows down to give instructions or help the caller who is encountering problems

Linguistic Attributes:

Standard American English

Vocabulary is simple without too much technical jargon

Uses abbreviated phrases instead of complete sentences where appropriate

The character traits in the persona definition are well aligned with the image the business wants to convey. They also embody the type of broker whom customers want to interact with: someone who will provide them with accurate information and is capable of executing transactions efficiently and without error.

We present the persona definition to our client. Because the persona influences many aspects of the design, we want to make sure that the business accepts it before we move forward. Lexington Brokerage is happy with the profile. The executives admit that they have never gone through the persona design process and are excited to see how the speech application will be different from the look and feel of their touchtone system. We will begin the process of selecting a voice actor right away. Although that is part of the production phase, it is a good idea to start early. Chapter 17 describes in detail the process for selecting voice actors.

7.3.7 Nonverbal Audio

Given that our metaphor is a trader having a conversation with a stockbroker, we decide to create a sparse audio environment that consists mostly of the stockbroker's voice. For design elements such as landmarks and latency, we will use language to communicate to the caller what is happening in the application. We will, however, use Lexington Brokerage's branding sound at the very beginning of the interaction (just before the welcome prompt). This is the same sound that appears in the current touchtone system.

7.4 Conclusion

This brings us to the end of the definition phase of the project. We have a detailed definition of requirements, driven by an understanding of the business, the users, and the application. We have made some important design decisions up-front in order to create a framework for detailed design and achieve unity and consistency throughout the application. We are now ready to move into the detailed design phase of the project.

The next part of the book covers the methodology for detailed design, discusses a set of design principles to use in making detailed design decisions, and presents the detailed design steps for the Lexington Brokerage project.

Part III: Design Phase: Detailed Design

Chapter 8. Detailed Design Methodology

We are now ready for the detailed design, the stage when the designer (or design team) pulls together all the details into a coherent, cohesive design. Skilled practitioners are able to immerse themselves in the details, crafting stretches of dialog, while still keeping the larger context and goals in mind. The requirements and high-level design steps we have completed at this point facilitate this ability.

As we go through the detailed design steps, you will see many of the methodological principles, first presented in Chapter 3, applied. We place significant emphasis on end-user testing, although the goals and approaches here are different from those in the definition phase. In this phase, you will see a number of process steps that help focus your design decisions on conversational design and consideration of appropriate context.

In addition to the methodological principles described in Chapter 3, this chapter describes a set of design principles for VUIs. These guide you when crafting dialog strategies, call flows, and prompts. Some of these principles are common to all types of user interfaces, whereas others are specific to voice user interfaces.

Detailed design involves the creation of a complete description of the call flow and all the prompts played by the system. The outcome of this phase is a detailed design document, and possibly a prototype.

This chapter covers the following:

- Anatomy of a dialog state
- Call flow design
- Prompt design
- User testing
- Design principles

8.1 Anatomy of a Dialog State

The smallest unit typically represented in a call flow diagram is a single dialog state. Most often, a **dialog state** involves a single interchange between the caller and the system. However, if the system handles unexpected input, it may reprompt the caller within the same dialog state. Here's an example:

SYSTEM: How much do you want to transfer?

CALLER: <unrecognized speechrecognizer returns reject>

SYSTEM: Sorry, *how* much do you want to transfer? [rising intonation]

In this case, the system is still in the same state, still trying to elicit the transfer amount, and using the same grammar to try to recognize the input.

For each dialog state, you must describe a number of components. A typical case includes the following:

- **Initial prompt(s):** This is the prompt that is played when the conversation first reaches this dialog state. As you will see in section 8.3, it is often important to define more than one possible initial prompt, depending on recent history (e.g., which state preceded the current one). Section 8.3 describes the methodology for defining prompt wording.

- **Recognition grammar:** Although the full development of the recognition grammar is saved for a later step, it is appropriate to create a high-level definition of the grammar as you define a dialog state. This definition will be used as a guide by the grammar developer. The high-level definition should describe the specific items of information to be returned from the grammar (e.g., the destination city for a travel application), possibly by specifying the names of slots the grammar can fill. Additionally, it is often useful to provide a number of sample expressions to give the grammar developer an idea of the range of expression expected. For example, if you provide sample expressions for a yes/no grammar such as "Yup," "Okay," and "Yeah, that's what I want," the grammar writer will be aware of the need to flesh out the grammar to cover the wide variety of ways callers may express themselves to indicate yes or no.

- **Error handling:** Many things can go wrong during recognition. The caller may say something not covered by the grammar, or background noise may prevent accurate recognition. In both these cases, the recognizer is likely to return a reject. The caller may not respond at all, in which case the recognizer may return a no-speech timeout. For each type of error message that the recognizer may return, the designer must specify appropriate handling (such as the reprompt, "Sorry, *how* much do you want to transfer?" in the earlier example). Chapter 13 discusses error handling in detail.

- **Handling of universals:** Chapter 5 describes universal commands. Occasionally, a universal must be overridden in a particular dialog state.

- **Action specification:** There are a number of actions that may happen during execution of a dialog state that should be specified. For example, the application may access a backend database or other system. If so, you must specify handling for all possible failure modes as well as for success. Additionally, you should specify transitions to other dialog states along with any logic or conditions for the transfer (the call flow diagram, shown in a moment, is a convenient place to specify transitions).

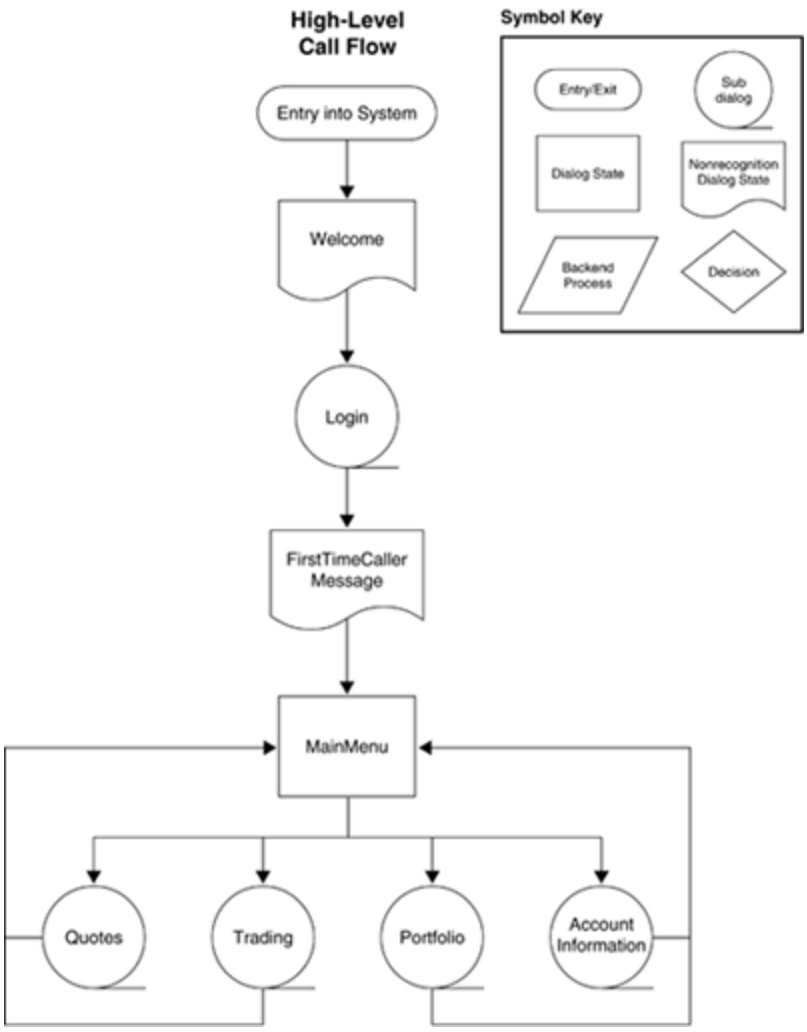
These are the components of a typical dialog state. The next two sections cover the methodology for creating the two primary elements of a design: the call flow and the prompts.

8.2 Call Flow Design

When you design the call flow, you must think through all the design strategies in complete detail: how the caller will navigate through the system, how menus will be structured, how information will be presented to the caller, where NVA will be played, where other systems will be integrated, how failure modes will be dealt with, and so on. Chapters 813 have extensive coverage of design guidelines and specific dialog strategies.

Figure 8-1 shows an example of a high-level call flow diagram for the Lexington Brokerage application (Chapter 14 shows, in detail, the development of call flows for this application). The example shows several types of states. The Main Menu state is a dialog state of the type defined in section 8.1. It consists of a prompt, a grammar, error and universal handling, and, depending on what was recognized, transitions to a following state. `Welcome` and `FirstTimeCallerMessage` are **nonrecognition states**: They play messages but do not perform recognition. Login is a placeholder for a **subdialog**: a detailed dialog that is fleshed out in a later diagram. Following Main Menu are four subdialogs, each representing one of the four major functional areas of the application. Each of these is also a subdialog, fleshed out in later diagrams.

Figure 8-1. High-level call flow diagram for the Lexington Brokerage application.



Note that you do not flesh out the wording of prompts and messages when you design the call flow. That is part of a later step, to be covered later in this chapter. The states of the call flow diagram are typically given descriptive labels so that the intended meaning of the prompts is clear.

Follow these guidelines in determining the structure of your call flow diagrams:

- Choose a structure that represents the natural logic of the application. Subdialogs should be meaningful units of functionality.

- Use informative names for states. If necessary, add notes or even placeholder prompt wording in order to make the functionality clear.

- Choose a set of conventions (e.g., shapes, as in Figure 8-1) and use them consistently to indicate the various types of states.

- Let dialog states be the lowest-level unit on the call flow diagram. The internal details of each dialog state (e.g., error handling) should be defined in the dialog spec.

8.3 Prompt Design

When you design prompts, you must communicate clearly to the caller, appropriately capture the persona of the application, keep the caller engaged, maximize caller success and comfort, and avoid ambiguities. Much of the look and feel of the application will depend on the wording of prompts. Chapter 10 covers many of the specific guidelines and lessons about prompt wording. Here we discuss the methodological issues.

There are well-understood methodologies for writing prompts in a way that will lead to natural-sounding, comprehensible communication. The single most important point is to bring the design experience close to the user experience. This is what we mean by conversational design, one of the basic methodological principles discussed in Chapter 3. Conversational design has two aspects, both of which grow out of the fact that you are designing a *spoken language* experience for your callers. First, you should design prompts in their conversational context, and second, you should experience the prompts in an auditory mode.

8.3.1 Conversational Design

The key to conversational design is to think about each prompt in its **conversational context**. You do this by writing sample dialogs that traverse many of the possible paths through the application. A sample dialog is an actual conversational sequence, specifying both the system side and the caller side of a conversation. By working out the prompt wording through the creation of sample dialogs, you will be thinking of the prompt wording in context. You

should craft a large set of sample dialogs, covering prompts throughout the application. Some of the sample dialogs should cover error recovery and help prompts.

Many current tools encourage an approach to prompt design that involves a complete specification of all the prompts related to one dialog state (e.g., initial prompt, error prompts, help prompt), followed by a specification of all the prompts related to the next state, and so on. This approach often leads to stilted-sounding applications because the prompts are not conceived in context. Creating sample dialogs can solve the problem.

The following is a real example of two versions of a bill-paying application. The first version was written with a standard tool, going state-by-state, writing all the prompts for each state (i.e., not in conversational context).

(1)

CALLER: Bill payment.

SYSTEM: What bill would you like to pay?

CALLER: MasterCard.

SYSTEM: How much would you like to pay?

CALLER: Two hundred and fifteen dollars.

SYSTEM: What date would you like the bill to be paid?

CALLER: August first.

SYSTEM: Confirming: You'd like to pay MasterCard in the amount of two hundred and fifty dollars on August first. Is this correct?

CALLER: No.

SYSTEM: Which part would you like to change?

CALLER: The amount.

SYSTEM: How much would you like to pay? [same recording as first-time request]

We were called in to review and then rewrite the prompts for the system. We created new prompts using sample dialogs. The following was the result for the same path through the application:

(2)

CALLER: Bill payment.

SYSTEM: Sure, "bill payment." What bill would you like to pay?

CALLER: MasterCard.

SYSTEM: How much would you like to pay?

CALLER: Two hundred and fifteen dollars.

SYSTEM: And on what date would you like it paid?

CALLER: August first.

SYSTEM: Let me confirm that: You'd like to pay MasterCard in the amount of two hundred and fifty dollars on August first. Did I get that right?

CALLER: No.

SYSTEM: Which part did I get wrong?

CALLER: The amount.

SYSTEM: Sorry about that. What's the *correct* amount?

Clearly, this is not a scientifically valid comparison of the two techniques, given that different designers were involved. However, it is instructive to look at the nature of the differences between the two examples. Notice how much better the second version flows (try reading them both aloud). Notice the extra words, which help relate a prompt to what came immediately before it (e.g., "Sure, ...", "And ...").

It is especially instructive to compare the final prompt in each example. In both cases, the final prompt is resoliciting the *amount* to be paid. In the first example, the final

prompt, "How much would you like to pay?" is merely a reuse of the recording used the first time (every time callers enter the Amount state, that is the prompt played). In the second case, a different prompt is played, given that the Amount state is entered following a disconfirmation (it is entered in order to *re*solicit the amount, after an error). In this case, the prompt is written in a way that shows more conversational awareness. To begin with, the first clause of the prompt functions as an apologetic acknowledgment. Furthermore, the word "correct" is given informational focus through emphasis.

The difference in the final prompts is an example of a general guideline for prompting: To account for conversational context, it is often necessary to provide multiple possible initial prompts in a dialog state. The choice of which initial prompt to play should be conditioned on the immediate dialog history. You should always check all dialog histories for a state (e.g., all possible predecessor states) and determine which ones would benefit from alternative versions of the initial prompt. This need will become obvious if you design your prompts through sample dialogs.

The improvements in prompting based on sample dialogs bring value not only because the resulting conversation is more cohesive, but also because it is more comprehensible. Chapter 10 covers, in detail, the issues surrounding the crafting of prompts.

Sample dialogs afford the designer a view of otherwise discrete messages and prompts with the full benefit of context, as an interaction between the caller and the system's persona. Designing your application via generous sample dialogs ensures that the design process closely parallels the user's experience. It is hard to imagine how a

persona-rich application might be created without the illuminating perspective afforded by sample dialogs.

8.3.2 Auditory Design

The caller is engaged in a spoken language interaction. As discussed in Chapter 10, spoken language is fundamentally different from written language. Given the differences between spoken and written language, humans have very different expectations when reading than they do when hearing language.

To be sure that you have an accurate view of your prompts, it is essential to listen to them. We strongly recommend saying your prompts out loud as you craft them. Additionally, it is extremely valuable to read through sample dialogs with a colleague, one of you providing the voice of the system and the other the caller. Even very experienced designers will react differently when they hear their prompts actually spoken.

We have had the experience a number of times, when working with customers, that they react negatively to a dialog spec as they read through the prompts, often complaining that they are too "informal." However, if we present customers with prerecorded sample dialogs, those same prompts often sound fine to them.

Using the two techniques described in this sectioncrafting prompts through the creation of sample dialogs, and reading those dialogs aloudyou can move your design experience closer to the way the user experiences your system. We have observed significant improvements in the quality of prompts when designed in this manner. Using this approach will help you use your own innate conversational capabilities as you craft prompts.

8.4 User Testing

As we discussed in Chapter 3, designers are, in some ways, the worst people to judge the usability of their designs. Given their design role, they are thoroughly aware of how to use the system. End-user testing is key to evaluating and refining design decisions.

The previous two sections describe call flow and prompt design as two steps in the design process. The third important step is end-user testing. Keep in mind that these steps do not happen in strict sequence. A typical project goes through a number of iterations of testing and design refinement. One common approach is to begin by designing the primary paths through the dialog and do some early user testing. It is also advisable to design the riskiest parts of the system (from a usability point of view) early, leaving time for iterative testing and refinement.

The primary means of user testing at this phase is formal usability testing, as discussed in the next section. We also cover card sorting, a technique that helps you design menu hierarchies.

8.4.1 Formal Usability Testing

Usability testing merits a full book of its own. Luckily, a number of excellent references are available. We recommend Rubin (1994), Dumas and Redish (1999), and Nielsen (1993). In this section, we review at a high level the basics of usability testing and point out some of the special issues that arise in testing voice user interfaces.

Basic Approaches

Usability testing should begin early in the design process. One common approach, which enables early testing even before a prototype exists, is referred to as a **Wizard of Oz** (**WOZ**) test (Fraser and Gilbret 1991). The key idea is to simulate the behavior of a working system by having a human (the "wizard") act as the system, performing virtual speech recognition and understanding and generating appropriate responses and prompts. In some cases, the wizard actually says the prompts with his or her own voice. In other cases, a software system allows the wizard to choose the appropriate response from prerecorded speech files. The latter approach is preferable because it leads to more consistent system behavior across test participants.

The WOZ approach has a number of advantages:

- **Early testing:** You can begin testing as soon as you design the first pieces of dialog you wish to test.

- **No bugs:** A WOZ system is not subject to software and integration bugs. If you test using an early version of a working system or a prototype, you run the risk of hitting bugs that will interfere with your ability to test usability.

- **High grammar coverage:** Grammar coverage is a key ingredient of recognition performance. Given that it can take substantial effort to develop grammars, it is unlikely that an early prototype will have high grammar coverage. Poor grammar coverage will interfere with the ability to test usability (because users will experience many recognition errors that do not reflect the actual recognition performance you ultimately

expect to achieve). A WOZ, by virtue of using a human speech recognizer, has high grammar coverage. (Note: That can sometimes be a disadvantage, because a human will have higher grammar coverage than the ultimate system will have, but it is difficult to precisely simulate the behavior of the grammar before it is developed.)

- **Quick and easy changes:** It is far easier to change a WOZ than a working prototype. After a day of testing, it is often desirable to make some quick changes and be ready to run more participants the following day.

The primary disadvantage of a WOZ is the difficulty of simulating realistic recognition accuracy and grammar coverage.

Another approach is to run usability tests using a prototype. Typically, you do this later in the cycle than WOZ testing. The main advantage of a prototype is realism. The behavior of the system is likely to more accurately reflect the behavior of the final system than a version simulated by a human (assuming that you have reasonable grammar coverage and no destructive bugs).

Usability tests have traditionally been run in usability labs. Participants come to the lab, perform various tasks, and are interviewed by the moderator. However, usability tests of VUIs are increasingly being run over the telephone, with the participant at home. Telephone-based usability testing offers a number of advantages:

- You can more easily reach a geographically dispersed pool of participants, thereby possibly getting a sample that is more representative of the caller population.

- The participants are at home, in an environment and state of mind that may be more representative of real system use.

- It is less expensive than bringing participants into a laboratory and less inconvenient to participants.

The advantage of running a usability test in a lab is the ability to see the participant. Often, you can read body language that indicates the caller is unhappy or confused, something you miss if you are listening over the phone.

We run many of our usability studies using a Wizard of Oz system over the telephone. This allows us to get quick, early feedback inexpensively. Chapter 15 describes evaluative usability tests, which are run after a complete working system has been created. They are run using the real working system.

Task Design and Measurements

In a typical usability test, the participant is presented with a number of tasks, which are designed to exercise the parts of the system you wish to test. Given that it is seldom possible to test a system exhaustively, the tests are focused on primary dialog paths (i.e., features that are likely to be used frequently), tasks in areas of high risk, and tasks that address the major design goals and design criteria identified during requirements definition.

You should write the task definitions carefully to avoid biasing the participant in any way. You should describe the goal of the task without mentioning command words or strategies for completing the task.

In addition to the subjective measures to be described later (e.g., the perceptions and opinions of the participant), a number of performance measures will require tracking during the test. Performance measures should include task completion (does the participant successfully complete the assigned task?) and efficiency (does the participant take the most direct path to completion or end up going through error recovery procedures, restarts, and help commands?). The participant's specific path through the dialog should be noted, and success or failure of various error messages and help messages should be recorded. Additional performance measures should cover, where possible, measurements identified during requirements definition as success metrics. Any other measures that can be used to indicate relevant performance issues and provide design guidance will also be useful.

Selecting and Recruiting Participants

A typical test includes 1015 participants. The most important rule of thumb about participant selection is that the participants must be representative of the ultimate end-user population. One way to enlist participants is through a recruiting firm. These firms typically have huge lists of potential participants, with significant demographic and other information about each person. You can define criteria for the participants to meet. Recruiting firms typically charge a fee for each participant they provide. Additionally, it is often useful to provide a financial incentive to participants.

In some cases, the company deploying the system can provide you with a list of its customers who can act as usability study participants. This can be very useful, especially if the system targets a specific group, such as users of the Web site. Whether you are using a recruiter or

getting suggested participants from the client company, you should create a screening questionnaire that includes questions about demographics, level of education, experience using various systems, and any other criteria that are relevant to the test. The results of the questionnaire can be used to narrow the choice of participants.

Running the Test

Before running the test with your first participant, test it with one or two people to make sure everything is working and that the instructions and task descriptions are clear. When you carry out the tests with the real participants, either audiotape them (with their permission) if the test is conducted over the telephone, or videotape them if they are in a usability lab.

Begin by describing the purpose and nature of the test. It is important to assure the participants that they are not being tested. Rather, the goal is to get their help assessing and improving the design of the system. Assure the participant that you are not vested in the quality of the current system but are trying to find any problems. Therefore, you will not be offended by negative feedback.

It is often useful to do some debriefing with the participant after each task, in addition to a more general debriefing at the end. While subjects perform the task, note issues and problems that arise so that you can ask questions later to try to understand how they were thinking. Don't interrupt and help has soon as you see a participant having trouble with a task. Often you will learn the most by listening to how people react in those situations.

The general debriefing, when they have completed all tasks, usually consists of a questionnaire designed to elicit their subjective reactions, followed by a more open-ended discussion of their experience with the system. The questionnaire consists of a number of statements about their experience, with a rating of 1 to 7 indicating how much they agree or disagree with the statement (where 1 is "strongly disagree" and 7 is "strongly agree"). These are referred to as **Likert scales**. Some of the statements may be general, such as, "The system was quick and efficient," whereas others may be specific to the features tested. Questionnaires may also include questions requesting short answers and comments.

The questionnaire is typically followed by a more open-ended discussion. Begin with a general question, such as, "What did you think?" From there, drill down into specifics. Try to learn about how participants were thinking, what their mental models were, what assumptions they had about how the tasks should be performed, and so on. Focus on understanding problems and issues for the user, not on coming up with solutions.

Analysis of Data

The purpose of data analysis is to identify and prioritize problems. Often, problems with an interface will surface in a number of ways during a test. For example, a lack of clarity about how to perform a certain task may result in numerous error messages and requests for help, lack of task completion for some participants, lower Likert scores for certain questions, and negative subjective feedback.

Results should be compiled. Responses to Likert statements should be summarized, showing individual values and means, as well as the mean across all Likert statements.

For each task, you should compute completion rates and use of error recovery procedures and help; summarize this information in a table, along with summaries of problems with the tasks and comments from participants. Similar participant comments should be grouped and noted in a table, with counts.

Part of the purpose of analyzing the data is to look for trends. Sometimes, a basic underlying problem will have symptoms that show up in a number of places. For example, a problem with the general error recovery strategy may result in problems in a number of tasks. In that case, you want to fix the general problem rather than deal with each of the symptoms.

You should prioritize problems with a number of issues in mind:

- **Scope:** Is the problem local (e.g., an ambiguous prompt), or does it affect many parts of the application (e.g., an ineffective error strategy)?
- **Frequency:** How many participants had the problem?
- **Recoverability:** When participants experience the problem, do they fail to complete the task, or do they ultimately recover and successfully complete the task?
- **Success metrics:** How much will this problem keep you from meeting the success metrics agreed upon in requirements definition?

Proposals to fix each problem should be worked out with the designer. Otherwise, you risk making suggestions that do not work in the larger design context.

8.4.2 Card Sorting

Card sorting is a technique used to gain insight into how people categorize information and to find out what labels they use for the categories (Balogh 2002). It is a useful technique when you design applications having complex menu hierarchies. The results can help you design an intuitive menu structure and choose useful names for menu choices.

There are two primary approaches to card sorting tests. In the first approach, you give participants a set of index cards labeled with the names of various items. The participants' job is to sort the cards into groups of items that go together. It is a good idea to limit them to some maximum number of groups. When they have finished sorting, have them create a name for each group, and ask them to explain their thinking behind the groups chosen.

Here are examples of the types of items that may appear on the index cards for a hypothetical telecom application:

I'm having a problem with my handset.

I want to add call forwarding to my service.

How much do I owe on last month's bill?

I don't remember my voice mail PIN.

How does call waiting work?

Can I check my balance?

I need a duplicate bill.

I'm being charged for calls I didn't make.

I'm moving, and I want to cancel my service.

I'm not getting a dial tone at home.

Do you have a voice dialing option?

I was in the middle of a call, and it got dropped.

The second approach is similar to the first, except that you begin with a set of existing category names and ask the participants to sort the items into those categories. This is a good way to test an existing design for a menu structure.

8.5 Design Principles

Based on analyses of data from calls to deployed systems, usability studies of deployed systems and systems in development, and carefully controlled laboratory studies, six core VUI design principles have emerged. These principles can be applied to optimize the design of dialog strategies, call flows, and prompts:

- **Minimize the cognitive load:** Minimize the short-term memory load on callers, the complexity of concepts that callers must understand, and the number of things they must learn in order to use the system successfully. Chapter 9 covers techniques for controlling cognitive load.

- **Accommodate conversational expectations:** Human-to-human conversation is guided by conventions and unconscious expectations. These same expectations are in effect when humans converse with machines (Reeves and Nass 1996). Adhering to these natural expectations not only helps an application flow more easily but also increases comprehension. Chapters 10 and 11 explain how to design prompts based on conversational principles.

- **Maximize efficiency:** Callers want speed and efficiency. The fewer the number of steps your dialog design requires, the greater the perceived efficiency of the phone call. Chapter 12 discusses efficiency.

- **Maximize clarity:** Clarity is always key. This applies to low-level details, such as clarity about the meaning of

prompts, and high-level details such as clarity of the appropriate mental model for using the system. Chapter 12 covers clarity and methods of balancing efficiency and clarity appropriately when you must trade off one with the other.

- **Ensure high accuracy:** Recognition errors must be minimized, because they seriously degrade usability and caller confidence. Chapter 13 covers VUI design techniques for high accuracy.

- **Gracefully recover from errors:** When errors occur, callers often become confused and frustrated. If an automated system lacks quick and graceful recovery from errors, callers will refuse to complete tasks, either by hanging up or by requesting transfer to a live agent. Chapter 13 covers the types and causes of errors and describes techniques to help a caller seamlessly and gracefully recover.

Interestingly, these six principles, which have emerged largely from analysis of live data from phone calls made to spoken language systems, coincide closely with user interface design guidelines and principles developed in other fields of study, such as graphical user interface (GUI) design. See, for example, Shneiderman (1998) or Nielsen (1993). However, the way these principles are applied to VUI design is different from the way they are applied to other types of interfaces. Although the first principle (minimizing cognitive load) is an issue for all user interfaces, it is of special importance for auditory interfaces, given the cognitive challenges that result from their nonpersistent nature. The second principle (accommodating conversational expectations) is specific to voice user interfaces, although the accommodation of user

expectations in general is certainly important for all user interfaces.

8.6 Conclusion

The detailed design process presented in this chapter is driven by a number of the methodological principles discussed in Chapter 3. Approaches have been described for soliciting *end-user input* to support design decisions, considering *context* during design, and focusing the process on *conversational design*. We have also introduced a set of design principles that should guide design decisions.

Skill at applying the process will take experience in designing real applications. To illustrate the process in operation and to show some of the issues, decisions, and trade-offs that arise during detailed design, in Chapter 14 we continue the sample application for Lexington Brokerage, showing the detailed design phase. However, first we discuss the design principles in detail and show how they are applied to detailed design decisions (Chapters 913).

Chapter 9. Minimizing Cognitive Load

Cognition is the processing of information from the world around us. It includes perception, attention, pattern matching, memory, language processing, decision making, and problem solving. **Cognitive load** is the amount of mental resources needed to perform a given task.

All user interfaces make cognitive demands on users. Users must master special rules of system use, learn new concepts, and retain information in short-term memory. They must create and refine a mental model of how the system works and how they should use it. Systems that use purely auditory interfaces further challenge human memory and attention because they present information serially and non-persistently.

Successful user interface designs must respect the limitations of human cognitive processing. If a design requires the user to hold too many items in short-term memory or to learn a complex set of commands too quickly, it will fail. This chapter describes a number of guidelines for minimizing the cognitive load on users of spoken language interfaces.

There are three cognitive challenges you should consider as your design progresses:

1. Conceptual complexity: How complex are the new concepts callers must learn? How well do new mental structures match concepts and procedures that users are already familiar with?

2.

Memory load: How much information must callers hold in their short-term memory? How much new material (e.g., commands, procedures) must they learn?

3. Attention: Is it easy for the caller to attend to the most salient information? Will callers' attention be divided? If they are momentarily distracted (e.g., while driving), can they seamlessly continue their interaction with the system when they are ready?

The following sections discuss each of these potential challenges and present guidelines for handling them.

9.1 Conceptual Complexity

Conceptual complexity is, in part, a product of the new concepts the user must learn and the inherent complexity of those concepts. However, understanding the cognitive challenges goes beyond counting concepts and measuring their complexity. It is also a matter of understanding human capabilities (what is hard and what is easy for humans to understand and learn) and understanding the context in which users will operate (e.g., how the current application will mesh with existing user knowledge, skills, expectations, and mental models).

In this book we do not try to present a theoretical framework that allows precise prediction of the difficulty of particular design decisions. The necessary knowledge to create such a theory is incomplete. Instead, we present a set of guidelines that will help you minimize the cognitive challenges for your callers. This section covers the following guidelines:

- **Establish constancy:** Create **constants**, or universal commands, that are always available to the caller regardless of context. Universal commands should be designed to let callers recover from problems or receive help using the system.

- **Ensure consistency:** Do similar tasks in similar ways throughout the application. For example, whenever a caller traverses different kinds of lists in a large portal application, make the same set of list traversal commands available. In this way, you minimize the amount of new material the caller must learn.

- **Set the context:** Set the context for your callers. Make it clear why the system is taking specific actions. Appeal to their world knowledge and expectations in order to simplify new concepts they must understand.

9.1.1 Constancy

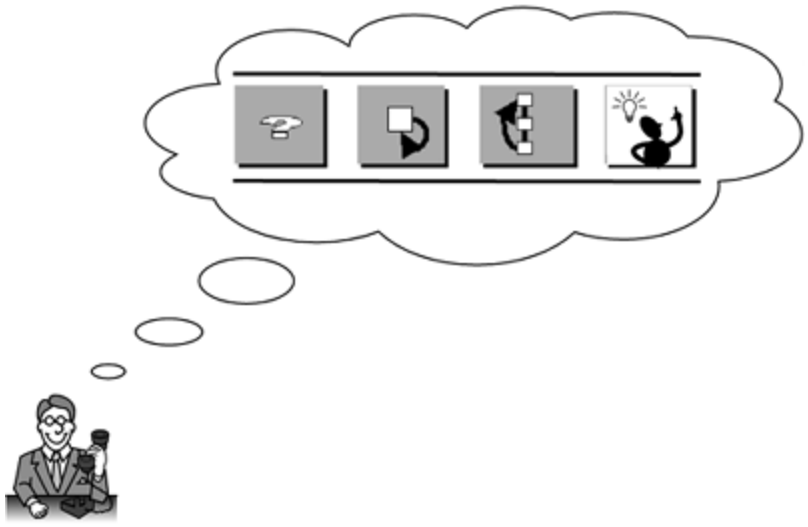
Graphical user interfaces take advantage of the ability to display information, often lots of it at the same time, on the computer screen. For example, many GUIs display a toolbar (see Figure 9-1), usually at the top of the screen. This toolbar, which typically consists of a row of icons representing actions, provides both a visual reminder of popular actions and a physical means to initiate them.

Figure 9-1. A GUI toolbar displays unchanging icons.

The toolbar is constant: It remains on the screen, and the icons never change. The toolbar's constancy reduces the user's need to memorize a set of actions and commands.

Similarly, VUIs can achieve constancy by using universals: a small set of spoken commands that are always available regardless of context (see section 5.2.2). After callers learn the universal commands, they can use them at any point as well as in future calls. These commands become, in effect, a mental toolbar of always available actions (see Figure 9-2).

Figure 9-2. Callers form a "mental toolbar" of spoken universals.



It is not practical to expect a caller to learn more than a very small number of universal commands. This number may become slightly larger if a standard set of universals is established and widely used throughout the speech industry.

Given that the number of universal commands should be small, it is best to associate the commands with functions that a caller can use to get out of troublefor example, asking for additional help or instruction, moving to a different part of the application, or requesting to speak to a live agent. Successful use of such universals should improve transaction completion rates, automation rates, and user satisfaction.

You should choose command words or phrases for universals that are intuitive and easy to remember (for example, "help"). The commands should have the same meaning, no matter when they are spoken. For example, saying "Help" should always mean that the caller wants more detailed instruction about what can be done at the

current point in the dialog. The instruction the caller hears in response will vary depending on the current context, but the universals should always be available.

Two standards bodies have been looking into universals: the Telephone Speech Standards Committee (TSSC 2000) and the European Telecommunications Standards Institute (ETSI 2002). Both committees have solicited input from the design and development community about its experience with universals. The two groups have conducted experiments to see what terminology occurs most naturally to users when they want to elicit the universal behaviors. Both committees have reached similar tentative conclusions.

The following list shows the set of universals that we recommend for all applications. The word or phrase in brackets is the actual command callers would use. Our choice of universals is based on the results of the two standards committees as well as our own experience with deployments. In the future, if an industrywide standard is accepted, we support conformance. The entire industry, and our callers, will benefit if a consistent set of universals is used for all applications.

- Clarification universals
    - [help]: Provide help or additional instruction about the current dialog state.
    - [repeat]: Repeat the most recently played prompt.
- Navigation universals

    - [main menu/start over]: Return to the beginning of the application (following any login process).
    - [go back]: Back up to the preceding step.

- Termination universals
    - [operator]: Transfer to an operator or customer service agent.
    - [good-bye]: Allow the caller to say "Good-bye" and respond appropriately so that the caller is comfortable hanging up.

The good-bye command is included because analysis of deployment data has shown that callers say it even when they are never told it is an available command. Usability studies have suggested that many callers are more comfortable saying good-bye, rather than just hanging up, especially when transactions are involved. It gives them confidence that hanging up does not prevent their transaction from being completed.

In general, callers must be taught about universal commands, or else they won't use them. One approach is to mention the help command in the initial prompt, which the caller hears when first entering the system. Description of the other universals can be included as the last part of help prompts, as well as part of any error recovery prompts. For example, a banking application might have the following initial prompt:

> *Welcome to Western Valley Bank. If you ever have a problem while you're using this service, just say,*

> *"Help." Now, would you like to pay a bill, check your balance, or transfer money?*

If the caller says "Help" while in the middle of an operation for obtaining an account balance, the system response might be the following:

> *Okay, here's some help. You requested an account balance, but I don't know which account. You can say, "Savings" or "Checking." Anytime you like, you can also say "Main menu" or "Operator."*

9.1.2 Consistency

You can also reduce callers' cognitive load by giving careful attention to consistency. When you design dialogs, the idea is to let the caller do similar things in similar ways. For example, there may be a number of lists that can be traversed in the course of an application (e.g., the stocks in the caller's portfolio, outstanding buy or sell orders, the caller's stock watch list). Using the same traversal strategy lets callers carry over what they learn from traversing one list to the traversal of all other lists.

Consistency can be applied at many levels. Dialog strategies should be consistent, as should terminology. Grammar coverage also should be consistent; for example, if you use money-amount grammars in two places, don't allow "fifty bucks" in one place but not the other.

In general, callers learn from every interaction. Each exchange between callers and the system refines callers' mental model of the system. That model includes their view of the system's capabilities, the commands available to perform various tasks, the richness and flexibility of

grammars, and so on. Being consistent helps make the caller's learning process more efficient and effective. Failing to be consistent contributes to learning inefficiency, leads to confusion, raises the out-of-grammar rate, and provides a much less satisfactory experience for callers.

Let's consider an example of list traversal. In the following dialog with a brokerage system, two types of lists are traversed: the list of companies in the caller's watch list and the list of outstanding orders for buys and sells. Notice that the caller uses the same set of terms for navigating each list.

(1)

SYSTEM: What would you like to do?

CALLER: Play my watch list.

SYSTEM: International Business Machines, eighty two, up one point three six. Amazon, thirty six, down . . .

CALLER: Next.

SYSTEM: Ford Motor Company, ten point . . .

CALLER: Next.

SYSTEM: Charles Schwab Corporation, eleven . . .

CALLER: Finished.

SYSTEM: What would you like to do next?

CALLER: Tell me my open orders.

SYSTEM: You have five open orders. Here's the first one: sell one hundred shares of Microsoft . . .

CALLER: Next.

SYSTEM: Buy two hundred shares of Cisco Systems at seventeen . . .

CALLER: Next.

SYSTEM: Buy five hundred shares of International Business Machines at eighty two . . .

CALLER: Finished.

SYSTEM: What would you like to do next?

9.1.3 Context Setting

Context setting is another important concept that can be applied to the design of voice user interfaces to minimize cognitive load (Weinschenk and Barker 2000). Psychological research has shown that people can understand and remember information more easily when it is presented with the appropriate context. For example, consider the following passage (Bransford and Johnson 1973):

> The procedure is actually quite simple. First you arrange things into different groups. Of course, one pile may be sufficient depending on how much there is to do. If you have to go somewhere else due to lack of facilities, that is the next step; otherwise you are pretty well set.

When asked to remember as many ideas in this passage as possible, subjects remembered only about three. However, when told beforehand that the passage is about laundry, they recalled twice as many ideas. Context helps people to associate new information with familiar concepts, easing cognitive load.

One way to set the context in a user interface is to use a metaphor. As discussed in Chapter 4, a metaphor is a familiar object or schema that is used to help facilitate understanding in another domain. You may recall the examples of the desktop metaphor and the shopping cart metaphor.

To investigate whether metaphors actually help users of voice interfaces, researchers at British Telecom conducted a study that compared three shopping applications equipped with a voice interface (Dutton, Foster, and Jack 1999). One system had no metaphor and simply described merchandise through a menu structure. Another had a department store metaphor in which the callers used a virtual elevator (with sound effects) to move from floor to floor. The system described the merchandise on display for that floor. A third system had a catalog magazine metaphor in which the system described merchandise presented in magazine pictures.

Callers rated the system with the department store metaphor significantly higher than the system with no metaphor. (The system with the magazine metaphor scored between these two.) In addition, callers were better able to navigate the system when a metaphor was present. These findings indicate that context setting through the use of a metaphor can improve user satisfaction and effectiveness.

9.2 Memory Load

Callers cannot process large amounts of new information at one time and will not remember new information if it is not immediately useful to them. There are a number of techniques for creating menus, wording prompts, and providing instruction that help minimize the load on a caller's memory.

9.2.1 Menu Size

In an influential article titled "The Magical Number Seven, Plus or Minus Two," Miller (1956) described a pattern of human short-term memory in which people can store seven, plus or minus two, items. Often, designers use this as a guideline for the number of items to put in lists, menus, and so on. However, listening to sentences over the phone while trying to extract and remember information from these sentences is much more taxing than the tasks Miller used in the lab. The caller's task is more akin to the listening task in which subjects are asked to listen to a series of sentences and remember the last word of each sentence (Daneman and Carpenter 1980). In experiments using this completely auditory approach in which sentence comprehension is also taking place, people can remember only about three items on average.

Other research on human memory has shown that people naturally cluster items in threes and that recall is best when information is divided into groups of three or four items (Broadbent 1975; Wickelgren 1964). Taken together, the research results suggest that the caller's memory load should be kept quite small. A reasonable guideline is to

limit menus to three or four items. Both Gardner-Bonneau (1992) and Schumacher, Hardzinski, and Schwarz (1995) make the same recommendation of four or fewer items in a menu.

9.2.2 Recency

When you write prompts, if callers are told specific words to use in response, make that the last thing they hear. For example, "To hear the list again, say, 'Repeat list'" is better than, "Say, 'Repeat list' to hear the list again." It taxes callers' memory less if the item they need to remember is the last thing they hear. This effect is often referred to as the **recency effect**. This ordering of *function* and then *action* in prompt wording has been adopted from standards of touchtone systems (Balentine 1999).

In addition to the recency effect, there are linguistic reasons to structure spoken sentences with the information of importance at the end. These are covered in Chapter 10.

9.2.3 Instruction

Applications that have many features and capabilities, especially those that will be used repeatedly by the same callers, often include instructional modes as part of the interactive application. Instruction sheets sent through the mail, or reminder cards with lists of commands, are not very effective. Most users do not read the instructions before attempting to use the system. Thus, applications need to be self-explanatory. It should be possible for an inexperienced user to get all needed help while using the service. We discuss two approaches here.

Tutorials

Some systems offer online tutorials, demos, or a combination of both. The option of hearing a tutorial is typically offered the first time a user calls the system. This approach is used mainly in subscription services or services that expect a lot of repeat use (for example, personal agents, banking, or brokerage account access). A **tutorial** consists of step-by-step instructions for using certain features of the system. A **demo** consists of a recorded interaction between a simulated caller and the system. The system voice speaks the prompts that are played during real system use.

Tutorials that present a demo of a user interacting with the system (CCIR-4 1999) and interactive tutorials (Kamm, Litman, and Walker 1998) have been shown to be effective instructional tools for new users. However, if too much information is provided or if the information is only described (rather than presented in an interactive mode), users have a hard time digesting it (Balogh, LeDuc, and Cohen 2001). There are two rules of thumb about tutorials:

1. Teach only a very small number of concepts.
2. Make it interactive. Have the caller actually perform the action.

Just-in-Time Instruction

When you need to describe a large amount of functionality, there are drawbacks to relying solely on a tutorial or demo. First, a caller will have difficulty following a lengthy description of diverse functionality. Second, if a feature is not exercised immediately, it is likely to be forgotten. In

general, callers will not have the patience to listen to a lengthy description, especially if it is not helping meet their immediate needs.

The notion of **just-in-time instruction** can address these two limitations of tutorials (Cohen 2000). The basic idea is to provide only the instruction relevant to the task immediately at hand, just before the caller needs to perform the task. The amount of new information introduced at that point is small, and it is exercised immediately.

For example, consider a personal agent application with a rich set of capabilities. Rather than hear a detailed tutorial the first time they use the system, callers are instructed about particular capabilities the first time they access them. For example, the first time callers ask for traffic information they might get the following message:

> *You can get up-to-the-minute traffic reports for the major roadways in any city simply by saying the city name. You can also save time by saying the name of the road or highway and the city name. For example, you can say, "Highway 101 in San Francisco."*

Just-in-time instruction can be offered the first time a caller exercises a particular capability. Additionally, you can offer instruction if a caller is having problems using the system, such as frequent rejects, timeouts, misrecognitions, and so on. If the caller is not making optimal use of system features (for example, not using shortcuts or not taking advantage of the richness of a grammar to efficiently input data), you can offer instruction.

The next two examples compare two approaches to instruction. In (2), first-time callers are offered a tutorial. The tutorial is lengthy and covers a large amount of

information. It is doubtful that a caller will retain very much of what is covered, especially in future calls to the system. In (3), the application uses just-in-time instruction. The same material is covered (this example shows only the instruction on quotes and watch lists). However, it is offered in small pieces, at the time when that information is relevant and useful to the caller's current activity.

(2)

SYSTEM: Welcome to Princeton Brokerage. I can help you get quotes, set up a watch list, trade equities or options, and access your account information. To get a quote, simply say the name of the company. To set up a watch list, say, "Set up watch list," and then you will be asked for each company you want to put in your list. After you set up a watch list, you can get quotes for all of the companies on your list simply by saying, "Watch list." To make additions or deletions, say, "Edit watch list." To make a trade, say, "Buy" or "Sell," followed by the name of the company you want to trade, the number of shares, and the price. For example, you can say, "Buy a hundred shares of Intel at

eighty-two," or "Sell fifty shares of Apple at the market." To trade options, say, "Options." To access your account information, say, "Account information." You will then be able to ask for your cash position, your holdings on any particular company, or a summary of all of your holdings.

(3)

SYSTEM: Welcome to Princeton Brokerage. I can help you get quotes, set up a watch list, trade equities and options, and access your account information. What would you like to do?

CALLER: I'd like a quote.

SYSTEM: To get a quote, just say the name of the stock, mutual fund, or index name.

CALLER: IBM.

SYSTEM: International Business Machines is selling at eighty two, up one point three six.

CALLER: Cisco.

SYSTEM: Cisco is selling at seventeen point two five, up point one seven.

CALLER: I wanna set up a watch list.

SYSTEM: Sure, watch list. But first, let me tell you a bit about how it works. I'm going to help you put some companies on your list. In the future, you can get quotes on your companies simply by saying, "Watch list." Now, what's the first company to add to your watch list?

CALLER: Cisco.

SYSTEM: What's the next company name? Or, you can say. "Finished."

CALLER: IBM.

SYSTEM: Tell me the next company name, or say, "Finished."

CALLER: Intel.

SYSTEM: Next one?

CALLER: America Online.

SYSTEM: Next?

CALLER: <pause>

SYSTEM: Sorry, what's the next company name? <pause> Or just say, "Finished."

CALLER: I'm finished.

SYSTEM: Okay! Now whenever you want to get quotes for the companies on your watch list, just say, "Watch list." And to add or delete a company, say, "Edit my watch list." Now, what would you like to do next?

9.3 Attention

Attention is the process of selecting what to focus on, given the multitude of possibilities available (Preece, Rogers, and Sharp 2002). The way you present information to callers can have a significant effect on how easily they can attend to the information of greatest interest.

Let's look at an example of reading back flight information in a travel planning application. Imagine that a caller has requested a flight from New York to Boston "this afternoon." Assuming that the system has found four flights in its database that match the caller's criteria, it must inform the caller of the possibilities and request a selection. Imagine the following presentation of the list of four possible flights:

(4)

SYSTEM: United Airlines flight 47 leaves New York Kennedy Airport at 1 p.m. from gate 36 and arrives at Boston Logan at 1:45 p.m. at gate 22. American Airlines flight 243 leaves New York Kennedy Airport at 2:15 p.m. from gate 12 and arrives at Boston Logan at 3 p.m. at gate 47. American Airlines flight 260 leaves New York Kennedy Airport at 3:45 p.m. from gate 15 and arrives at Boston Logan at 4:30 p.m. at gate 42. United Airlines flight 52 leaves New York Kennedy Airport at 5 p.m. from gate 38 and arrives at Boston Logan at 5:45 p.m. at gate 31. Which would you like?

Whether or not one of these flights meets the caller's needs, the presentation of the information is so cluttered as to derail the caller's ability to attend to the information that is most important for the decision. Now imagine the following alternative:

(5)

SYSTEM: There are flights at 1 p.m., 2:15, 3:45, and 5. Which would you like?

CALLER: How about the 2:15 flight.

SYSTEM: American Flight 243 leaves New York Kennedy Airport at 2:15 p.m., arriving at Boston Logan at 3. Would you like to book this flight?

In this case, only the information of greatest concern to the caller is provided, easing the decision. If your application will present possibly complex information, you should note that fact during requirements definition. You should make sure you understand the prospective caller's goals, priorities, and decision criteria. In this way, you can optimize the presentation of information and not unduly challenge the caller's ability to attend to the information of greatest concern.

In some cases, divided attention is inevitable. For example, when users are driving, situations may arise that temporarily demand their complete attention. Systems designed for use while driving must accommodate this need by providing the caller with control over the pacing of the interaction. Users might exert such control by, for

example, using a pause/resume feature. Or the application might clarify the conversational context after a timeout so that the caller can seamlessly continue with the dialog. (Note: The issue of VUI design for driver safety is an important area that needs careful research. The suggestions in this paragraph are meant to be illustrative only. They are not based on research with subjects in real or simulated driving situations.)

In other cases, you may desire to communicate information without derailing the caller's current activity. Imagine a personal agent application that manages callers' phone calls and voice mail and can read them their e-mail over the phone. If a new voice mail message arrives while users are listening to e-mail, the application can use a recognizable earcon to notify them without interrupting the flow of e-mail reading. The message "New voice mail has arrived" is communicated via the earcon without disrupting the caller's attention.

In general, the first step in facilitating attention is to understand the caller's goals and priorities. Then you can design strategies to make only the pertinent information available. At the same time, the system can accommodate the caller's needs to attend to information and events outside the purview of the application.

9.4 Conclusion

We have reviewed guidelines to help you deal with cognitive challenges, including conceptual complexity, memory load, and attention. As with all design guidelines, they must be applied with careful consideration of context.

The next two chapters discuss the accommodation of conversational expectations. Chapter 10 discusses issues of prompt wording (*what* the system says), and Chapter 11 discusses issues of prompt prosody (*how* the system says it).

Chapter 10. Designing Prompts

A VUI dialog is a type of conversation. Perhaps this is even more the case for telephony-based VUIs; after all, the telephone itself is an icon of conversation in modern culture. Even though one of the participants in a VUI dialog is a machine, both the user and the designer who engineered the dialog share a few basic expectations that also hold for human-to-human conversations. Both parties expect the use of a common language; each expects the other to be cooperative; and each expects that the other will possess basic cognitive faculties such as intelligence, short-term memory, and an attention span (which are artificial in the case of the machine but are nonetheless expected). Another level of expectation is implicit in the way users approach spoken language systems, and it has to do with the way prompts are worded.

Humans approach all linguistic tasks with unconscious expectations and assumptions not only about content but also about form. Because VUI prompts are communicated through spoken language, their form should satisfy the spoken language norms that users naturally expect. And the spoken language system that VUI users are most familiar with is everyday conversation. We recommend that prompts be modeled after conversational language, in the interest of promoting the design of human-centered, user-friendly, optimally comprehensible VUIs. The alternative is a way of communicating that is less familiar and less natural and therefore less comprehensible.

This chapter demonstrates how specific elements of human-to-human conversation can be applied to prompting. We review some basic linguistic concepts for creating a spoken language experience that users will find

familiar and engaging. You will also see a number of cases where the intelligibility of prompts is greatly enhanced via the use of conversational language.

It is sometimes thought that "conversational" necessarily implies informality and chattiness. For our purposes, **conversational** refers more technically to the linguistic form of everyday spoken interactions. Indeed, there are many types of conversations in real life, from chatty and informal to serious and formal. The high-level recommendations made throughout this chapter apply to the design of *all* VUIs, including those that require an air of formality.

It is also sometimes thought that the goal of conversational interfaces is to lead unwitting VUI users into thinking that they are interacting with a human being, but this is not what we advocate. Our goal is to leverage the linguistic forms that VUI users are best acquainted with. In fact, linguistic research tells us that conversational language incorporates many design features that are specially adapted and ideally suited to the task of communicating via sound. Thus, many of the principles in this chapter are about improving listening comprehension and enhancing the user's level of comfort by exploiting the linguistically familiar.

In designing speech applications, we are also designing a language experience for the user. Although **speech** technically refers to such matters as acoustics, physiology, sound production, and perception, **language** involves the intimate connection of structure, meaning, and use, all of which are context-sensitive. Linguistic form is shaped by forces that range from perceptual and cognitive, on the one hand, to social and cultural on the other. The form of conversational language is not haphazard but rather is systematic and principled. Dialog designers should refer to

this naturally occurring system as a resource for prompting, at least to the extent that the current technology allows. As much as possible, naturally occurring spoken language should serve as a guide for writing successful speech applications.

The ideas presented in this chapter should assist you in creating socially and linguistically appropriate characters for your application (see Chapter 6 for more on persona design). It is a common belief that an application's persona depends on hiring the right voice talent. In fact, not even the most experienced, skilled actor with the most pleasantly resonant voice can make inherently nonconversational prompts sound naturalfor example, "Please say the duration" instead of a more natural wording such as, "How long do you expect it to last?" As in the movies, good actors depend on good scripts. To create the sense of a familiar, engaging persona to represent your company's brand or image, conversational norms should guide the way prompts are written. Reaping the benefits of a well-designed persona depends on your ability to write messages that suggest natural spoken language.

In usability studies of a voice-browser application, callers expressed a sense of loyalty to the professional, matter-of-fact, but personal and familiar character representing the systeman administrative assistant in her early 40s. Users felt comfortable entrusting to her their most sensitive personal information, such as credit card details. But a well-designed, likeable persona is essentially an illusion, a mosaic built up of consistent, conversational messages that collectively suggest a coherent, sociolinguistically familiar presence. Usability informants reacted favorably not only to the voice itself, but also to the way prompts were worded (discussed in this chapter) and to the way they were spoken (discussed in Chapter 11).

The remainder of this chapter covers the following topics:

- Conversation as discourse
- Cohesion
- Information structure
- Spoken and written English
- Register and consistency
- Jargon
- The Cooperative Principle

10.1 Conversation as Discourse

To apply elements of naturally occurring language to VUI prompting, you need to understand conversation as a type of **discourse**: naturally occurring, human-to-human language. Discourse is language above and beyond the use of individual words and sentences studied in isolation. Discourse analysts make important assumptions about the nature of human language:

- Language always occurs in a context, and its structure is systematically context-sensitive.
- Language is essentially a communicative phenomenon, and its structure incorporates certain design features specially adapted to this end.

In spoken language, there are discourse types other than everyday conversation, each having its own set of requirements concerning form and content. There are, for example, fairy tales, sermons, news reports, radio advertisements, courtroom trials, lectures, eulogies, airplane safety demonstrations, and auctions.[1]

[1] Some examples of discourse genres particular to writing are short stories, love letters, warning labels, road signs, e-mail messages, advice columns, recipes, cover letters, offer letters, appliance warranties, horoscopes, gossip columns, shopping lists, obituaries, and anything from the Internal Revenue Service.

There are a few basic characteristics of naturally occurring discourse, including everyday conversation, that dialog designers should consider. These are important considerations even if the VUI you are designing is not an English language application. Discourse has the following characteristics:

- **Discourse is principled.** Language is systematic on many levels. Every spoken language has its own rules for putting sounds together to make words and for putting words together to make sentences. For example, "blick," "ving," and "brizzle" are not words in English, but they *could* be. "Try new-and-improved lemon-fresh Brizzle … with grease-busting enzymes!" In contrast, "sbab," "vlum," or "knerpf" must all be rejected as possible words because these forms violate the rules of sound sequencing that are particular to English. Similarly, "That dog seems to be dreaming" is a perfectly fine sentence, but "That dog seems dreaming" is not. Conversations are like words and sentences in that they, too, conform to a highly structured system in which some creations are not allowed. In the world of VUIs, engineered conversations are at times as **ungrammatical**, in the sense of nonoccurring or unacceptable, as are forms such as "Sbab" and "That dog seems dreaming." In this chapter, you will see many examples of nonconversational prompts, along with examples of how they can be improved.

- **Discourse is universal.** Just as all languages have complex syntactic and phonological systems, so do all languages have sophisticated discourse patterns in their spoken form. Even if the VUI you are designing is not an English language application, the basic concerns of this chapter are applicable to all human languages, even though the details may vary from language to language.

- **Discourse is conventionalized.** Another reason for taking care with the wording of prompts is that discourse is **conventionalized**, or shared throughout a linguistic community. Regardless of educational background or socioeconomic standing, all native

speakers of English agree, for example, on the use of the discourse marker "by the way" to mark a conversational contribution as tangential or unrelated but worthy of mention. Similarly, native speakers of English generally place elements of focus at or near the end of a sentence, which is where they expect to retrieve them, as well. It is not possible for VUI designers to invent a new feature of discourse, but they can capitalize on those conventions that already exist.

- **Discourse is unconscious.** The mechanics of discourse generally escape our awareness. Consider, for example, the use of "this" and "that" as discourse **pointer words**. As described later in this chapter, native speakers of English use these words differently in speaking than they do in writing, although very few are aware that a difference exists. We might notice that something doesn't sound right when a nonnative speaker uses "this" instead of "that," but most people are unable to articulate exactly why. As it happens, the way that pointer words are used in many VUIs corresponds to the written rather than the spoken mode. Because VUI prompts are spoken, however, pointer words in prompts should reflect the conventions of speaking rather than writing. Writing spoken dialog is different from writing language that is to be read.

The following sections center on specific features of spoken language and explain how you can leverage them to optimize comprehension, comfort, and familiarity in your VUI.

10.2 Cohesion

Cohesion is the glue of discourse. Consider the following written piece of discourse:

(1)

George couldn't wait to get to France. *However, he* didn't stay *there* long.

"He" refers to "George," "there" refers to "France," and "however" establishes an adversative or contrastive relationship between the proposition that George was eager to go to France and that George's visit to France was short-lived. The example illustrates three types of cohesion devices. **Cohesion** refers to explicit linguistic devices, such as "however," "he," and "there," that help bind language into a coherent whole. **Coherence**, in turn, refers to the functional unity of a piece of discourse. Cohesion can be found not only in a written monolog, as in the example, but also in all dialogs.

Cohesion devices facilitate and reinforce comprehension of the whole, but they themselves are not responsible for creating meaning. Rather, they are natural cues that speakers and hearers use to signal and retrieve meanings

that underlie utterances in context. The cohesion devices that are particularly relevant to prompt design are pronouns, discourse markers, and special pointer words such as "this" and "that."

10.2.1 Pronouns and Time Adverbs

A special feature of human-to-human language is that information in one unit of talk often presupposes information presented in a previous one, as exemplified in (2) through (4) (Schiffrin 1998).

(2)

I saw the cat the other day. *It* was still wandering around without a home.

(3)

I saw a robin the other day. It was the first *one* I saw this spring.

(4)

We moved here in 1982. We didn't even have jobs *then*.

In conversation, pronouns such as "it" and "one," as well as time adverbs such as "then," let the listener know that the referent at hand is the *same* as a referent uttered previously.[2] In (2), (3), and (4), the reader or listener can easily make sense of these sentences by recovering the missing information from an earlier point in the discourse.

[2] In writing, however, pronouns sometimes refer to something that comes up later in the text. For example, in the sentence "As soon as *she* got home, *Pat* called her lawyer," "she" and "Pat" are coreferential. In conversation, we would more likely say, "As soon as *Pat* got home, *she* called her lawyer."

Compare the stilted effect created by the repetition of the term "bookmark(s)" in the prompts listed in (5) with those in (6), rewritten with the pronoun "one."

(5)

SYSTEM: You have five *bookmarks*. Here's the first *bookmark.* . . . Next *bookmark*. . . .That was the last *bookmark*.

CALLER: Delete a *bookmark*.

SYSTEM: Which *bookmark* would you like to delete?

. . .

SYSTEM: Do you want to delete another *bookmark*?

(6)

SYSTEM: You have five *bookmarks*. Here's the first *one*. . . . Next *one.* . . . That was the last *one*.

CALLER: Delete a bookmark.

SYSTEM: Which *one* would you like to delete?

. . .

SYSTEM: Do you want to delete another *one*?

With the simple replacement of the noun "bookmark" with the pronoun "one," the prompts in (6) flow more naturally than those in (5). If VUI designers write prompts in context and listen to the dialog that emerges, they will be more likely to appreciate the ubiquity of pronouns in spoken language and to reap their stylistic benefits.

10.2.2 Discourse Markers

Discourse markers are defined as "sequentially dependent elements which bracket units of talk" (Schiffrin 1987). They connect utterances (or portions thereof) by succinctly relating what has just been said to what is about to be said through time, result, contrast, and other such notions. The following is a list of discourse markers,

organized according to class of use (adapted from Quirk and Greenbaum 1973).

- **Enumerative:** first, second, third; for one thing, and for another thing, to begin with, for starters; in the first place, in the second place; one, two, three . . .; a, b, c . . .; next, then; finally, last, lastly; to conclude
- **Reinforcing:** also, furthermore, moreover, then, in addition, above all, what's more
- **Equative:** equally, likewise, similarly, in the same way
- **Transitional:** by the way, incidentally, now
- **Summative:** then, (all) in all, in conclusion, in sum, to sum up
- **Apposition:** namely, in other words, for example, for instance, that is, that is to say
- **Result:** consequently, hence, so, therefore, thus, as a result, somehow, for some reason or other
- **Inferential:** else, otherwise, then, in other words, in that case
- **Reformulatory:** better, rather, in other words
- **Replacive:** alternatively, rather, on the other hand
- **Antithetic** (or contrastive): instead (blend of antithetic with replacive), then, on the contrary, in contrast, by

comparison, (on the one hand . . .) on the other hand

- **Concessive:** anyhow, anyway, besides (blend of reinforcing with concessive), else, however, nevertheless, still, though, yet, in any case, at any rate, in spite of that, after all, on the other hand, all the same, admittedly

- **Temporal transition:** meantime, meanwhile, in the meantime

- **Attitudinal, commenting on truth:** actually, in actuality, in (actual) fact, strictly speaking, nominally, officially, technically, theoretically

Occasionally customers resist the use of discourse markers in their VUIs on the grounds that they are perceived as "informal" or "slang." In and of themselves, however, discourse markers as a linguistic category are neither formal nor informal. The following markers, for example, are actually more formal than what we are used to hearing in everyday conversation: *in addition, thus, therefore, hence, nevertheless, on the contrary, conversely, by comparison, in contrast, equally, all other things being equal, rather,* and *in conclusion*. Markers that epitomize informal conversation are, for example, *for starters, anyhow*, and *so.*

There are significant benefits of **discourse sensitivity**, including the appropriate use of discourse markers, to an engineered dialog. Compare examples (7) and (8) (from Giangola 2000). Both sets of prompts aim to collect the same pieces of information to schedule an appointment in a personal information manager application. The prompts in (7) conspicuously lack discourse markers, whereas those in (8) were rewritten to project a natural, conversational style.

Here we find both kinds of cohesion devices that have been discussed: pronouns and discourse markers (both in italics).

(7)

Please say the date.

Please say the start time.

Please say the duration.

Please say the subject.

(8)

First, tell me the date.

Next, I'll need the time *it* starts.

Thanks. <pause> *Now,* how long is *it* supposed to last?

Last of all, I just need a brief description . . .

Discourse markers add value to prompting in several ways. First, recall the central assumptions that discourse analysts

make about human language. Language is a context-sensitive system that incorporates design principles specially adapted for communication. The redundant nature of discourse markers reinforces the functional relationship between two units of discourse. The discourse marker in the utterance, "Joe is a couch potato; his brother, *on the other hand* . . ." induces us to expect sharp contrast, giving us a pragmatic preview of what comes next. You, the reader, might have thought to complete the sentence with "is a triathlete," but you would not complete it with "hates to exercise." In (8), discourse markers such as *first, next,* and *last of all* orient listeners to their position in a sequence of questions.

Second, discourse markers suggest a humanlike awareness of how the dialog is progressing. Not only does the application recognize specific responses such as "Yes," "No," "Three p.m.," and "Operator," it seems to recognize how the user's contributions to the conversation relate to those of the system on more abstract cognitive and social levels. In these roles, discourse markers can be said to serve **conversation management** functions. This point is most robustly demonstrated later, in the discussion of markers such as *oh, by the way,* and *actually*, which are deceptively simple but functionally loaded.

As you can see from these examples, discourse markers serve a critical role in imparting naturalness to engineered dialogs, in addition to their essential role in projecting a familiar and favorable persona. In both robotic, context-insensitive (7) and natural, context-sensitive (8) dialogs, the information gap that initially exists between the application and the user is resolved only one piece at a time, and this has the potential to frustrate callers. Usability feedback reveals that the wording in (7) actually heightens the listener's awareness that information is being collected piecemeal. One respondent said that hearing one

"Please say" after another was annoying, presumably owing to its unrelenting repetition. In general, listeners associate the dialog in (7) with such negative descriptors as "frustrating," "impatient," and "bored," whereas the dialog in (8) is associated with the positive descriptors "attentive" and "helpful."

Let's now turn to the use of specific discourse markers and see how they can enhance the natural flow of prompts in speech interfaces. These markers are *now, by the way, oh, otherwise,* and *actually*. We also look at conversational uses of *okay* and *sorry*. These are not typical discourse markersbecause they can stand alonebut they function as discourse markers nonetheless.

Now

Now functions as a kind of paragraph-changer, a verbal indentation, as you saw earlier in "Thanks. *Now,* how long is it supposed to last?" (8). This discourse marker often introduces a new stage in a thought sequence, marking a progression of topics. For example, when a single prompt has more than one topic to cover, the discourse marker *now* subtly and gracefully ushers the caller's attention to the next topic in the sequence. In example (9), the caller has just completed a number of steps to enroll the caller's voiceprint.

(9)

SYSTEM: That's all I need. We are done getting your voiceprint. *Now,* since you're new, would you like some help getting started?

Because *now* signals a change in topic in the discourse and does not refer to actual time, we can say things such as "*Now*, in the *old* days . . ."

By the Way

Just-in-time instruction, the instructional technique introduced in Chapter 9, lends itself to introduction with the discourse marker *by the way*, as exemplified in (10).

(10)

CALLER: Call 555 465 2468.

SYSTEM: Calling 555 465 2468. *By the way,* when you're done with this call, you don't have to hang up. <pause> Just say, "Come back."

As in human-to-human conversation, *by the way* introduces a tangential thought, often of parenthetical importance.

Oh

Such a small word, and yet volumes could be written about the discourse functions of *oh*. Although most dictionaries sweep *oh* under the carpet as a mere "hesitator" or "filler" expressing surprise, Schiffrin (1987) proposes that the central, prototypical function of the marker *oh* is to signal some sort of cognitive mismatch, communicating something along the lines of, "What I know or what I think does not match what you know or think."

In (11), the caller attempts to navigate to a site that is not considered secure. The interface informs that this will not be possible and tells why.

(11)

CALLER: Go to <name of a Web site>.

SYSTEM: *Oh,* I can't go to that site. It's not considered "secure."

In (12), the same interface informs callers that their personal profile cannot be bookmarked (referenced as a preferred voice site).

(12)

CALLER: Make that a bookmark.

SYSTEM: *Oh,* that can't be bookmarked . . .

This use of *oh* is related, by extension, to a **self-editing** use, as exemplified in (13) and (14), where the vertical bar represents the start of a new audio file.

(13)

CALLER: Go to sports.

SYSTEM: Sure, sports! | Oh, that's not available at the moment, but would you like to try something else instead?

(14)

CALLER: Get my messages.

SYSTEM: Okay, messages. | Oh, looks like you don't have any messages right now.

As a self-editing marker of cognitive mismatch, *oh* is invaluable in cases like these. The system response in (13) started with the implication that sports was available, and in (14) that messages were. The second message in each case contradicts the initial assumption. But introduced by *oh*, the otherwise incompatible piece of information is ushered in smoothly and gracefully, cast as a sudden

change in the system's knowledge. In other words, the character played by the voice actor has only just now found out that sports or messages aren't available, so it can't be held at fault. The character was only trying to be attentive and cooperative initially but, for reasons beyond his or her control, was unable to deliver.

Be careful not to pepper the dialog with *oh* for reasons that are hard to justify. For example, the caller opts for "weather" at the main menu and the interface responds with the acknowledgment, "Oh, weather." Depending on how you say it, this phrase is open to a number of interpretations. One reading could be that the interface's persona has temporarily forgotten that weather existed as an option. Another reading might be that the persona is surprised that the caller has asked for weather, as opposed to some other option. Yet another possibility is that the persona is secretly wishing that the caller had not made this selection. (Along these lines, think about how the voice actor is supposed to say "Oh, weather." How would you coach the delivery?) These interpretations all resonate with Schiffrin's analysis of *oh* as a marker of cognitive mismatch, but this use is not appropriate here. In contrast, an appropriate context for "Oh, weather" would be to recover from a misrecognition, as in "Oh, *weather*," as opposed to what had been misunderstood as a request for traffic or horoscopes. In this context, the discourse marker appropriately signals cognitive mismatch: "What I heard before isn't what I'm now hearing." In any case, a more appropriate wording for the original context would be, "Sure, weather" or "Okay, weather," which communicate more neutral acknowledgments.

Because different discourse markers have different context-sensitive uses, it is critical that you choose carefully.

Actually

As with *oh*, there are a number of other discourse markers that suggest that an utterance may not be completely true or exactly right in realityfor example, *officially, technically,* and *theoretically*. Other discourse markers, such as *actually* (15)*, really, in (actual) fact,* and *in reality*, indicate what is considered to be the "real" truth.

(15)

CALLER: Get my personalized weather report.

SYSTEM: Actually, you had a list of cities for me to check on, but it's been deleted. To make a new list, go to the registration Web site. For now, just tell me what city and state you want the weather for. <pause> Or a city and a country.

One application we have worked on makes use of *actually* in certain error prompts, as in (16). The caller tries out the navigational command "Go forward," although he or she has not yet visited any voice sites.

(16)

CALLER: Go forward.

SYSTEM: Actually, I can't move ahead since we haven't gone anywhere yet. To hear a list of places you can go, just ask what's available.

Actually introduces the "real truth" and prepares the listener for some detail that he or she has probably not anticipated.

Otherwise

This is probably the most abused of discourse markers in speech interfaces. It frequently appears in error-recovery prompts, as exemplified in (17).

(17)

SYSTEM: If you'd like to leave a message, say, "Yes"; *otherwise*, say, "No."

In real-life conversations, the information that follows *otherwise* is not strictly predictable from what came before, in spite of this marker's literal "if not" definition. The natural, informative use of *otherwise* is illustrated in (18) and (19).

(18)

If you're running late, call me. Otherwise, I'll come by at eight-thirty.

(19)

We'd better leave now. Otherwise, we won't get good seats.

It would be conversationally abnormal for someone to say, "If you're running late, call me. Otherwise, don't call me" or "We'd better leave now. Otherwise, we'll leave later." These "otherwise" alternatives are so predictable from their antecedents that the phrase as a whole is devoid of informational meaning. There is a conversational principle that can be roughly paraphrased as "Be informative." (This principle is a corollary to the Cooperative Principle discussed later in this chapter.) The use of *otherwise* in prompts such as (17) does not move the hearer into a sufficiently informative space, as would be expected in authentic discourse.

Prompts such as (17) may sound unnatural to users for another reason. "Yes" and "no" were natural responses to the state-initial prompt, which would have been something like, "Do you want to leave a message?" some two turns earlier in the conversation. What (17) is really saying is, "If you want to leave a message, answer as directly as possible my *last* question, because we're in the same recognition state and so the same recognition grammar is active." So the wording in (17) is designer-centered because it assumes familiarity with the concept of recognition states and active grammars.

In user-centered prompting, in contrast, *otherwise* would direct the caller into a more informative space, as in (20). Or you could reword the whole prompt and avoid its use altogether, as in (21).

(20)

If you want to leave a message, start talking after the beep. Otherwise, <pause> feel free to hang up.

(21)

Sorry, did you want to leave a message?

Yes or no, did you want to leave a message?

Did you want to leave a message? *[parenthetically]* (Just say, "Yes" or "No.")

The next two items are not discourse markers in the technical sense, but they serve a similar transitional function.

Okay

In one application we have designed, whenever the caller answers a question with no or a synonym, the ensuing prompt begins with *okay*, or some other acknowledgment, as exemplified in (22).

(22)

SYSTEM: Did you want to review some more of your personal profile?

CALLER: No.

SYSTEM: *Okay,* what's next?

Okay acknowledges the caller's refusal and smoothly paves the way for the next dialog state. Notice that if we remove *okay* (23), not only is the result a stilted-sounding dialog, but also the dialog gives the vague impression that the system might not have heard what the caller just said.

(23)

SYSTEM: Did you want to review some more of your personal profile?

CALLER: No.

SYSTEM: What's next?

In (22), *okay* could be replaced with *all right, no problem,* or *not a problem*, depending on the persona.

Sorry

A common criticism of the use of *sorry* in prompting is, "But, come on, now. How can a computer be sorry?" But *sorry* is useful, not so much for apology but rather as a signal that the forthcoming contribution will somehow fail to meet the listener's expectations. It can be employed as a mild bracing advisory. When your local video store clerk informs you that the movie you seek to rent week after week is yet again unavailable, there may be no hint of regret in the tone of "Sorry, it's out" or "Sorry, we don't have it."

In the recording studio, skilled voice actors can read "sorry" more as a transitional device than as a genuine expression of full-fledged, heartfelt regret. For example, in one application we have worked on, second-time recognition

errors trigger the generic prefix-type prompt in (24) as well as a number of randomized paraphrases. The voice actor's delivery of "sorry" is professional and neutral-sounding, a transitional device that mirrors the conventionalized use of "sorry" in day-to-day public life.

(24)

CALLER: My PIN is three six, um, no, it's *two* six four seven.

SYSTEM: *Sorry,* I didn't catch that.

Avoid using discourse markers just to make the interface sound more like a real person. In our view, *the primary purpose of using discourse markers in a VUI is to facilitate comprehension, thereby reducing cognitive load*, and not to create the deception that the system is a human attendant. Discourse markers, after all, are a design feature of natural discourse that allows speakers to reinforce meaning relationships between utterances, in turn allowing listeners to decode them more easily. Prompt writers should take advantage of this "design feature" of natural discourse, but with care.

10.3 Information Structure

For the purposes of this book, let **information structure** simply refer to the placement of contextually determined "old" versus "new" information, whether in recorded messages or in natural conversation. For example, consider the following interaction between one of the authors (JG) and a local directory assistance service.

(25)

LIVE 411 OPERATOR: Hi, this is Joan. What city?

CALLER: Mountain View.

LIVE 411 OPERATOR: All right?

CALLER: CostPlus.

RECORDING: The number you requested, five, five, five, nine, six, one, six, zero, six, six, can be automatically dialed by pressing one now. An additional charge will apply.

Except for the prosody of the phone number (see Chapter 11), there is nothing strange about the recorded message, as far as individual words and sentences go. Yet the way the system gives the requested number is nonetheless uncomfortable. Although we have not conducted a formal study, anecdotally people say, "I have to hang up in the middle of the sentence if I want to remember the number." The problem has to do with the way this recording figures in contextspecifically, the way that the "old" versus "new"information has been laid out for the listener. From the point of view of most callers, the new piece of information, or **focus**, should be the requested phone number (555 961 6066), but this information is buried in the part of the sentence that is conventionally reserved for old, or nonfocal, information. Such placement actually encourages us to forget what the number is, even though that was the purpose of the call to begin with.

VUI designers and prompt writers should be aware that there is a neutral (default) position of focus in English, as determined by what is called the **End-Focus Principle** (Quirk and Greenbaum 1973). That is, native speakers of English naturally place new or focal information at or near the end of the sentence, and this is where native hearers of English naturally expect to retrieve it. An example of a conversational message that not only complies with the End-Focus Principle but also facilitates short-term memory would be, "Here's the number: 555 961 6066. For automatic dialing, press one . . ."

The way people naturally structure information is highly context-sensitive. In the case of the directory assistance recording, it is *context* that makes the positioning of the phone number inappropriate. It is easy to imagine some other context where a phone number would occupy this position appropriately, as in frequently heard messages of the type, "The number you have dialed, 444-4444, is not in service. Please check the listing and dial again." In this context, the already dialed phone number is *old* information, and so its nonfocal placement in the middle of the sentence is justified.

To be technically accurate, end focus is given to the last **open-class** item or proper noun in a clause (Quirk and Greenbaum 1973).[3] Open-class items are nouns, adjectives, verbs, and adverbs. These categories are called "open" because they are open to new members (witness the creation of such verbs as "downsize," "e-mail," and "dis"). In contrast, **closed-class** items are articles, demonstratives, prepositions, and conjunctions (these are categories that reject new members; no one can invent alternatives to words such as "the," "that," "of," or "and"). In the sentence "Sean Connery was born in Scotland," the last open-class item is the noun "Scotland." By default, it is the focus, the new piece of information in this sentence. In contrast, "Sean Connery" is the **topic** (subject) of the sentence, or the old piece of information on which the speaker makes some comment. Old information is generally placed in the subject, whereas new information is generally housed in the predicate.

[3] Focus can occur at earlier points in the sentence, but this requires the use of special, "contrastive" stress and intonation, as in "Sean *Conn*ery was born in Scotland," which would answer the question of *who* was born in Scotland.

Because prompts are sometimes written out of context, it is easy to find examples that violate principles of information structure, which is context-sensitive. For example, in a

certain application that recognizes dates, there are special messages associated with months that have only thirty days. These messages are played only when the recognizer returns, for example, "June thirty-first," or "February twenty-ninth" in a nonleap year, with high confidence. To comply with the End-Focus Principle, we would expect such messages to be worded as in (26), but in fact, they are worded as in (27).

(26)

CALLER: Make it for June thirty-first.

SYSTEM: Actually, June has only *thirty* days.

(27)

CALLER: Make it for June thirty-first.

SYSTEM: There are only thirty days in *June*.

The wording in (26) appropriately casts "June" as the subject of the sentence. In this position, it assumes the role of topic, on which some new comment will be made. The new information is the "only thirty days" part, which is appropriately cast in the predicate, where it receives end focus.

It seems plausible that the End-Focus Principle has something to do with the short-term memory phenomenon known as the recency effect, as discussed in Chapter 9. Elements that come at the end are more prominent and easier to remember precisely because they are the last thing one hears. To exploit these linguistic and psychological principles, it makes sense to structure directions so that the novel element, such as which key callers should press or which voice command they should say, follows the topic, which is the objective or task. Preferably, the touchtone or voice command should be placed at the end of the prompt.

Careless handling of focal information can lead to ambiguity and wrong inferences. For example, a certain application allows callers to choose from a list of song titles and download the musical selection to a cell phone. To select a song, the caller is supposed to say, "That one!" upon hearing the desired title. In the sample interaction in (28), the system prompts the caller to say whether or not it got the selection right, simultaneously informing the caller about the cost:

(28)

[System plays list of Britney Spears selections] "Oops, I Did It Again" <pause>

CALLER: That one!

SYSTEM: Would you like to buy the song "Oops, I Did It Again" for $4.99?

What is being sought here? Is it confirmation of the song selection ("Oops, I Did It Again") or of the price ($4.99)? Context and world knowledge likely indicate the former, but the information structure, with the price in the end-focus position, suggests the latter, as if the price were open for negotiation. In other words, the prompt is ambiguous, permitting two valid interpretations. As a legitimate response, a haggle-prone caller might legitimately reply, "No, but I'll give you a buck fifty for it." The prompt can be recast unambiguously as something like this: "Good choice <pause>'Oops, I Did It Again' is only $4.99. Do you want to go ahead and buy it?" Here, the price has been removed from the scope of interrogation, and the prompt has only one possible interpretation.

In light of the End-Focus Principle, we should have a renewed appreciation for the communicative function of the passive voice in conversation as well as in prompts.

Consider the active and passive sentences in Figures 10-1 and 10-2, respectively. In both cases, "Leif Erikson" is the **agent** (that is, the do-er), and "North America" is the **patient** (the affected element) of the verb "discover."

Figure 10-1. In the active voice, the agent is the old or topical, and the patient is the new or focal.

Active Voice (answers "What did Leif Erikson discover?")
Leif Erikson discovered North America.
Agent **Patient**
Topic/old info **Focus/new info**

Figure 10-2. In the passive voice, the patient is the old or topical, and the agent is the new or focal.

Passive Voice (answers "Who discovered North America?")
North America was discovered by Leif Erikson.
Patient **Agent**
Topic/old info **Focus/new info**

These examples illustrate a fundamental difference in the use of the active and the passive voice. In the active voice, the agent is the topic, or old information, and the patient is the new information, or focus. In the passive voice, these roles are reversed: The patient is the topic (or old information), and the agent is the focus (or new information). So the passive voice is useful because it allows the speaker to reverse the usual order of agent and

patient, thereby reversing their default informational status.

The choice of the passive voice is more than appropriateit seems to be requiredin the recorded announcement in (29).

(29)

This program has been made possible by a grant from the Ford Foundation.

If we attempt to rewrite the sentence to "avoid the passive voice," as suggested in some texts on writing style, the result is strangely off-balance: "A grant from the Ford Foundation has made this program possible." In the context in which we would hear (29), the piece of information that requires focus is "the Ford Foundation." What is old, topical information is "this program," the one we've been watching for the past hour. The activized version of (29) strikes the ear as off-balance precisely because it flouts the End-Focus Principle.

The passive voice is used appropriately in the following two examples from a telecommunications VUI. The company that owns the application wanted to make it very clear to callers who exactly would be handling their request. Some requests would be handled by the new, automated speech system, whereas others would be routed to customer

service representatives, as before. The exchanges in (30) and (31) illustrate these two dialog possibilities.

(30)

CALLER: Copy of my bill.

SYSTEM: Your request for a copy of your bill will be handled by our Automated Express Service.

(31)

CALLER: Clarification, please.

SYSTEM: Your request for clarification of an item on your bill will be handled by the next available representative.

In the system's responses in (30) and (31), the old information"your request for a copy of your bill" and "your request for clarification," respectivelyis maintained as old information in the subject slot, and the new pieces of information"our automated express service" and "next available representative"are accordingly given end focus. If the prompt writer had put these sentences in the active voice to satisfy his or her grammar-checking software, the result would have been stilted, context-insensitive messages such as, "Our automated express service will handle your request for a copy of your bill."

This discussion of information structure reveals the significance and the implications of speech's linear organization. The prompt writer must consider the role of context in judiciously sequencing elements for a familiar, comprehensible listening experience.

10.4 Spoken Versus Written English

In both touchtone and speech applications, it is easy to find messages that reflect the norms of written rather than spoken languagefor example, "I must first verify your account number" and "You may now record your message." Spoken and written language differ for a number of reasons (Crystal 1992). Speech is dynamic and transient. No sooner is a word or phrase spoken than it's gone forever. Also, in spoken language, the participants are present. Typically, they interact face-to-face. The speaker usually has a clear notion of the person he or she is addressing.

Writing, in contrast, is static and permanent. Written words are there for as long as the medium lasts. Because of this permanence, written language is generally more formal than spoken language, and it ensures the survival of certain conservative words, phrases, and grammatical structuresfor example, "To whom it may concern." In addition, there is always a lag, often of unknown duration, between the time of the writing and the time of the reading, as well as different settings, which may also be unknown, where the writing and the reading take place.

Another important difference is that the writer is distant from the reader, cannot see the reader, and may not know who the reader is. Of course, this is also the case with speech applications. However, all speech applications crucially depend on spoken language, which has evolved in a different direction than writing because speech is *prototypically* face-to-face and *prototypically* more personal. To find yourself participating in a conversation, even one that has been engineered, with a persona who is reading formally written text at you, the content of which depends on your own unrehearsed, spontaneous responses,

is unprecedented in authentic discourse. Because it cannot be likened to any real-world experience, we can actually consider it an "antimetaphor," and so this kind of artificial formality is undesirable for interface design. As we've mentioned before, the highest-level metaphor that will most easily usher the VUI user from recognition state to recognition state is that of an everyday conversation. VUI prompts should therefore adhere to spoken rather than written norms, still guiding and directing the user through the interaction.

In the sections that follow, we examine a few differences between speaking and writing in English and demonstrate their relevance to prompting. These differences motivate the methodological suggestion in Chapter 8 to read your prompts aloud; as you listen to them, you will react differently than when you merely read them on the printed page.

10.4.1 Pointer Words

One manifestation of the many differences between spoken and written language lies in their use of **pointer words**for example, *this, that, here,* and *there*. The most basic use of pointer words is to indicate position or location of things in physical space: "*This* car isn't for sale, but *that* red one over *there* is." As this example illustrates, the distinction is roughly equivalent to "near" versus "far." It is typical, however, for languages to extend the use of the physical pointers to the domain of discoursein other words, to point either forward or backward in a stretch of written or spoken language. Examples: "Here's what happened" (pointing forward to what is about to be said) and "There you go again!" (pointing backward to what someone else has just said).

Prompt writers should be aware that the directionality of *this* versus *that*, in particular, differs depending on whether they are used in spoken versus written language, as shown in Figure 10-3. These examples demonstrate that the back-pointer of choice in speech is *that*, whereas the back-pointer of choice in writing is *this*.

Figure 10-3. In speech, that points backward, but in writing, this is often preferred.

Speaking, Back-Pointing
I tried to open the door, but *that* didn't work.

Writing, Back-Pointing
I tried to open the door, but *this* did not work.

The next pair of examples (Quirk and Greenbaum 1973) relates to speaking only. As shown in Figure 10-4, *that* points backward, and *this* points forward.

Figure 10-4. In speech, that points backward, and this points forward.

Speaking, Back-Pointing
They get Newsweek. I'd never subscribe to a magazine like *that*.

Speaking, Forward-Pointing
He told it like *this:* George was running down the road and all of a sudden…

In conversation, therefore, statements such as the ones in (32), (33), and (34) are most easily interpreted as forward-

pointing. (In other words, "this is" is synonymous with "here's.")

(32)

This is the last item on the list.

(33)

This is your last message.

(34)

This is the end of your favorites list.

Although these statements are used and interpreted as forward pointers in everyday conversation, they are sometimes used in VUIs to point backward, in compliance with the norms of *written* English. The unfamiliar use of spoken *this* as a back-pointer in these dialogs would likely impede listening comprehension.

An example of a prompt that uses both *this* and *that* appropriately is shown in (35).

(35)

This is the number I heard: 555 749 9591. Did I get *that* right?

This points forward to the phone number and could be replaced with "here." *That* points backward to the number and could be replaced with "it."

10.4.2 Contraction

Spoken language tends to favor contractions (*you're, can't, it's, don't*), whereas written language tends to avoid them (*you are, cannot, it is, do not*). Example (36) is a conversational prompt with three contractions.

(36)

. . . Finally, if *you're* finished with your bookmark list, just say "*I'm* done," and *I'll* put it away.

Do not expect a professional voice actor to take the liberty of substituting contractions for uncontracted forms. Although your voice actor may do you this occasional favor, those in the business consider it unprofessional to edit scripted material extemporaneously unless there is some understanding or agreement to the contrary.

Contractions have occasionally been discouraged in VUI designs based on the incorrect assumption that they are characteristic of "lazy" speech. If contractions were lazy, then people who are feeling lazy should be able to respond affirmatively to a question such as, "Are you hungry?" with simply "Yes, I'm." Native speakers of English unanimously reject the contraction in such cases. Another impossible contraction is "you're" in "Sandy's tired, and you're, too." Such examples suggest that the distribution of contractions in speech probably has more to do with syntactic and rhythmic constraints than with undesirable personality traits such as sloth or carelessness.

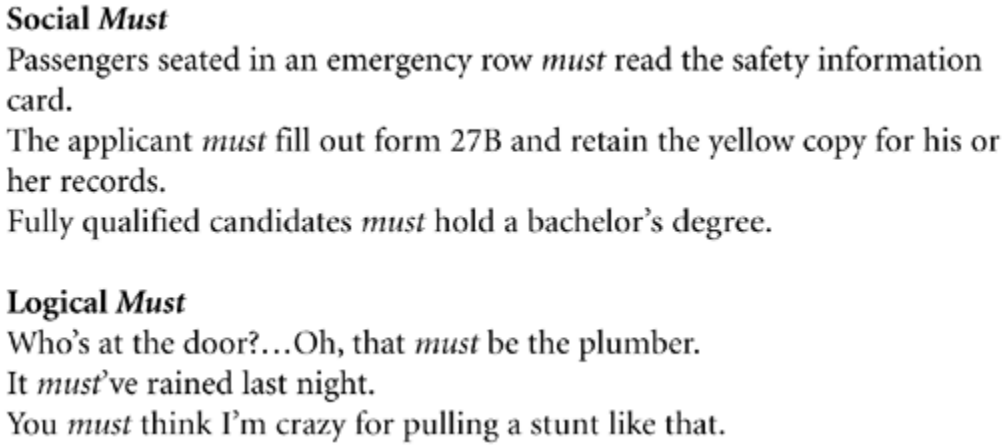
In other cases, contractions have been avoided on the grounds that the content of the VUI is, for example, financial, and banking demands a certain level of formality. There is no evidence, however, that people stop using contractions in speech just because they work in a bank or are trading shares in a volatile market. Contractions should be used in your VUI to the extent that they are a defining

characteristic of everyday spoken language, regardless of the conversation's level of formality. The point is to provide users with a familiar, comfortable language experience.

10.4.3 *Must* and *May*

Must and *may* have two basic meanings each. One can be loosely described as social/interactive, and the other one as a logical/probability meaning (Celce-Murcia and Larsen-Freeman 1999). The social/interactive meaning of *must* is "obligation" or "necessity," and the logical meaning is "deduction" or "inference" (see Figure 10-5).

Figure 10-5. Must has social as well as logical meanings.



Social *Must*

Passengers seated in an emergency row *must* read the safety information card.

The applicant *must* fill out form 27B and retain the yellow copy for his or her records.

Fully qualified candidates *must* hold a bachelor's degree.

Logical *Must*

Who's at the door?…Oh, that *must* be the plumber.

It *must*'ve rained last night.

You *must* think I'm crazy for pulling a stunt like that.

According to Melrose (1999), in speech, native speakers of North American English reserve *must* for inferences, which is the logical/probability meaning. In other words, although people use *must* to signal obligation in writing, they avoid it in speech, even in formal settings. Instead of *must*, the expression they use for obligation or necessity is generally *have to* (e.g., "Sorry, but I *have to* leave now").[4] Finally,

Melrose notes that the social *must* seems to be more common in British Englishfor example, "You *must* come over for dinner soon" and "We *must* correct that problem as soon as possible."

[4] Melrose also found that *(have) got to* or *have gotta* is used in both social and logical contexts, especially when there is some sense of urgencyfor example, "You gotta lend me ten dollars" (social use signaling obligation) and "You've gotta be kidding me" (logical use signaling inference).

The stilted quality of the prompts in (37) and (38) can therefore be repaired by replacing social *must* with more conversational alternatives, as in (39) and (40), respectively.

(37)

You *must* say your PIN one digit at a timefor example, two one zero zero.

(38)

We *must* first get your starting point.

(39)

Go ahead and say your PIN one digit at a timefor example, two one zero zero.

(40)

First, let's get your starting point.

Like *must*, the verb *may* also has social/interactive versus logical/probability uses, and also like *must,* it is only the logical/probability use that we find in everyday conversations. The social/interactive use of *may* is to ask for or grant permission, whereas the logical use is to express possibility (see Figure 10-6).

Figure 10-6. Like must, may has social as well as logical meanings.

Social *May*
You *may* leave the room.
May I be excused?
Cell phone and two-way pagers *may* not be used during the flight.

Logical *May*
It *may* rain tomorrow.
I *may* have a couple of twenties in my wallet.
You *may* not be getting enough sleep.

With social *may*, the speaker is implicitly referencing a position of authority and an inequality in status, so this is a relatively formal way to grant or request permission. In these cases, *can* is more egalitarian and friendlier. In North America, there is less social stratification than in other language communities around the world, so, not surprisingly, *can* is more prevalent than social *may*, and it is preferred in contexts such as, "You can (may) go now" (Bailey 1999; Melrose 1999). Presumably, North Americans are put off by the implicit social inequality, formality, and coldness of *may* in its social use. In contrast, social roles and setting are irrelevant to the use of *may* in the logical/probability context. In Figure 10-6, the speaker is assessing the probability of rain, independent of the social aspects of the conversation (Celce-Murcia and Larsen-Freeman 1999).

In the interest of projecting a more favorable persona, social *may* in (41) and (42) should be reworded as in (43) and (44), respectively.

(41)

. . . At any time, you may also ask for help.

(42)

When you are finished recording, you may hang up or press pound for more options.

(43)

. . . At any time, you can also ask for help.

(44)

When you're finished recording, feel free to hang up. Or to hear some more choices, press the pound key.

Note in (44) the contraction "you're" and the restructuring of information to accommodate the End-Focus Principle.

10.4.4 *Will* Versus *Going To*

In general, *will* is more formal and impersonal as an indicator of futurity, whereas *going to* is more informal and personal. Because it is more formal, *will* is often preferred in writing. For native speakers of English, however, sometimes these verb forms cannot be interchanged. In conversation, *will* is commonly used to signal spontaneous volition, as in (45) and (46), and future contingency, as in (47) and (48). In contrast, *going to* signals planned intention, as in (49) and (50), and "the future on its way," as in (51) and (52).

(45)

A: That was my mother's favorite vase.

B: Oops! I'll buy you another one.

(46)

A: I can't really talk right now.

B: Okay, I'll call you tonight.

(47)

If you buy a house, you'll get a huge tax write-off.

(48)

If you put your pawn there, *he'll* win the game.

(49)

A: What's Harvey doing with my pliers?

B: He*'s going to* fix the TV antenna.

(50)

A: What's on your agenda for tomorrow?

B: I'*m gonna* get a haircut.

(51)

Stop the car! I'*m going to* be sick!

(52)

Look, it'*s going to* rain.

If *going to* in these examples is replaced with *will*, the result is very unnatural, and yet *will* is often used in VUI prompts where we would conversationally expect *going to*. Take, for example, the prompt in (53), which is part of a lengthy enrollment process in a subscriber-based service and is followed by instructional information.

(53)

I will now record your account number. Say the number one digit at a time.

This prompt can be rewritten to conform to conversational norms by replacing "I will" with "I'm going to." In addition, the placement of the adverb "now" between the two verbs ("will" and "record") yields another structure particular to written discourse or in some formal spoken genres, such as when a professional magician says, "I *will now saw* my lovely assistant in half." We can rewrite (53) as (54).

(54)

Now I'm going to record your account number. Tell me the number one digit at a time.

Another example comes from a VUI that requires several steps to provide the caller with driving directions. At one point in the sequence, we receive some helpful orientation, shown in (55).

(55)

We will now collect your address.

Although this prompt is intended to be heard over the telephone, it is representative of written discourse. First, we hear *will* instead of *going to*, even though planned intention is a better metaphor in this context than spontaneous volition. Again, we find the placement of "now" between the auxiliary ("will") and the main verb ("collect"). Finally, the use of "collect" in the sense of "elicit information" is unnecessarily technical. This prompt can be recast as shown in (56).

(56)

> Now I'm going to ask you a few questions to find out where you're going.

This is longer than the original, but it is easier to listen to, because it avoids those discourse features that are alien to spoken language and instead exploits familiar linguistic conventions.

10.4.5 "Romans Perspire, Anglo-Saxons Sweat"

We've noted that written language often favors certain words, phrases, and grammatical structures that are avoided in speech. This section focuses on vocabulary in spoken versus written language.

The title of this section, "Romans perspire, Anglo-Saxons sweat," alludes to the richness of the vocabulary of the English language. Owing to our dual inheritance of words from the Romance and Germanic language families, we often have recourse to two words that are roughly synonymous, as in the case of verbs such as *sweat/perspire, put out/extinguish,* and *leave/exit*, nouns such as *fish tank/aquarium, crowd/multitude, lunch room/cafeteria,* and *drink/beverage*, adjectives such as *kingly/regal, funny/humorous, friendly/amicable,* and *fat/corpulent*, adverbs such as *yearly/annually,* and so on. In terms of usage, however, Latin-derived alternatives (such as *perspire* and *extinguish*) usually have a more

formal, technical ring, whereas their Anglo-Saxon counterparts sound more informal and colloquial.

When we write, especially in the case of more careful genres, we gravitate toward more formal (Latin-derived) alternatives, given the relative permanence or persistence of written messages. The writing of prompts, which are intended to be spoken, should avoid this tendency. The list of verbs in the left column in Table 10-1 was culled from a review of several speech and touchtone applications; the right column shows more colloquial, contextually appropriate equivalents.

Table 10-1. Formal Versus Colloquial Verbs

FORMAL	COLLOQUIAL
acquire	get
activate an account	set up an account
create an account	set up an account
create a bookmark	make, add a bookmark
encounter [information]	find, come across

FORMAL	COLLOQUIAL
encounter [difficulty]	have problems
exit list	be done with a list
experience difficulties	have problems
obtain	get
pause	take a break
provide	give
receive	get
request, collect	ask for
respond	answer
return	go back
select	choose, pick
terminate	end, finish

10.5 Register and Consistency

Register has to do with the level of formality of a piece of discourse. Compare, for example, (57), (58), and (59).

(57)

To whom do you wish to speak?

(58)

Who would you like to speak to?

(59)

Who do you wanna talk to?

Example (57) is the most formal, and (59) is the least. The following specific linguistic features differentiate the register of these questions:

- Formal "To whom . . .?" versus informal "Who . . . to?"
- Formal "wish" versus less formal "would like" versus "want to," pronounced colloquially as "wanna"
- "Speak" versus more informal "talk" to mean "have a conversation"

For Halliday (1994), register is a multidimensional construct that consists of three components:

1. **Mode** refers to the channel of communicationfor example, written versus spoken, in person versus remote. This parameter is invariable in VUIs.
2. **Field** has to do with the content of the discourse as well as the social setting in which the language is being used. Field is often reflected in word choice.
3. **Tenor** involves the roles and relationships of the participants. In other words, who is talking to whom? Imagine, for example, that the caller has just heard,

"Sorry, but there's a problem with that passcode" and is then asked to visit the registration Web site. In (60) through (63), we have four wording possibilities. Although their function in context is synonymous, they differ in tenor.

(60)

You must visit the registration Web site at phone dot ACME Widget dot com.[5]

[5] As we have seen, the use of social *must* in (60) is not typical of spoken English, although it is popular in touchtone and VUI applications.

(61)

Please visit the registration Web site at phone dot ACME Widget dot com.

(62)

. . . but why don't you visit the registration Web site at phone dot ACME Widget dot com?

(63)

You might want to visit the registration Web site at phone dot ACME Widget dot com.

Because (62) and (63) are indirect requests, they imply a social relation between the speaker and listener that is different from that of (60) and (61); the speaker in (62) and (63) is more polite and more deferential.

Furthermore, these wordings illustrate an important linguistic concept: The literal, superficial meaning of an utterance is often different from the meaning it conveys in context. Only literally is (62) a question, and only literally does (63) assert the possibility of a desire ("might want"). In actuality, these are socially intelligent ways of telling someone to go do something. In any case, the tenor of

messages in an application depends largely on the persona chosen for that application, along with the persona's intended social role vis-à-vis the user.

You must consider mode, field, and tenor when writing prompts. Whatever you decide in terms of the most appropriate register for your application, make sure it's exercised consistently throughout your dialog. Inconsistent register mars the first draft of an otherwise impeccable dialog design for a money transfer demo shown in (64) through (67).

(64)

What account do you want to transfer from?

(65)

What account do you wish to transfer to?

(66)

I'm sorry, I didn't understand. Say the name of the account you wish to transfer money to. For example, you could say, "Savings account."

(67)

You wish to transfer five hundred dollars from your savings account to your money market account. Is this correct?

These prompts transmit social cues that clash. This particular use of *wish* (6567) is literary, and in speech it conveys social distance and impersonality.[6] In addition, (67) reflects formal written discourse in its use of *this* as a back-pointer. On the other hand, sentence-final prepositions (6466) are conversational. Example (66) uses the formal, distant, impersonal-sounding *wish*, but the use of "could" (in "you could say, 'Savings account'") to make a request is

relatively personal, indirect, and gentle. (Compare, for example, "You *must* see a doctor" with "You *could* see a doctor.") Dissonant social cues in these prompts undermine the establishment of a single voice belonging to a coherent personality.

[6] There is a conversational use of *wish*, but it expresses contrary-to-fact desire, as in "I wish it would stop raining" or "I wish I had a million dollars."

When reviewing dialog specifications, we occasionally get feedback about the use of prepositions at the end of sentences. Consider the following:

(68)

AUTO ATTENDANT: Who would you like to speak to?

(69)

TRAFFIC: What "hot spot" do you want a report for?

(70)

BANKING: What account are you transferring from?

(71)

Please say the name of the person *to whom* you'd like to speak.

(72)

Please tell me the "hot spot" *for which* you would like a traffic report.

(73)

Please say *from which* account you would like to transfer funds.

In actuality, the use of *who* instead of *whom* in (68) and the sentence-final placement of prepositions in (68) through (70) is not only acceptable in conversation but also characteristic of conversation.

10.6 Jargon

Jargon is defined as "the technical or specialized language of a trade, profession, or similar group." Like most other professional groups, speech technologists use a fair amount of jargon. Speech technologists include dialog designers, linguists, software engineers, and other speech scientists. Jargon is acceptable if the speaker is communicating with others in the same group. Otherwise, the use of jargon can induce reactions ranging from incomprehension to mild alienation.

In a demonstration of a VUI designed to furnish driving directions, we happened to hear the prompt in (74), although the jargon-free wording in (75) would have been more effective.

(74)

An error has been generated. Returning to Main.

(75)

Sorry, there was a technical problem, so we'll have to go back to the main menu.

The wording of the first part of (74) reflects the technical concept of error generation. In the second part of the prompt, "main" is a shortened form of "main menu." This truncation is frequent in the conversation of dialog designers, who may easily overlook the association between "Returning to Main" and the same-sounding "Returning to Maine." Although the context makes the meaning clear, this humorous connection comes more readily to listeners who are unfamiliar with dialog design.

In a dialog specification document for an application intended for airline employees, we encountered the message in (76). In this context, however, a more appropriate, user-friendly message would be the one in (77).

(76)

No travelers are defined for this employee.

(77)

Sorry, but it looks like the employee hasn't designated anyone for flight benefits.

The element of jargon is this particular use of "define," which is familiar to programmers and database specialists but not to typical users of this travel industry VUI.

Note, however, that jargon in itself is not always undesirable in a VUI. In this same interface, an example of appropriate jargon is the frequent use of the verb phrase "list for (a flight)," as in "Do you want to list for this flight?" When an airline employee "lists" for a flight, he or she is making an unconfirmed booking. This is different from "waitlisting," because employees can "list" for a flight that has plenty of availability. In any case, "list" is appropriate here because it is the jargon of airline employeesin other words, jargon that the users themselves will expect to hear and want to use in the interface.

10.7 The Cooperative Principle

We have been looking at how to craft prompts so that they sound natural in context. In this section, we turn our attention to the natural interpretations that listeners make when they hear messages in context.

Conversation is more than just people exchanging information. When we participate in a conversation, we follow the **Cooperative Principle**: We "get along" by sharing assumptions and expectations regarding the topic, how the conversation should develop, the quality and quantity of the contribution that each participant is expected to make, politeness, consistency, and so on. Conversation's naturally cooperative undercurrent enables us to interpret each other's utterances. Consider the examples in (78) and (79).

(78)

A: I saw Mark having dinner with a woman last night.

(79)

A: What a day! I need a drink.

B: Have you ever been to the Eagle?

Assuming that we all know that Mark is a married man, the sentence in (78) means that Mark was dining with a woman who was not his wife, even though his wife is of course a woman. Presumably, if Mark had been dining with his wife, the speaker would have said so, in the interest of being cooperative. In (79), if speaker A acts on the assumption that B's reply is relevant, A will assume that B's question is intended to function as a recommendation to go to a place called the Eagle. Again, in the spirit of cooperation, A can further interpret that the Eagle is nearby, is open, and is a place that serves drinks. Imagine if one of these conditions were not true. If, for instance, the Eagle were hundreds of miles away, B's conversational behavior would indeed be inappropriate.

Because the meanings of messages in conversations are not tied to their literal, superficial meanings, prompt writers must take care with the interpretations that listeners will infer in context. In example (80), the welcome message permits an inference that is later contradicted by the main menu prompt.

(80)

WELCOME PROMPT:	Hello, and welcome to the Frequent Buyer Rewards Line. You can now redeem points online, at www dot frequentbuyer dot com, forward slash, Points.
MAIN MENU PROMPT:	How can I help you? <pause> You can say, "Buy points," "Redeem points," "Transfer points," ...

The wording of the welcome message allows callers to infer that they must go online to redeem points. In other words, it seems as though this feature is not supported by the speech interface; otherwise, why would the VUI's persona be providing the Web address? Despite this legitimate interpretation, "redeem points" turns up among the main menu options in the very next prompt.

A corollary of the Cooperative Principle requires that speakers be informative. Conversation analysts refer to this principle as the **Maxim of Quantity** (Grice 1975), which holds that a speaker's contribution to a conversation is as informative as is required to advance the perceived purpose of the conversation. This principle is sometimes

violated in speech applications. Example (81) is from a voice mail system. The context is as follows: The subscriber has just logged in with a passcode and is receiving his or her new message count.

(81)

The following [two, three, four . . .] *new* messages have *not* been heard.

Assuming that the system's contribution is informative, the caller has no choice but to infer that sometimes *new* messages can *already* have been heard. The message strikes a chord of dissonance because, for most people, a "new message" is one that has not yet been heard. After using this voice mail system for a while, you may (or, sadly, may not) realize that a "new message" is intended to refer either to a message that you have never heard or to a message that you *have* heard, even if in part, but have not yet saved or deleted (e.g., when you skip a message). A more straightforward version of the message would be simply, "You have [two, three, four . . .] new messages." Giving common words new and counterintuitive definitions can impede comprehension and adversely affect the usability of a VUI.

From these examples involving the Cooperative Principle and corollaries such as the Maxim of Quantity, we see that

conversations are more than literal messages designed to exchange information. For Richards (1980), conversation "consists of exchanges which are initiated and interpreted according to intuitively understood and socially acquired rules and norms of conversational cooperation, which can in turn be manipulated to create a wide range of meanings beyond the level expressed directly by the utterances in the conversation themselves." This chapter argues that interactions with speech applications must not be treated as exceptions in this regard.

10.8 Conclusion

We approach all verbal tasks with certain expectations having to do with the linguistic form of utterances in context. Speech interfaces are no exception, and the design team should regard them as conversations. Dialog designers and prompt writers should take advantage of the fact that everyday conversation is the communication system most familiar to our users.

We occasionally encounter resistance to conversational prompt design on the grounds that spoken language is thought to be inherently inferior, careless, or too relaxed for the customers of Company X. In these cases, prompts that lack contractions or everyday cohesion devices are endorsed under the rubric of "better writing." But conversational language is a sophisticated system of its own, having developed for reasons of its own. Better prompting is not a question of "better *writing*" in the traditional sense. Unlike written language, spoken language has evolved from prototypically face-to-face interactions over the course of millions of years. Spoken language incorporates features and principles that are specially adapted for this mode of communication and that we should observe and exploit in prompting.

To leverage users' familiarity with spoken language, we make the following recommendations, based on observations of the form of naturally occurring language:

- Attend to the differences between spoken and written language. These two types of communication have evolved differently owing to the nature of the signal (ephemeral versus persistent). What people are used to hearing is not what they are used to reading.

- Use cohesion devices (such as pronouns and discourse markers) appropriately. They enhance the functional unity of a dialog, reinforcing relationships of meaning between messages. Listeners rely on them for comprehension.

- Observe and exploit the principles of information structure. These principles dictate where listeners naturally expect to retrieve old versus new information (as determined by context).

- The sociolinguistic phenomenon of register reflects directly on your brand and corporate image. In particular, consider mode, field, and tenor, which are essential ingredients of persona design. Inconsistency in any of these areas can leave users with an unfavorable impression. Stick with a style that befits the content and the desired social role of the VUI persona vis-à-vis the user.

- Jargon is another sociolinguistic phenomenon that at times should be avoided, at other times exploited. Appropriate use of jargon begins with knowing who your user is.

- The Cooperative Principle accounts for how and why language users read between the lines. Consider the possible inferences that users will draw from the messages you compose. Be informative, but not in excess, and anticipate how users will be informative in responding to prompts.

Successful prompt writing depends on a view of the VUI as much more than a mechanism for collecting information via acoustic signals. A user-centered speech application is

really a *language* interface, where "language" implies both a cognitive dimension (e.g., use of cohesion devices, principles of information structure, and pointer-word directionality) and a social dimension (e.g., jargon, vocabulary and grammar choices, and register). VUI designers who are mindful of these dimensions of linguistic experience are well on their way to creating a VUI that will engage users and project a favorable corporate image or persona.

It is also of paramount importance to consider the form of prompts *in context*. A prompt is not a simple, isolated translation of a recognition return, such as <action GET_QUOTE>, or "Too Much Speech." Every prompt is like a small piece of tile in a mosaic, in that the whole is perceived as greater than the sum of the parts. When you write prompts, always consider the context of the dialog, because the structure of naturally occurring language is itself context-sensitive. Context sensitivity is what listeners expect, whether they are participating in a human-to-human conversation or an engineered dialog.

Conversational language is one of Mother Nature's greatest masterpieces. Prompt writers and dialog designers should regard it as a source of inspiration for user-centered design, tempered with an awareness of the limits of the current technology.

Chapter 11. Planning Prosody

The number you have dialed, Four? Four? Four? Four? Four? Four? Four? Is not in service. Please check the listing and dial again. <click!>

Biased by the immutability of the printed word, most people think a "four" is a "four" is a "four." Actually, the digit's context in the string and the string's context in the larger utterance make a big difference in how this word is spoken. Printed representations such as "four" and "4" do not do justice to the richly textured system of spoken language, which is, of course, the essence of voice user interfaces. In fact, there are many ways to say the word "four," depending on context. This is prosody.

Chapter 10 argues that VUIs should exploit the communication system that users are most familiar with. To this end, we look at natural (human-to-human), everyday spoken language and distill certain features and principles that can be applied to VUI prompting. The focus of that chapter, however, is wordingwhat to say, what words to use, and how to arrange them in a prompt. In this chapter we look at how the naturalistic prompts we write should actually *sound* in context. In technical terms, what is their ideal prosodic structure? For a given recording, what prosody will be the most familiar, the most comfortable, and the most comprehensible to users? What prosody will

enhance users' perception of the interface and the persona that fronts it? Just as context plays an essential role in determining the form of prompts, context is also of utmost importance in the prosody of prompts.

The following is a high-level outline of the rest of this chapter:

- What is prosody?
- Functions of prosody
- Stress
- Intonation
- Concatenating phone numbers
- Minimizing concatenation splices
- Pauses
- TTS guidelines

11.1 What Is Prosody?

The sounds of language express meaning both verbally and nonverbally (Crystal 1995). **Verbal meaning** refers to the use of vowels and consonants to build up words, phrases, and sentences. Chapter 10 is concerned with verbal meaning in engineered dialogs, but the conversational wording of prompts is only half the story. The other half is how the message is delivered. To impart structure and expression to meaning, speakers use intonation, stress, rhythm, tone of voice, and silence in the form of pauses. Linguists refer to this **nonverbal meaning** as **prosody**.

Many of the same issues that you must consider to achieve naturalness in prompt wording also account for naturalness of prosody. It is not enough to consider only the linguistic structure of prompts as they appear in written form. The focus should be on how the user will hear them in context, which is fundamental in shaping the prosody of utterances in human-to-human interactions. This chapter outlines the basics of prosody and explains its importance in encoding and decoding meaning.

Prosody is a level of linguistic analysis that is systematic and principled. Prosody is an element of **grammar**, the implicit knowledge that native speakers have about their language.[1] Just as syntax is a component of grammar, so is prosody. Of course, we would never intentionally write prompts that defy the rules of syntax, as in "Please to say now you PIN," so we must not ignore basic rules of prosody. This chapter demonstrates how attention to prosody pays off in interactions that are more comprehensible and more comfortable for the VUI user.

[1] In this chapter we use a broader definition of "grammar" than earlier in the book. Earlier, we used "grammar" to refer to the language knowledge we, as designers and grammar

developers, supply to the machine. Here, it is used more generally to refer to the language knowledge a human has.

11.2 Functions of Prosody

Intonation helps us identify grammatical structure (Crystal 1995). Without prosody, it is hard to imagine how listeners might parse an incoming phonetic stream into clauses and sentences. Consider the contrast in (1) through (3), with paraphrases given in square brackets.

(1)

You know. *I* don't. [So don't ask me.]

(2)

You know, I *don't*. [As a matter of fact, I really don't.]

(3)

You *know* I don't. [You know that I don't.]

In addition, many grammatical contrasts rely on intonation systematically, as in the case of statements versus certain types of questionsfor example, "I should" versus "I should?"

Intonation also serves a textual function, communicating relationships of contrast and coherence among larger units of talk. Newscasters, for example, shape monologs into distinct paragraphs and news stories. Sportscasters rely on special prosodic patterns to report the progress of fast-paced events and mounting action in, for example, soccer games, basketball games, and horse races.

Prosody is also informational. It draws the listener's attention to what is new and reveals the speaker's belief, intent, and knowledge in ways we take for granted. To appreciate the degree to which subtle differences in prosody effect sharply divergent informational meanings, consider five potential interpretations of the simple sentence "I should go," as shown in Table 11-1 (adapted from Quirk and Greenbaum 1973).

Similarly, the intonation of "any" can reverse the meaning of an utterance such as, "Don't get her ANY flowers." If "any" falls, the meaning is "Don't get her any flowers at all," but if "any" rises and falls, it means "Don't get her just any flowers; make sure they are special." These examples demonstrate the central role that prosody plays in mediating the interpretation of linguistic forms.

Table 11-1. Five Meanings of "I Should Go"

	PROSODY	MEANING
I should GÒ.	Falling tone on stressed "go"	[Neutral, default.]
I should GÓ?	Rising tone on "go"	Is that your advice?
Ì should go.	Falling tone on "I"	Not you!
I SHÒULD go.	Falling tone on "should"	And I defy you to deny it!
I SHÔULD go.	Rising-falling tone on "should"	But I don't think I will.

Prosody also serves psychological functions, helping us to organize speech into units that are easier to perceive and remember (Crystal 1995). A sequence of ten digits, as in North American phone numbers, would be difficult to remember if we did not rely on prosody to chunk them into three groups.

A more general function of prosody is to convey attitude and emotion. For example, in response to the question, "Will you marry me?" Crystal (1995) registers nine nuances of meaning for the response "Yes" in the prestige accent of England known as "RP" (for Received Pronunciation). These shades of meaning include delight (a rise-fall contour, as in "Wonderful!"); detachment, boredom, or resignation (a level low tone, as if to say, "If I must" or "I give up"); and guarded suspicion (a low rise contour, as if to ask, "What's the catch?") Finally, prosody conveys personal and social identity. For example, people pursuing certain occupations are readily identified through their distinctive prosodic patterns: courtroom lawyers, evangelical preachers, newscasters, sports commentators, drill sergeants, and so on (Crystal 1995). Flight attendants, for example, have a distinctive habit of shifting stress onto certain words when making announcements over the aircraft's public address system. For example: ". . . a small orange oxygen mask *will* descend from the compartment situated immediately *above* you," "Upon opening the overhead bins, *please* be aware that contents *may* have shifted *during* flight," and "We additionally request that all passengers remain *in* their seats *until* the captain *has* turned off the fasten-your-seat-belt sign."[2] This function of prosody in particular can be exploited in the design of high-end, persona-rich VUIs.

[2] Flight attendant prosody can be explained as an attention-getting device.

Now that we have reviewed the main functions of prosody in everyday, authentic language use, let's examine some high-level, but essential, features of English prosody and then apply them to prompting.

11.3 Stress

Stress is the variation in loudness that differentiates strong and weak syllables, a variation that in turn characterizes a word's spoken identity (Crystal 1995). Thus the word "originality" can *only* be the word "originality" if primary stress falls on the "nal" syllable. If we stress the "lit" syllable instead, the word no longer means "originality." The word "produce" means two very different things depending on which syllable is stressed: either the verb that is synonymous with "create" or the noun that refers to fruits and vegetables.

Stress also applies to longer utterances. For example, in the question "Does he like it?" sentence-level stress falls on the verb "like." The use of stress at both the word and the sentence level is critical for intelligibility. Non-native speakers of English who fail to master the stress patterns of this language may have a hard time making themselves understood.

The chief function of sentence-level stress is to signal informational prominence, or focus. The focal point of an utterance is the piece of information the speaker intends the listener to consider "new" and to pay special attention to. By default, in English we focus on the last open-class element of the sentence (as discussed in Chapter 10). The default stress position in "Does he like it?" is therefore on "like," because this is the last open-class element of the question; "it" is a pronoun and therefore a closed-class item. For further illustration of this default stress principle and how it can be overridden, consider example (4).

(4)

Somebody must have TAKEN it.

The last open-class item in this sentence is the participle "taken," and so it receives stress by default. Less commonly, however, other parts of the sentence may also host prominence, as in (5) and (6), where stress functions contrastively.

(5)

Somebody MUST have taken it. [It's no use your arguing.]

(6)

SOMEbody must have taken it. [. . . even if *you* didn't.]

This is known as **contrastive stress**, which refers to the way that people override default accent patterns to highlight any word (or portion of a word) they please. Unlike default stress, contrastive stress can even fall on closed-class items, which are not stressed under normal (default) conditionsfor example, "and" in "John AND Mary went? Together???", "un" in "I thought she looked UNhappy," and "the" with a long *e* in "The Four Seasons is THE place to stay on the Big Island."

Users benefit significantly when you use contrastive stress appropriately in your VUI. Consider example (7).

(7)

SYSTEM: I heard "Tennessee." Is that RIGHT?

CALLER: No.

SYSTEM: What about "Texas"? Is THAT right?

"Is that RIGHT?" expresses the default stress pattern; stress falls on the last open-class item of the sentence, which is the adjective "right." In the second prompt ("Is THAT right?"), stress falls on the pronoun, which is a closed-class

item, for contrast. By shifting stress from "right" to "that" as the dialog progresses, the VUI's persona effectively gives the impression that he or she is keeping track of the conversation, as in authentic, human-to-human conversations. The appropriate use of contrastive stress endows the persona with humanlike linguistic intelligence as well as an attentive character.

In contrast, there are VUIs that repeat the same prompt, such as "Is this correct?" several times in a short sequence, as many times as there are confirmations needed. This context-insensitive repetition is unnatural and prosodically unfamiliar to users. Reminiscent of a broken record, it detracts from our perception of the persona.

When attention is not paid to linguistic context, the resulting prompts may contain stress anomalies, as in (8).

(8)

CALLER: Get my MESsages.

SYSTEM: You have no new MESsages.

The system response in (8) stresses "messages," as if this were established by the context as the new focal point, but it is not. Compare prosodically anomalous (8) with the more conversational, prosodically well-formed (9).

(9)

CALLER: Get my MESsages.

SYSTEM: You don't HAVE any messages.

In destressing "messages," the system response is prosodically appropriate. It appears to be keeping track of the conversation in a humanlike way, as a dynamic flow of information, and this is where the system in (8) fails. Whereas (8) simply issues a context-insensitive translation of the backend return "N equals zero," the context-sensitive message in (9) further manages the caller's assumption that there *are* messages to be retrieved, by shifting the informational focus to the negated verb of having.

A similar stress violation occurs in a bill payment application in (10).

(10)

CALLER: Bill payment.

SYSTEM: What bill would you like to PAY*?*

CALLER: MasterCard.

SYSTEM: How much would you like to PAY?

CALLER: Two hundred and fifteen dollars.

SYSTEM: On what date would you like it PAID?

Participating in this dialog, over the telephone, a user is likely to think, "Something sounds stilted or unnatural, but what exactly is it?" When we read the transcript of this dialog, however, the chief culprit leaps off the page, thanks to the use of SMALL CAPS. The verb "pay" bears sentence-level stress in every utterance in which it appears, one after the other, as if it were the new informational focus of each prompt.

The dialog sounds better if "pay," after it makes its debut, is prosodically defocusedin other words, destressed. There

are several other changes that can make the resulting dialog seem more natural, as shown in (11).

(11)

CALLER: Bill payment.

SYSTEM: Sure, bill payment. What bill would you like to PAY?

CALLER: MasterCard.

SYSTEM: How MUCH would you like to pay?

CALLER: Two hundred and fifteen dollars.

SYSTEM: And on what DATE would you like it paid?

To this point, our examples have featured cases where we would naturally expect to find contrastive stress, but

because the messages have been recorded with little or no attention to context, contrastive stress is inappropriately lacking. In the next example, however, we find the opposite scenario. Contrastive stress seems natural in the context of the recording session, but it turns out to be inappropriate in the context in which the caller will experience the recording. In the context of reporting the time of day in natural, everyday conversation, it is the "M" of "AM" and "PM" that is stressed, as shown in (12).

(12)

EIGHT ay-EM

FOUR pee-EM

In this context, "A" and "P" are weak. The strong syllables are those at the **edge** (the beginning and end) of each phrase, that is, the hour and "M." In the recording studio, however, voice actors sometimes stress "AM" and "PM" contrastively (13).

(13)

AY-em

PEE-em

As a result, the stress pattern in (13) would sound appropriate only in the context of questioning or clarifying "AM" *versus* "PM," as in, "Did you just say 'AY-em' or 'PEE-em'?" or "The flight isn't leaving at six AY-em; it's leaving at six PEE-em." Unfortunately, this stress pattern is unsuitable for reporting the time in a neutral way. Instead of hearing prosodically well-formed (12), what we hear instead is anomalous "EIGHT | AY-em" and "FOUR | PEE-em," where the vertical bar (|) indicates a concatenation break.

The case of "AM" and "PM" is only one example of a commonplace prosodic violation in concatenated messages that can be explained as an artifact of inadequate scripting. That is, concatenation units are often recorded in simple list format, without the benefit of contextual cues. Whoever is recording prompts, however, should be aware that lists are subject to their own prosodic patterns, which are not always compatible with the intended use in the dialog. The concatenation result is prosodically ungrammatical, meaning that it sounds strange.

Often, the use of contrastive stress also affects the prosody of numbers. For example, when we count, we naturally stress the unit preceding "teen," as in (14). In other words, we count contrastively.

(14)

. . . THIRteen, FOURteen, FIFteen, SIXteen, SEVenteen, EIGHTeen, NINEteen . . .

Whether or not this stress pattern is appropriate in a specific dialog context, many voice actors will read a list of numbers on a script in just this way if they do not know the dialog context. This is only natural. Apart from such number lists, however, stress may fall on either the first or the second half of each item, depending on the specific syntactic context. Compare the stress pattern of numbers not followed by nouns in (15) with numbers followed by nouns in (16).

(15)

Message fourTEEN.

Today is June fifTEENTH.

Today's high will be twenty-two degrees Fahrenheit, with a low of eighTEEN.

The New York Mets defeated the San Diego Padres twenty to thirTEEN.

(16)

You have SEVenteen messages.

The refund amount is THIRteen dollars and twenty-eight cents.

It's NINEteen degrees below zero, with the wind-chill factor of . . .

. . . a SIXteen percent increase.

In (15), stress falls on the last syllable of the numbers ("-TEEN"), whereas in (16), it falls on the first syllable (e.g., "SEV-"). This is known as **stress shift**. It is beyond the scope of this chapter to provide a formal analysis of this phenomenon, but in short, the patterns in (15) and (16) reflect the preference in English for stress at the edges of certain kinds of syntactic units (Selkirk 1995). The phrase "THIRteen-year-old GIRL" satisfies this preference, whereas "thirTEEN-year-old GIRL" violates it at the left edge and is therefore prosodically anomalous.[3]

[3] This principle is superseded, however, when stress is required for contrastfor example, ". . . with a high of NINEteen and a low of THIRteen" and "The score was SIXteen to FOURteen."

Stress shifts are an example of a purely structural principle of prosody that is often violated when speech is concatenated. You must always consider context surrounding units such as "a.m." and "seventeen." To avoid creating prosodic anomalies, you should always contextualize concatenation units on the recording script for the director and voice actor (see Chapter 17).

11.4 Intonation

The most critical aspects of nonverbal meaning are often communicated through intonation. Technically defined, **intonation** refers to the linguistic use of movement of pitch, but we can think of it simply as melody in speech. Intonational meaning is expressed by different pitch levels, called **tones**, which are connected to form tonal sequences known as **contours**. Intonation is rarely represented in writing. One obvious case is the opposition between a statement and a yes/no question; another is the use of italics for emphasis. In general, however, intonational features can be written down only by using special transcription systems (Crystal 1995).

Our discussion of intonation is divided into two parts. First is a brief introduction to the basic intonation patterns or contours of American English. The second discusses the uses of these contours in context.

11.4.1 Basic Intonation Contours

Think of some different ways that you can say the number "four." You can say "four" as an answer to a question: "Four!" The same word can function as a question all by itself: "Four?" Yet another possibility is, "If you guessed 'four,' you're absolutely right."

These three ways of saying "four" exemplify the basic contours that characterize American English intonation patterns. There are many others, but these three are essential for prosody-sensitive concatenation. The basic contours are as follows:

- **Rising-falling, final** (e.g., "Four!")
- **Rising** (e.g., "Four?")
- **Rising-falling, nonfinal** (e.g., "If you guessed 'four' . . .")

To characterize these three contours, let's assume for simplicity only three *relative* levels of pitch: Level 1 is the lowest, level 2 is midrange, and level 3 is the highest.

A few disclaimers are in order:

- These numbers are not in any way intended to represent precise pitch values. They are best regarded as ranges of possible pitches. For example, the rising-falling, nonfinal contour both starts and ends at a midpitch level, but this does not mean that it starts and ends on exactly the same melodic note. Rather, the contour starts and ends in the same relative pitch range.
- The intonation system proposed here is not to suggest that there are only three levels of tone. For example, a fourth tone, higher than level 3, is often used in particularly emotive discourse, but this is not relevant to the VUIs we have deployed.

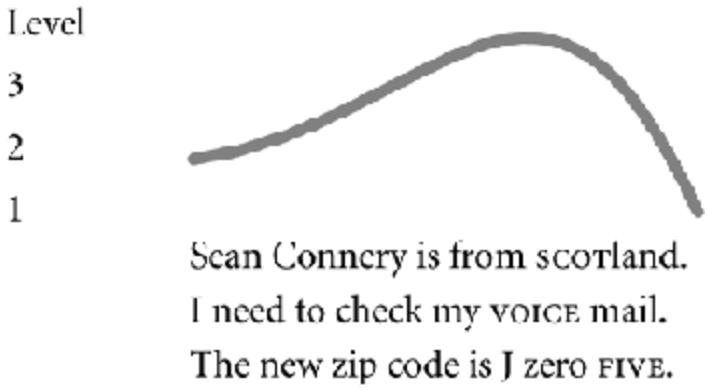
- The inventory of contours presented here is not intended to be exhaustive. They are only the most basic ones.

- The contours presented here are greatly simplified and are intended to serve as abstractions. For example, there may be a number of intonational peaks in any of these contours, depending on the details of sentence-level stress placement.

Rising-Falling, Final

Figure 11-1 shows the default pattern for simple declarative sentences. However, like many of the patterns that follow, this particular contour has additional uses. These are relevant for concatenation and are described shortly.

Figure 11-1. The rising-falling, final contour is the default pattern for simple declarative sentences.



Remember that the graphical representations of intonation contours throughout this chapter are not intended for accuracy but serve only as basic sketches. These abstractions are effective in illustrating certain key aspects of intonational grammar, in particular the concept of boundary tones, which is explained shortly.

The delivery of a statement as depicted in Figure 11-1 is neutral. By default, it begins at a midlevel, then rises, and then falls to a relatively low pitch. As noted earlier, the peak, which reaches level 3, predictably falls on the stressed syllable of the rightmost open-class item of the utterancehere, "VOICE mail." (The prominence peak is marked by the use of SMALL CAPS.) Of course, because the statement peaks on "VOICE mail," it cannot serve as a response to "WHO is checking voice mail?" or "What are you DOING with your voice mail?" In these cases, the peaks would be "I" and "CHECKING," respectively.

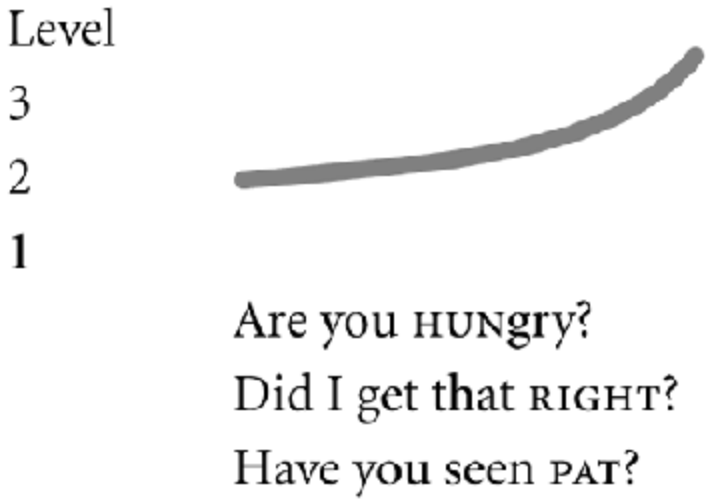
Significantly, in English intonation structure, intonational meaning is signaled at the right edge (the end) of the utterance, and often at the right edge of smaller phrasal constituents. These tonal markers are known as **boundary tones** (Pierrehumbert 1980). This concept is crucial not only for proper concatenation but also for the proper recording of prompts in general.

For convenience, linguists represent boundary tones by using the percent sign (%). This use of the percent sign has nothing at all to do with arithmetic; for example, "%1" means that a particular utterance or phrase ends at pitch level 1. Because the three intonation contours are distinct at their right edges and because the rising-falling, final pattern ends at pitch level 1, let's refer to this pattern as **contour 1**.

Rising

The rising contour (Figure 11-2) is most often associated with yes/no questions. This contour also serves other functions, described later.

Figure 11-2. The rising contour is often associated with yes/no questions.

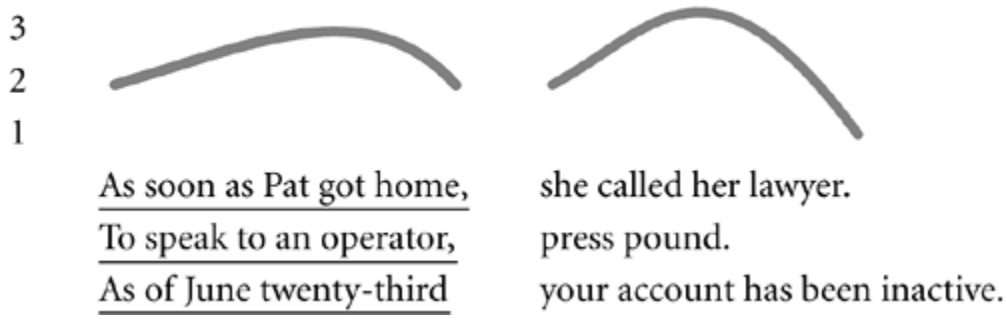


Like contour 1 in Figure 11-1, the rising contour starts at midlevel but ends in a high boundary tone, %3. We therefore refer to the rising pattern as **contour 3**.

Rising-Falling, Nonfinal

This pattern corresponds to the first of the pair of clauses (the dependent or subordinate clause) in each sentence in Figure 11-3. (The second in each pair is a type 1 contour, as described earlier.)

Figure 11-3. The rising-falling, nonfinal contour reflects complex sentences.



The dependent clauses "as soon as Pat got home," "to speak to an operator," and "as of June twenty-third" are in the nonfinal position. There is more to follow to complete the thought, and this is signaled intonationally via a contour that does not fall as low as in the case of final phrases. Because this pattern ends at midlevel, %2, let's refer to it as **contour 2**.

Contour 2 can sometimes be used as a nonfinal building block for longer contour 1 messages. See, for example, the concatenation plan for sentences such as "Transferring one hundred dollars from checking to savings" in Figure 11-4.

Figure 11-4. You can use contour 2 in association with contour 1 messages.

Ideally, this utterance would constitute a single breath group, ending at %1. To facilitate recording and concatenation, however, it may be helpful to think of Figure 11-4 as a linking of two smaller constituents: contour 2 plus contour 1. In this example, the concatenated phrase "Transferring | X dollars" could be spoken as a contour 2 pattern, whereas the phrase "from savings to checking" must be a contour 1 pattern.

Later in this chapter, you will see that contour 2 is especially relevant for the concatenation of phone numbers, which begin with a kind of subordinate clause (in the case of seven-digit numbers) or with two such clauses (in the case of ten-digit numbers).

11.4.2 Contours in Context

Now we turn to the use of the basic contours in context. This discussion is particularly relevant to the concatenation of messages.

Lists

Lists are ubiquitous in speech applications, particularly when you offer the user numerous choices, as in menu and help prompts. Depending on the nature of the application, it is sometimes desirable or even necessary to concatenate lists when the choices vary or when the length of the list itself is variable, as is often the case with dynamically generated lists.

With proper planning, it is easy to make concatenated lists sound natural. This is because list items are often surrounded by a slight pause in naturally occurring discourse, especially when the delivery is slow or emphatic. Because these concatenation units are cushioned by silence, less attention is drawn to the undesirable splicing effect that generally marks concatenation junctures.

Figure 11-5 depicts default list intonation, although other list patterns are also possible.

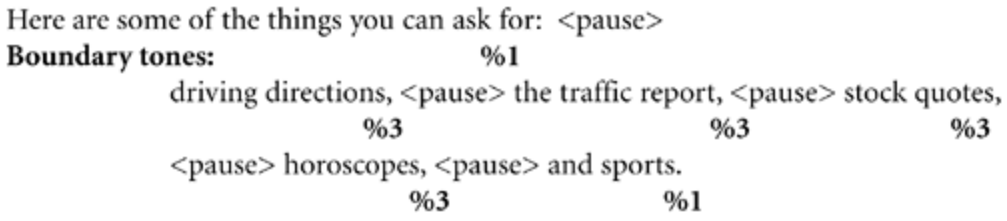
Figure 11-5. This sentence uses default list intonation.

Lists consist of a series of contour 3 phrases, ending with contour 1.[4]

[4] There are other, less common intonation patterns for lists. See, for example, Quirk and Greenbaum (1973).

Figure 11-6 shows the concatenation plan for an error prompt that provides the caller with a list of choices.

Figure 11-6. This concatenation plan helps listeners comprehend a list of choices.



Note that "and sports," as well as "and rye," are best treated as a single concatenation unit. The reason is that in natural speech, this phrase is phonetically continuous, with no break between the conjunction ("and,") and the final option. Notice also the use of pauses. These correspond with naturally occurring breaks and ensure a more natural result despite concatenation.

Yes/No Questions

Questions that expect a response of either yes or no are called **yes/no questions**. In North American English, these sentences generally conform to the basic rising pattern. Concatenation items that fall at the end of such questions must therefore conform to contour 3, ending in a high tone.

The example in Figure 11-7 show the concatenation plan for a prompt confirming datesfor example, "Did you say Saturday, January first?" or "Did you say Monday, May twenty-second?"

Figure 11-7. This concatenation plan covers a prompt that confirms dates.

Did you say DAY_OF_WEEK, MONTH ORDINAL?
Boundary tones: %3

The concatenated result should convey contour 3, that is, a steady rise from level 2 to 3. In addition, phrasing should reflect natural speech. When we say a date such as "Wednesday, June seventh," it is possible to pause between the day of the week and the rest, as indicated by the comma. Observing this pause during recording will facilitate natural-sounding concatenation, because the pause will coincide with the concatenation splice. In contrast, because the month and the ordinal constitute a continuous, fluid stream in natural speech, the voice actor should leave only the slightest of pauses between the month and the ordinalonly enough for the sound engineer to isolate the desired wave files, but not so much as to suggest an explicit break or sense of separation.

If it is feasible, you can avoid concatenation altogether by recording all necessary yes/no possibilities for a given context, as in the prompt set in (17).

(17)

Did you want the home phone?

Did you want the work number?

Did you want the cell phone number?

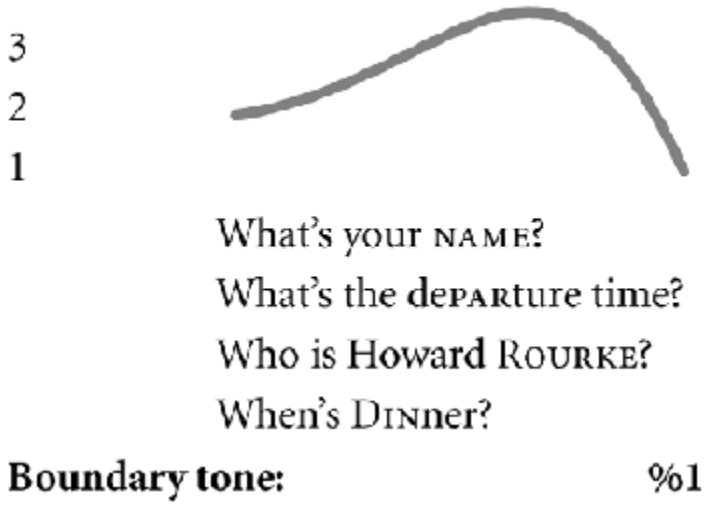
There is no need to concatenate such simple questions as these. Nonconcatenated prompts will sound better.

Wh- Questions

Wh- questions are questions that start with *who, what, where, when, why*, and *how*. They are also called **information questions**, because their function is to elicit information, in contrast to yes/no questions, whose function is to elicit either an affirmation, agreement, or acceptance, on the one hand, or denial, dissent, and rejection on the other.

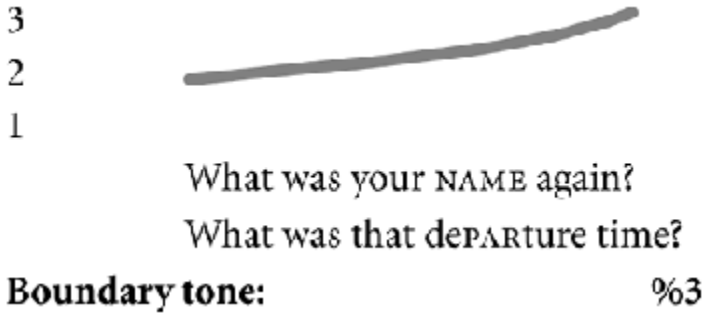
Wh- questions have two possible intonation structures, depending on the underlying intent. Usually, Wh- questions function as first-time requests for information. Figure 11-8 depicts the basic pattern for Wh- questions.

Figure 11-8. The basic intonation pattern of Wh- questions is contour 1.



The default intonation structure of Wh- questions is contour 1. There is, however, another possible use of Wh-questions, which is to request repetition or clarification, as when we hear "What's your name?" or "What was your name again?" (Figure 11-9).

Figure 11-9. The default intonation structure of Wh- questions that ask for repetition or clarification is contour 3.



The intonation of requests for repetition or clarification is contour 3, the same as for yes/no questions.

The prosodic distinction between Figure 11-8 and Figure 11-9 is important for recording prompts and messages, regardless of concatenation. The function of most error prompts is to request repetition. The voice actor should therefore inflect such requests as in Figure 11-9, with contour 3, rather than as in Figure 11-8, with contour 1. Of course, if the voice actor is reading a list of prompts bereft of context and if direction is inadequate, there is little chance of capturing the appropriate prosody.

As you have seen, when contrastive stress is used appropriately, the use of suitable intonation contours also gives the impression that the system possesses humanlike intelligence and is attentive to the progress of the dialog. Consider example (18).

(18)

CALLER: Get a quote.

SYSTEM: Who do you want a quote for?

CALLER: [utterance not recognized]

SYSTEM: Sorry, who do you want a quote for?

The first prompt should be contour 1 (falling), and the second prompt contour 3 (rising). In this way, the system's intonational behavior will comply with callers' expectations based on their experience with authentic conversations. If, however, the first prompt is recorded with contour 3, then it will appear to the caller that the system misrecognized "Get a quote" as a request for a quote on a specific company but that it failed to recognize the company name, as if the caller had said "Get me a quote for Blah Blah Blah Incorporated." The caller would be justified in responding, "Hold on! I haven't said *for who* yet!" The appropriate use of intonation is crucial not only for instantiating a persona that possesses linguistic intelligence and attentiveness to the caller's needs, but also for making the interface comprehensible and usable.

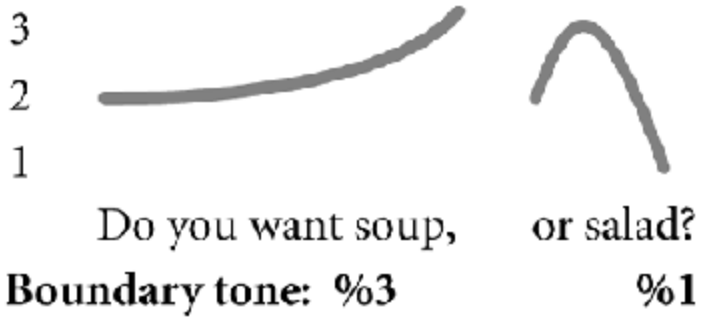
Before we move on to the next intonation pattern, let's briefly evaluate a popular but wrong idea about questions in English. It can be summed up as follows: "Whenever you see a question mark, your voice should go up." Some English teachers dispense this advice to nonnative speakers, and there are even text-to-speech engines that have been designed to uniformly impose a rising intonation contour on all sentences ending in a question mark, including Wh- questions. As you have seen in this section, however, first-time requests for Wh- information do not "go up" without coming back down, and down low. The next section describes another type of question that also ends on a low tone.

Either/Or Questions

Either/or questions aim to get the listener to choose one of two or more explicitly stated options rather than provide a simple yes or no. Because they function as information questions, they end on a low tone (%1), in accordance with first-time information requests, described earlier. Figure 11-10 shows an example of an either/or question.[5]

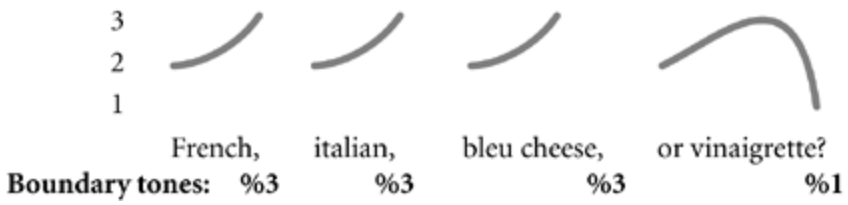
[5] Of course, it is also possible to use a continuously rising pattern, but then the question in Figure 11-10 becomes a yes/no question. The answer would be "Yeah, soup" or "Yes please, salad."

Figure 11-10. An either/or question ends on a low tone.



Questions with more than two options, as in Figure 11-11, follow the intonation pattern typical of lists.

Figure 11-11. Multiple-choice questions follow the intonation pattern of lists.



The intonation patterns of simple as well as complex either/or questions can be generalized as follows: The last list item must fall to a low tone (contour 1), whereas the others must rise to a high tone (contour 3).

As in the case of lists, "or" plus the final option should be recorded as a continuous chunk, without the intrusion of a concatenation splice.

This pattern has other uses, such as reporting a pair of temperatures in a weather report (19).

(19)

. . . with a high of twenty-TWO, and a low of thirTEEN.

11.5 Concatenating Phone Numbers

When we consider the prosodic structure of phone numbers in natural discourse, we can imagine various ways that they might be concatenated. For example, phone numbers can be concatenated one digit at a time (the **digit-by-digit** method) or concatenated by groups of digits. The digit-by-digit method is by far the most common, because it requires the fewest number of recordings. In contrast, the grouping method requires more than a thousand recordings, but it can yield results that approach very natural-sounding speech.

Either way, good-sounding concatenation requires a careful plan, to be executed by a team consisting of a director, a sound engineer, and a voice actor. Regardless of the method, attention to boundary tones and pauses is essential for comprehensibility, as well as likeability of the persona who delivers them.

11.5.1 The Prosodic Structure of Phone Numbers

In natural speech, phone numbers are often spoken as sentences consisting of two clauses (for seven-digit numbers) or three clauses (for ten-digit numbers). These cases are depicted in Figures 11-12 and 11-13, respectively. In both cases, the last group is contour 1, and any preceding clauses are contour 2.

Figure 11-12. Seven-digit phone numbers are spoken as a sentence with two clauses.

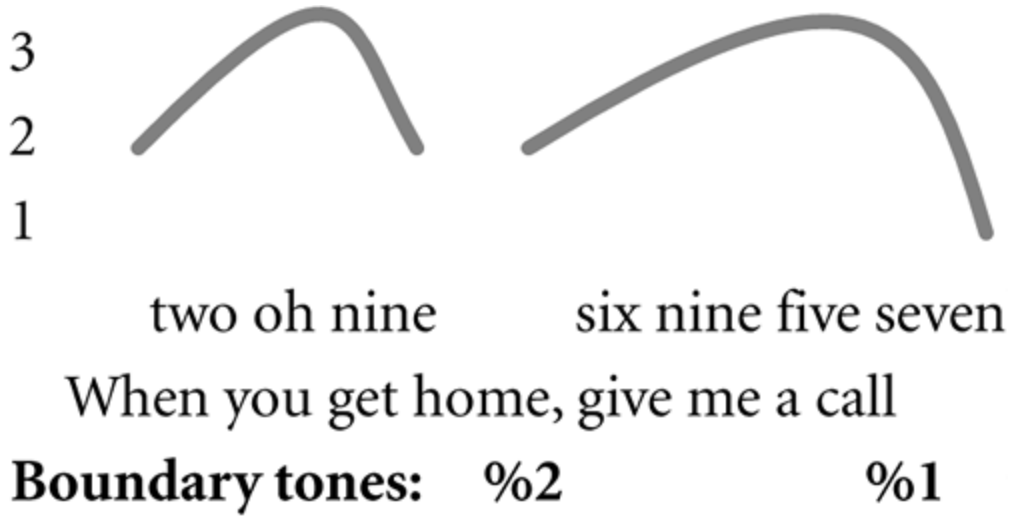
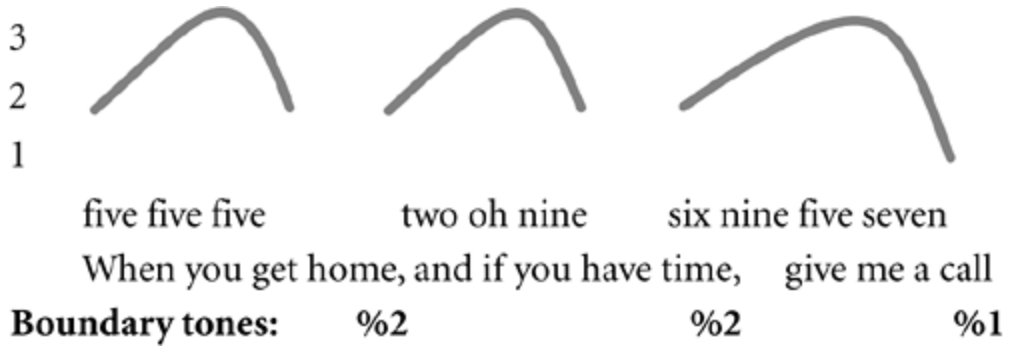


Figure 11-13. Ten-digit phone numbers are spoken as a sentence with three clauses.



Taking the natural prosody of phone numbers as a departure point, we will describe two strategies for concatenation: digit-by-digit concatenation and concatenation by digit groups. Either way, it is critical that

you attend to boundary tones, because they are a fundamental element of the prosodic grammar. In the representation in Figure 11-13, for example, the first and second groups are congruent, both exhibiting a type 2 contour, and the last group describes a type 1 contour.

11.5.2 Concatenation Digit-by-Digit

Concatenation digit-by-digit sounds far from perfectly natural, given the inevitable splices that separate one digit from the next, thereby creating phonetically anomalous sound sequences. However, you can build phone numbers with three sets of digits zero through nine so that at least you satisfy users' basic prosodic expectations. By exerting control over the boundary tones, tempo, and pauses of the concatenated phone numbers, you comply with these basic expectations and thus ensure the comprehensibility of your concatenated message.

You can concatenate phone numbers by using three recorded versions of the digits zero through nine, as presented in Figure 11-14.[6]

[6] For the second of the last four digits, instead of contour 3 you can use a neutral, flatter alternative, as long as it is "high."

Figure 11-14. This plan for concatenating an example ten-digit phone number relies on three recorded versions of the digits zero through nine.

Example digit	Contour
five	3
five	3
five	2
Pause, approx. 200 milliseconds	
two	3
oh	3
nine	2
Pause, approx. 200 milliseconds	
six	3
nine	2 (or 3)
five	3
seven	1

To supply the boundary tones that listeners will naturally expect, the last digit of the phone number must fall to a low tone of 1 (assuming that the phone number is at the end of a declarative sentence), and the last digit of the area code and the last digit of the prefix must end up at the midrange

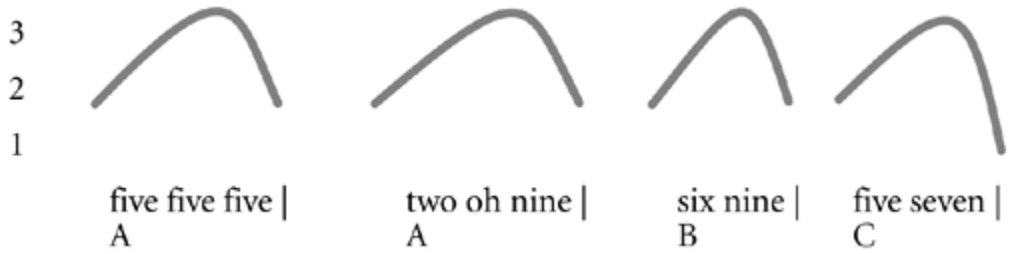
tone of 2, as in contour 2. The remaining digits will be contour 3, as in list intonation, or a flatter, more neutral, but high alternative. In the recording studio, these recordings can be elicited in the context of actual phone numbers. Finally, we insert pauses between the digit groups for proper phrasing.

Again, the two most important considerations for facilitating the comprehension of concatenated phone numbers are pauses and boundary tones. Together, these elements give listeners an implicit sense of exactly where they are in the sequence, as well as what to expect next. The same concerns hold for the concatenation of phone numbers by groups, the topic of the next section.

11.5.3 Concatenation by Groups

For very natural-sounding results but at the cost of having to create a great many more sound files, you can concatenate seven- or ten-digit phone numbers according to the example in Figure 11-15.

Figure 11-15. Concatenation by groups results in natural-sounding phone numbers.



Here, group A refers to three-digit chunks recorded with contour 2. In other words, both the area code and the prefix should be drawn from the same directory of audio files because they both consist of three digits and conform to contour 2. Group B refers to two-digit chunks of contour 2. Group C also refers to two-digit chunks, but of contour 1. Together, B and C are best recorded as a four-digit unit consisting of two subgroups (e.g., "one three" and "six seven"). There should be a slight pause in the middle, sufficient for the sound editor to make a clean cut.

Table 11-2. Sound Files Required for Group Concatenation

GROUP	REQUIRED WAVE FILES	DESCRIPTION
A	000, 001, 002, . . . 999	Triplet, nonfinal; for area codes and prefixes

GROUP	REQUIRED WAVE FILES	DESCRIPTION
B	00, 01, 02, . . . 99	Pair, nonfinal; for beginning of subscriber number
C	00, 01, 02, . . . 99	Pair, final; for end of subscriber number

An alternative but perhaps more difficult strategy is for groups B and C to form a single type 1 contour. In this case, B must rise from 2 to 3, and C must fall from 3 to 1. Again, group C should fall only to %1 if the phone number comes at the end of an assertive utterance.

This approach to concatenation thus requires the inventory of sound files shown in Table 11-2.

Thus, the total number of recordings needed for this strategy is 1,200. This number may be too time-consuming and unwieldy for constrained development situations, but given the time and resources, you can get amazingly natural results. As in all concatenation projects, natural-sounding results depend on recordings that match in volume, speed, timbre, energy level, and personality.

What makes this particular approach sound convincingly real is that the concatenation breaks coincide exactly with the points at which speakers might naturally insert pauses. Because of this alignment, there are no undesirable concatenation splices to disrupt the phonetic flow. Avoiding such splices is the topic of the next section.

11.6 Minimizing Concatenation Splices

Because concatenation most often relies on splicing together what would otherwise be a smooth phonetic stream, it often creates a sense of phonetic disruption, irregularity, and artificialitya problem that is separate from the problem of intonation. It is therefore important to plan concatenation breaks wisely and to be mindful of the phonetic flow of natural speech. In some cases, as you will see, certain concatenation breaks can and should be avoided altogether.

Example (20) illustrates a scenario in which a poorly devised concatenation plan calls attention to the unnaturalness of concatenation. (Recall that a vertical bar indicates a new sound file.)

(20)

This is the | [first second third . . . thirtieth] | saved message.

The plan in (20) reflects a popular but undesirable method of concatenation. The plan is as follows: "Just switch out the word that varies, and nothing more." This kind of thinking is

rooted in a concern for economy and efficiency, and that is fine; but the pronunciation of "the" depends on the word that follows it, at least in Standard English. Before a word starting in a consonant, such as "seventh," we pronounce "the" with a **schwa**, the reduced-energy, centrally placed vowel in English ("uh"). Before words that start with a vowel, such as "eighth," "eleventh," and "eighteenth," however, the vowel in "the" is pronounced tensely, as in "thee." Notice that in Standard English these words also require the article "an," rather than "a"for example, "an eighth."

Furthermore, for technical reasons that we cannot delve into here, the articles "the," "a," and "an" form a single **phonological word** with whatever word follows. Planning a concatenation break between "the" (or "an") and "eighth" would be phonologically similar to planning a break between "Se" and "attle" in "Seattle."

A superior concatenation unit would consist of "the" plus the ordinalfor example, "the seventh," "the eighth," "the ninth," and so on. Better still, the first concatenation unit could consist of bigger chunks such as, "This is the seventh," "This is the eighth," "This is the ninth," and so on. In addition to respecting the phonological wordfor example, "the eighth"this plan is also desirable because the fewer the concatenation units, the more natural the result.

The same mistake is made in the concatenation plan of a touchtone prompt of an airline application, as illustrated in (21). The goal here was to recycle the chunk "Please enter the."

(21)

Please enter the | hour, one to twelve, followed by the star key.

Please enter the | minutes, one to fifty-nine, followed by the star key.

The break planned between "the" and what follows is in the middle of a phonological word. Furthermore, because "hour" and "minute" require distinct pronunciations of "the," whichever way "the" is recorded is sure to be the wrong pronunciation in one of the two cases. As in (20), a better strategy would be to record "the hour" and "the minutes" each as a unit. Better yet, do not concatenate; instead, record the sentences whole.

Earlier in this chapter, you saw several concatenation plans in which concatenation junctures align with pauses that would occur in natural speechfor example, lists and phone number groupings. By exploiting the naturally occurring pauses that occur in speech, you avoid the physiologically impossible splice effects that frequently mark concatenation junctures and mar the naturalness of prompts. The concatenation break is less perceptible, and the overall result is smoother-sounding. To this end, we now demonstrate how you can make minor changes in wording to produce concatenation that sounds more natural.

The first example is drawn from the design of a weather application in a voice portal. The wordings of (22) and (23) are nearly identical, but (22) will be easier to concatenate for more natural-sounding results.

(22)

Today, | it'll be partly cloudy, | with a high of | 67 | and a low of | 51.

(23)

Today | will be partly cloudy, | with a high of | 67 | and a low of | 51.

The difference is that the grammatical function of "today" in (22) permits a following pause, whereas in (23) a pause is prohibited. The concatenation splice between "today" and "will" in (23) interrupts what would be a continuous, uninterrupted stream in natural speech.

Mindful of how naturally occurring pauses can mask concatenation splices, consider the error prompt in (24).

(24)

Is 555 593 1367 the number you want me to call?

No matter how the phone number is read back, there are at least two concatenation splices, one between "is" and the first digit of the phone number, and the other between the last digit and "the number . . . ." But in conversation, these are not natural points to introduce an interruption of any sort. The entire question in (24) constitutes a single breath group, describing an intonation contour type 3. It can be rewritten, however, so that the concatenation breaks coincide naturally with breath group boundaries, as in (25).

(25)

This is the number I heard: 555 847 7037. Did I get that right?

In (25), the phone number is naturally cushioned by silence at either edge. Another wording that obliges a pause to precede the phone number is, "Confirming: . . . ." If the phone number is concatenated by the grouping strategy, listeners probably won't notice that the prompt was spliced together.

It would be difficult, perhaps impossible, to make the concatenation strategy in (24) sound natural, for two reasons. First, splices before and after the phone number will be obvious and phonetically disruptive. Second, the phone number in (24) does not end on any particular boundary tone, because it occurs in the middle of a single, overarching type 3 contour, at least in natural speech. In contrast, (25) naturally comprises three prosodic "islands," each with its own distinct contour:

"This is the number I heard"	*contour 1*
PHONE_NUMBER	*contour 1*
"Did I get that right?"	*contour 3*

The only real concatenation challenge in (25) is the handling of the phone number. The plan in (25) is thus easier to execute than that of (24) and virtually guarantees smooth results.

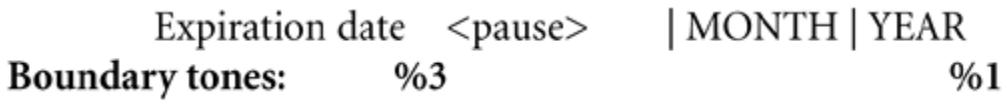
The next example shows that less (in wording) is sometimes more (in naturalness). Let's first examine the concatenation strategy we recommend. The context in (26) is the read-back of credit card information.

(26)

Expiration date: October, oh-one.

This is easy to concatenate for natural-sounding results. The recommended concatenation plan is presented in Figure 11-16.

Figure 11-16. Concatenation plan for reading back credit card information.



That is, "expiration date" will be delivered with contour 3, followed by a pause lasting about a quarter of a second, followed by two concatenation units that together express contour 1. This is a natural prosodic pattern that is perfectly suited to the reading aloud of items on a formfor example, "Ethnicity: Pacific Islander" on a census form.

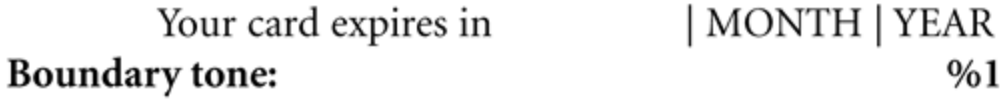
An alternative is the wording in (27).

(27)

Your card expires in October, oh one.

In natural speech, (27) represents a single breath group, but to concatenate this message, three concatenation units must be spliced together, as shown in Figure 11-17.

Figure 11-17. An alternative concatenation plan for reading back credit card information.



Concatenating three units so that they suggest a single breath group is more likely to yield an unnatural-sounding result than concatenating three units to suggest two breath groups. Recall that in Figure 11-16 only the MONTH and YEAR are subsumed by contour 1. Furthermore, a natural rendering of (27) would require that the *n* of "in" serve as the phonetic onset to the first syllable of "October," thus yielding the syllable "noc" ("in October"). Yet it is in the very midst of this prosodic atom, or syllable, that a concatenation splice has been planned. If the goal is a smooth and natural sound, Figure 11-16 is a safer bet than Figure 11-17.

This credit card example demonstrates that a brief, telegraphic recording, which is acceptable in this context, can offer fewer prosodic risks and can more robustly suggest natural prosody under concatenation than a chattier, prompt-only-in-complete-sentences approach.

11.7 Pauses

Just as the white areas on this page enable you to discriminate letters and words and sentences, brief pauses play an important role in the perception of speech and language. If your goal is to create a natural-sounding, familiar interaction for your users, you must consider the placement and duration of pauses.

Consider the error or help prompt exemplified in (28). Assume that the prompt must be concatenated because the items in the list are generated dynamically.

(28)

Here's what you can say: | "mailbox," | "address book," | "sports," | "news," | or "weather."

The syntax of (28) requires a pause, as indicated by the colon, following "say." Additional pauses, as indicated by the commas separating the list items, will facilitate naturalness and comprehensibility. In other words, there is a pause at each concatenation break. The duration of these pauses varies according to a number of factors, including the rate of the speaker's delivery and the use of the utterance in context. A good first try might be 400 milliseconds (0.4 seconds) for the colon and 200 milliseconds (0.2 seconds) for each comma, and then

increase or decrease from there, letting your ear be your guide.

Example (29) illustrates the case of a recording ending in a period, or full stop, preceding another recording.

(29)

Sorry, I still didn't get that. Tell me your PIN one more time.

Depending again on rate of delivery as well as the desired effect, a period can vary considerably in duration, although it is generally longer than any commas in the vicinity. Here, for example, there's a comma between "Sorry" and "I." A good first try might be in the range of 300 to 400 milliseconds. This pause should probably be adjusted in the code rather than in the actual audio file so that it can easily be tested and adjusted later.

Here are some rough guidelines to ensure proper rhythm. Insert silence in the following cases:

- After nonrecognition messages such as, "Sorry, I didn't catch that," "Let's try this a different way," or "OK, let's go back to the main menu" (assuming of course that these are followed by another recording)

- After an audio file when it is desirable to suggest a comma, a colon, or a long dash at the endfor example, "Here's the number I heard: . . ." or "This is the account number I've got on file for you: . . ."

- After a recording that is followed by text-to-speech (to ensure a less jarring hand-off of voices)for example, "The street address is . . . [TTS]

- Before a prompt if it is preceded by TTS (again, for a less jarring hand-off)

Do not insert silence in these cases:

- Until you are sure you have identified all the places where the silence will occur in the call flow and are confident that the result will be appropriate

- At the end of one file and again at the beginning of another file if there exists a point in the call flow where these two files abut

Many of these guidelines come together in example (30), which consists of a sample e-mail header.

(30)

Here's your first new message. Friday, 9:45 a.m. Subject: [performance reviews]. Message from: [Mitzi Dalton Huntley].

Following is a concatenation plan to realize this example. We are using the dollar sign ($) to indicate "milliseconds of silence" and [square brackets] to indicate TTS recordings. Notice that careful punctuation in the header systematically corresponds to different durations of silence in playback.

Concatenation Plan

"Here's your first new message" + $400

"Friday" + $200

"nine" + "forty-five" + "a.m." + $400

"Subject:" + $200

[TTS]

$400 + "Message from:" + $200

[TTS]

Note that silence should not be inserted between the hour and the minutes. The colon used in times of day ("9:45") is one of the more unusual examples of punctuation that serves a purely graphical rather than prosodic function. Nor should any silence be inserted between the minutes and a.m. or p.m.

Pauses can also communicate other kinds of higher-level information. Consider the messages in (31), (32), and (33).

(31)

Sure, sports! <pause> Oh, looks like that's not available at the moment.

(32)

Hold on while I get your reservation. <pause> Sorry, but there seems to be a technical problem.

(33)

Just a moment while I look that up. <pause> Actually, I'm going to have to transfer you to a customer service representative because there are more than nine people in your party.

In each case, if the second sentence immediately followed the first, without a sufficiently long pause, these messages would sound absurd. These silences are important because they help the listener reconcile the superficially incompatible relationship between the two sentences. The system response in (31) starts with the implication that the sports option is available. The content of the second sentence, however, contradicts the initial assumption. The function of the pause plus "oh" is to communicate a reasonable change in the speaker's knowledgein this case, concerning the availability of sports information. This creates the sense that the VUI's persona has only just now found out about the unavailability of the sports option and should therefore not be held at fault.

Such pauses are important because they comply with our expectations of the amount of time that elapses when people retrieve information for others in real life. Furthermore, phrases such as "hold on" and "just a moment," as in (32) and (33), induce the caller to take a temporary mental break. Without a sufficiently long pause following such phrases, the caller will be caught off-guard by the next utterance.

The pauses indicated in these examples should be dependent on the sentences that follow them so that they do not add to any latency inherent in the system. The duration of the pause should be relatively long; otherwise, the sequence will seem implausible. Depending on the general tempo of delivery, try one, one and a half, or two seconds. Again, the duration of such pauses should be adjusted in the code, for ease of testing and later adjustability.

11.8 TTS Guidelines

The epitome of science fiction is a human having an intelligent conversation with a computer. The image is commonplace in movies and television as humans talk to the holodeck on the *Enterprise*, to the android C3PO, and to Hal, the definitive supercomputer. What makes these interactions notable is the computer's apparently effortless understanding of human speech, as well as the clear, intelligible, natural, humanlike quality of the computer's speech.

The availability of high-quality text-to-speech (TTS) engines brings us ever closer to the worlds of our imagination, where computers speak like humans. As the process of generating natural-sounding spoken language from text becomes increasingly automated, we can expect to find many new practical applications of the technology to everyday life.

In the world of voice user interfaces, TTS offers a number of conveniences. Because there is no need for prerecorded audio files, implementation is relatively easy. Developers can produce prompts quickly and edit them easily. More importantly, the content of TTS messages can be spontaneous and can be changed dynamically. In contrast, applications that rely on prerecorded audio files alone are best suited for conveying information that is static. When the information changes often or needs editing, the voice talent must be called in to record either whole utterances or concatenation units that will in turn form whole utterances. But no amount of this kind of prerecording can handle situations in which information cannot be predicted or constrained, as in the case of e-mail playback. TTS renders these problems moot because it lets you easily

integrate spontaneous and dynamic texts, such as e-mail and late-breaking news stories, into speech applications.

Despite these conveniences, you should consider certain human factors issues when using TTS in your VUI. Even though the quality of TTS has greatly improved over the past ten years, users are still very much aware that TTS is not human speech, and many if not most users prefer recordings of human speech. One area of dissatisfaction is simply that TTS output is more difficult to understand. This is not surprising. As you have seen in this chapter, the prosody of messages is highly dependent on contextual cues. When a native speaker reads aloud an e-mail or news story, he or she gathers contextual cues over the course of the text, which can be lengthy. So in real life, the prosody of a particular sentence is often determined by information that was presented several sentences earlier. The current state of TTS technology does not acknowledge these kinds of contextual cues, let alone gather them and in turn realize them in natural-sounding prosody. The added cognitive burden required for comprehending TTS is especially problematic for nonnative speakers and elderly users.

Some of these issues, however, are mitigated to the extent that listeners seem to be able to adapt to the sound of TTS over time. There is evidence that repeated exposure to TTS improves comprehension. Presumably, listeners learn to "turn off" their implicit expectations for phonetic and prosodic naturalness. However, if you use a high-quality TTS engine and if the content of your output is easythat is, predictable and constrainedthen many comprehension difficulties can be avoided.

When you're deciding how to integrate TTS into an application, consider the following guidelines to help optimize usability.

11.8.1 Analyze Application Usage

Estimate how often and in what environments callers will be using the application. If most callers are one-time users in noisy environments, TTS will be more difficult to comprehend. Also, if callers are performing another task, such as driving, while listening to the system, then TTS will impose more cognitive demands on the caller and serve as a distraction.

Ideally, you should use TTS in applications that will be accessed by repeat callers in quiet, undemanding environments. Repeated exposure to the system will give callers the opportunity to become accustomed to the voice, something that has been shown to facilitate comprehension.

11.8.2 Choose an Appropriate Voice

Consider the target audience and type of application when you choose the gender of the TTS voice. The voice itself will affect users' impression of the system. Unsurprisingly, it has been found that the gender of TTS voices elicits cultural connotations, just as human voices do.

Also consider how the TTS voice will sound next to the voice of your prerecorded audio files. Actually, you may want to exploit the distinction between the two so that any negative impressions of TTS, which are likely beyond your control, do not "contaminate" users' favorable evaluation of the application's principal persona, over which you should have total control. For example, if the persona of a voice portal that reads e-mail and news is a woman, then the TTS rendering of headers and content should perhaps be in a man's voice. Unless you are using a very high-quality,

sophisticated TTS engine to read items whose form is predictable and constrained, do not attempt to pass off TTS as natural, persona-rich recordings.

11.8.3 When Possible, Use Audio Recordings

Because human speech is generally preferred over TTS, you should use audio recordings of a professional voice actor whenever possible. In some cases a single sentence may have sections that are dynamic, whereas other parts are always static, as in (34) and (35).

(34)

First message: | "meeting time."

(35)

The street address is: | 1313 Mockingbird Lane.

The question in these cases is whether the consistency of TTS would be better than combining TTS with recorded speech. Research at the University of Edinburgh (CCIR-5 1999) and British Telecom has shown that users prefer prompts that use both TTS and recorded speech rather than TTS for the entire prompt. So in these examples, "First message" and "The street address is" would be recorded by a voice actor, and "meeting time" and "1313 Mockingbird Lane" would be in TTS.

Note that the use of the colon in (34) and (35) is intentional. The voice actor should read the text so as to mindfully "announce" the entry of a new breath group delivered in a different voice, that is, the voice of TTS. The first part of both (34) and (35) should be read with a slight sense of suspension, and following the colon there should be a brief pause, as recommended here. Without the colon, we have found that professional voice talents are inclined to deliver these sentence fragments as concatenation units that wrongly suggest only the first part of a breath group and only the first part of an intonation contour. When these prosodically incomplete recordings are concatenated with their TTS complements, the result is jarring. (It is as if two voices are somehow working off one set of lungs!)

As always, set the context for your voice actors. Let them know that they are introducing TTS.

11.8.4 Make Content Easy to Understand

As stated earlier, TTS allows you to include information in your application that could not possibly have been recorded in the studioin other words, content you have little or no control over. When you do have control, however, try to use

simple vocabulary and grammar. The simpler the text, the greater the intelligibility and usability of the application. Text should provide ample context to ground the information to concepts that are generally known to most users. Repeating important ideas will also help users retain them.

If the content appears somewhat difficult, try slowing the TTS speaking rate. A comfortable rate for most listeners is between 150 and 200 words per minute.

11.8.5 Use Appropriate Formats

Often, we take for granted the way certain types of information are expressed. For example, the number 1313 in example (35) should be read "thirteen thirteen" instead of "one thousand three hundred thirteen." Similarly, the zip code "94536" should be read as "nine four five three six," never "ninety-four thousand, five hundred thirty-six."

Make sure that all abbreviations, too, are read back in the appropriate format. For example, "St. Andrews St." should be read back as "Saint Andrews Street"; otherwise, it will likely be unintelligible. Aberrant delivery will distract the listener and divert attention from the surrounding content.

11.8.6 Mark Up Text for Naturalness

There are also strategies for **marking up** the text to be delivered by TTS in order to get the most natural results possible. For example, some researchers have shown that inserting pauses between sentences and between major phrases and adding breath intake sounds before sentences can improve the naturalness of the text being presented

and make it easier to remember. In addition to the use of pauses to facilitate intelligibility, there are prosodic mark-up strategies so that TTS will simulate natural, humanlike stress and intonation patterns, as described throughout this chapter. You can also improve pronunciation by adding phonetic spellings to the TTS dictionary for words it consistently gets wrong. For an overview of methods for optimizing the quality of TTS for your application, see Ishihara (2003).

Human factors issues become important at many levels when you're thinking about how to integrate TTSfrom the high-level goals of your application to the details of number formats. Carefully considering all these issues will help make TTS an important part of a human-centered, user-friendly interface. As the technology that enables computers to understand and speak becomes more widely available and gains acceptance, the possibilities of how and where it can be used are almost endless.

11.9 Conclusion

Just as we would never write a prompt such as, "Please to say now you PIN," which is ungrammatical syntactically, we should never produce recordings that are ungrammatical prosodically. Recorded messages are prosodically ungrammatical when, for example, they exhibit inappropriate or alien intonation contours, inappropriate or missing pauses, stress and rhythmic patterns not found in human-to-human discourse, and physiologically impossible sound sequences. To avoid these problems, design choices need to reflect the systematic and context-sensitive structure of language at the level of prosody.

Every spoken natural language has its own highly structured grammar of prosody, and so does every language variety, such as African American Vernacular English (popularly known as Ebonics), Moroccan Arabic, and Sicilian. Prosody is an essential component of the way humans structure their intent and mediate interpretations in context. There is no reason to believe that people suspend prosodic assumptions and expectations when interacting with a speech application. After all, prosody is an integral aspect of spoken language, as essential to our system of verbal communication as syntax or semantics. When you record and implement messages, you should respect familiar, natural patterns of stress, intonation, and phrasing, mirroring authentic language use as much as the limits of technology allow.

Chapter 12. Maximizing Efficiency and Clarity

There is substantial evidence, from surveys and other forms of user feedback, that callers care a great deal about efficiency. **Efficiency** is the ability to complete transactions quickly without wasted steps or wasted time.

Given the serial nature of speech, efficient design of VUIs can be challenging. Consider a system that, in response to a search request, reads back a list of movies playing at local theaters. You can quickly scan such a list visually on a screen, focusing on the columns and items of interest, but listening to spoken playback of a long list of such items can be tedious. Design of an efficient spoken version may require flexible approaches that allow the caller to navigate the list quickly by various criteria, such as "playing after 7 p.m." or "playing in Palo Alto."

On the other hand, VUIs can take advantage of callers' spoken language capabilities in order to increase efficiency. People are adept at structuring complex sentences that package lots of information (e.g., "I wanna go from San Francisco to Boston next Tuesday, arriving by nine p.m."). Provided that the technology can handle such complex input with high accuracy, callers can often express their needs more efficiently with speech than with keyboard and mouse.

Clarity is a caller's understanding, at each point in an interaction, of what exactly is happening, which options are currently available, and how to accomplish his or her goals. Clarity is a product of many design elements, from the wording of individual prompts to the structuring of dialog strategies. Although some of the new natural language

understanding technologies provide a great opportunity to better meet callers' needs, they also present new challenges in making the system's capabilities clear to the caller and easy to exploit.

Sometimes designers must make trade-offs between efficiency and clarity. Especially when systems are used repeatedly by the same callers, there may be significant differences between the needs of first-time callers and experienced callers. A number of techniques can be used to accommodate both types of users.

This chapter covers approaches for maximizing efficiency, maximizing clarity, and managing the trade-off between efficiency and clarity when necessary. We also discuss creating appropriate mental models for users of advanced natural language understanding systems.

12.1 Efficiency

Callers' perceptions of the efficiency of an interface reflect a number of system characteristics. Clock time, although it has an effect, is less salient than a number of other measures. One such measure is the number of interchanges required to get the job done. Another is pacing, measured by the latency between the time a caller finishes speaking and the beginning of the next prompt. If these delays are much longer than what callers have learned to expect in human-to-human conversation, the dialog starts to feel sluggish and inefficient.

The single biggest barrier to the perception of efficiency is repeated requests for the same information. Many deployed systems request information such as name and account number when they transfer the caller to a live agent or to another service, even when that information has already been supplied earlier in the call. Callers find this tremendously frustrating. Such behavior by the system creates an image of the service (and, to some degree, of the deploying company) as unintelligent, inconsiderate, and uncooperative, as well as inefficient.

Poor efficiency can lead to reduced task completion because impatient callers are more likely to hang up. It also results in longer call durations (which increase costs), lowered user satisfaction, and even a reduced level of attention by the caller.

Many of the guidelines for improving the efficiency of interfaces are simple and obvious. But because they are often violated in deployed systems, we briefly review some of them here.

12.1.1 Don't Lose Work

The first rule of thumb is to avoid the problem just cited: Don't ask for the same information a second time after it has been gathered successfully. The reason deployed systems make such repeated requests is that it takes infrastructure (e.g., integration with a CTI system) to communicate information gathered by the application to the screen of a customer service agent. Sometimes it simply does not make economic sense to provide such infrastructure. However, it is important for managers to make that decision with full awareness of the significant impact such a decision will have on caller satisfaction.

There are other reasons callers may lose work. For example, if they are in the middle of specifying a transaction or filling in details of a form (e.g., parameters of a trip they are planning) and they navigate elsewhere in the application (e.g., by going to the main menu or jumping to some other part of the application), they may lose the information already entered.

This latter problem can be avoided. Whenever a command might result in lost work, the caller should be alerted and provided with an opportunity to cancel the command. This same guideline is typically followed by software utilities (such as word processors): When users try to exit a system without saving the latest version of the file they have been editing, they are alerted and offered an opportunity to save the file.

12.1.2 Make Frequent Tasks Efficient

The next rule of thumb is to focus on making the most frequent tasks the most efficient, even at the expense of

making other tasks less efficient. For many applications, the 80/20 rule applies: In short, 20 percent of the functionality is exercised 80 percent of the time. Optimizing frequent tasks may require changes in the structure of the dialog or in prompt wording. You may also have to change the order of presenting menu options or list information.

For example, consider voice-activated dialing (VAD). A VAD system must place calls as well as manage the user's call list (e.g., allowing users to add names to the list of people they can call). Most of the time, the user is placing a call. It's best to accomplish this task directly and simply, even at the expense of adding a step for users to get to the call-list management facility.

12.1.3 Provide Shortcuts

Some systems, such as personal assistants, are designed for repeated use. As callers become familiar with a system, they lose patience for listening to instructions about features they already know how to use. They also prefer to get their work done with fewer steps. You should provide shortcuts to let frequent callers navigate your system quickly and complete their tasks.

To teach callers about shortcuts, you can use hints. For example, if a personal agent system allows dialing from the top-level menu but a caller is taking a less direct route, tell the caller about the more efficient way to perform the operation. The hint might read as in example (1).

(1)

SYSTEM: What would you like to do?

CALLER: Place a call.

SYSTEM: Who would you like to call?

CALLER: Steve Smith.

SYSTEM: Okay, calling Steve Smith. By the way, in the future you can save time by requesting a call directly at the main menu. Just say, "Call," followed by the person's name.

Do not repeat hints very often, and do not provide hints if the caller has ever used the shortcut before.

12.1.4 Use Caller Modeling to Save Steps

Knowledge about a caller may be useful in reducing the number of steps required to complete a task. For example, a travel application that uses caller profiles can use

knowledge about where callers live and places to which they typically travel to reduce the amount of information they must provide when making travel arrangements. Similarly, knowledge obtained earlier in the current call (the **call history**) can be used to make the interaction more efficient later in the call.

The dialog in (2) shows an interaction with a travel planning application in which the system uses both call history and caller profile. The profile includes things provided explicitly by the caller, such as home address, as well as things learned implicitly from previous calls or earlier in this same call.

The caller states her destination city, but no city of origin. Rather than prompt for origin city, the system gets her home address from her profile and assumes that she is traveling from her home airport. The confirmation prompt makes the caller aware that the system has assumed the origin city. If it is wrong, she can change it.

The caller specifies New York as the destination. New York is served by a number of airports. The system assumes Kennedy Airport based on earlier experience with this caller; she often books flights to Kennedy Airport.

The caller requests a rental car. Her preference for midsize cars from Hertz is included in her profile. The system assumes she wants it available soon after landing at Kennedy Airport, basing that assumption on information obtained earlier in the current call. Again, the caller is made aware of assumptions in the confirmation prompt and can make changes at that time.

(2)

SYSTEM: Where are you traveling?

CALLER: I'm going to New York.

SYSTEM: When do you want to leave?

CALLER: Wednesday, March tenth, around one p.m.

SYSTEM: Just to confirmyou want to go from San Francisco International to New York, Kennedy Airport, on Wednesday, March tenth, leaving around one PM. Is that right?

CALLER: Yes.

SYSTEM: Great. Hold on while I check available flights <pause> There's an American Airlines flight leaving SFO at 1:20 p.m. and arriving at JFK at 9:30 p.m..

CALLER: Please book it for me.

SYSTEM: Okay. You're booked on American Airlines Flight 825, leaving SFO at 1:20 p.m. and arriving at JFK at 9:30 p.m. on March tenth. Now, would you like a return flight?

CALLER: No.

SYSTEM: How about a rental car?

CALLER: Yup. I'll need a car.

SYSTEM: I can reserve a midsize car with Hertz for thirty-four dollars a day. Is that okay?

CALLER: Yes.

SYSTEM: Great. I have a midsize car reserved for you with Hertz. You can pick it up at 9:30 p.m. on March tenth, at the Hertz desk at Kennedy Airport.

12.2 Clarity

Clarity is a goal at all levels of design of a voice user interface. At a very high level, this means clarity about such elements as the appropriate mental model for using the system. At an intermediate level, clarity is required about things such as the navigational strategy (how to navigate or access the system's features). At the lowest level, the design needs clarity about such elements as what is **in-grammar** (that is, what can be said) in each individual dialog state.

This section presents two examples to illustrate how thinking about clarity influences design. The first example covers creation of a clear mental model for call routing systems that use advanced natural language technology. The second example covers navigational clarity for voice portals.

12.2.1 Mental Models for Natural Language Understanding

When you design an advanced natural language understanding system, one of your primary challenges is to create in the mind of the caller a clear mental model of how to talk to the system. Directed-dialog systems are easier to design. You can simply present menus ("Do you want red, green, or blue?") or ask directed questions that make the answer set obvious ("How many shares would you like to buy?").

In contrast, when you give callers the flexibility and power of advanced natural language, the problem of describing

the boundaries around what they can say is much greater. The set of valid utterances is too large and diverse to describe explicitly. Still, even the most advanced systems have nowhere near the capabilities of humans. You therefore cannot simply fall back on the mental model afforded by human-to-human conversation. In general, we strongly advocate against misleading callers into thinking they are interacting with a human. That will result in assumptions of capabilities beyond what the system can fulfill.

In this section we discuss the issue of creating a mental model for natural language call routing systems, the class of natural language systems with which we have the greatest amount of data and experience. As discussed in earlier chapters, a natural language call routing system allows callers to connect to one of a wide array of services simply by describing what they want in their own words. Many of the lessons learned from this kind of application generalize to other types of natural language applications.

First, let's look at the mental model we want to create:

- Callers are talking to an automated system (so they shouldn't ramble on).
- Callers should describe their needs in their own words.
- Callers should be brief (approximately one sentence).

To understand these issues, Sheeder and Balogh (2003) ran a number of experiments comparing various approaches to prompting callers in a call routing system. The primary result of the experiments was that an effective mental model could be created by presenting a couple of examples at the beginning of the prompt. The examples should be in

the form of natural sentences rather than keywords. The most useful examples were those "focusing on imparting a sense of the expected *form* of the request, as opposed to an arbitrary semantic category label" (p. 110). Of the four examples tested, the most effective prompt was as follows:

> *Welcome to Clarion Wireless Customer Service. You can ask me things like, "How many minutes have I used?" and "I'd like to set up automatic payments." So, how can I help you with your account?*

This approach resulted in significantly higher task completion and significantly lower misrouting rates than the other approaches tested. Interestingly, it also resulted in lower **disfluency** rates (e.g., hesitations or word fragments). Furthermore, users rated the system with sentence-like examples as easier to use. The experimental results have since been corroborated by in-service data from a deployed system.

In related work, Boyce (1999) showed that using an earcon at the beginning of the welcome prompt was an effective means of communicating to the caller that the system was automated, not human. This technique can be combined with the approach we have outlined for natural language systems.

12.2.2 Navigational Clarity Through Landmarking

Navigational clarity refers to how clear callers are about where they are in an application and how they can go elsewhere. For example, users should know which of the available services or features they are currently accessing

and how to access other services or return to the main menu.

Voice portals, and voice browsing in general, have a greater level of navigational complexity than most other applications. They typically present a wide array of services and information sources. In most applications, callers have the means to transition quickly between services.

A common technique to clarify for callers where they are or where they are going is landmarking. **Landmarks** are auditory indicators that are clearly associated with each particular service. Landmarks can take a number of forms:

- **Verbal:** The system can simply announce the name of the service.

- **Persona:** Different services can be associated with different voices and different personas. To make the association clear, the persona should be carefully matched to the type of service.

- **Nonverbal audio:** Earcons can be designed to indicate different services (although it can be confusing to use more than a small number of earcons). Alternatively, an audio environment, such as environmental sounds or music, can create an ambiance related to the type of service.

One example of a voice portal with a very effective landmarking scheme is Tellme, which offers a variety of services such as stock quotes, news, entertainment directories, and sports scores. The Tellme service uses all three landmarking techniques consistently. When the caller transitions to a new service, it is announced verbally. Each service has its own distinctive persona, created by a

different voice actor, and each service has distinctive background sounds. The result is clarity about what service is being accessed at any particular time, as well as an engaging user experience created by the artistry of the design.

12.3 Balancing Efficiency and Clarity

There are times when efficiency and clarity must be traded off. Many such trade-offs are related to prompting. Sometimes it takes a long-winded prompt to describe clearly what is in-grammar or what is expected at each point in the dialog, but callers who are already familiar with the system will not want to listen. Following are guidelines you can use to accommodate first-time as well as experienced callers.

12.3.1 Stress Clarity in Individual Prompts

It's tempting to carefully craft the wording of prompts to make them as concise as possible. Although you can sometimes achieve great clarity with few words, if you are losing clarity for the sake of shaving off words, you are probably making the wrong trade-off. Keep in mind that lack of clarity wastes significantly more than the time taken by a few extra words in a prompt. Every time a caller must rely on error recovery procedures or help commands to compensate for the confusion caused by a cryptic prompt, significant time is wasted. In addition, caller confidence is lost and caller frustration increased. Sacrificing a well-written, smoothly flowing interchange to save a few words is not worth it.

The Dialogues 2000 project (CCIR-3 1999) examined user preferences for prompts. Researchers had participants interact with voice systems that had different prompt styles: fast-track ("Customer number") and neutral ("Please say your customer number"). The results showed that

participants rated a system with short, fast-track prompts lower than a system with longer, more verbose but more natural prompts. The data also suggested that the fast-track prompt wording became more acceptable if the user was exposed to the verbose prompts first. These findings high light the importance of making prompts for novice users very clear, even if they contain more words.

It is not wrong to craft prompts carefully to be concise. The guideline is to do so without sacrificing clarity and ease of communication.

12.3.2 Taper Prompts

Some prompts may recur numerous times within a single call. For example, the caller may come back to the main menu frequently. In that case, **prompt tapering** is often useful. The idea is to be very clear and descriptive the first time, or first few times, and then shorten the prompt. An example in Chapter 9 uses prompt tapering. We repeat an excerpt here:

(3)

SYSTEM: . . . Now, what's the first company to add to your watch list?

CALLER: Cisco.

SYSTEM: What's the next company name? (Or, you can say, "Finished.")

CALLER: IBM.

SYSTEM: Tell me the next company name, or say, "Finished."

CALLER: Intel.

SYSTEM: Next one?

CALLER: America Online.

SYSTEM: Next?

CALLER: . . .

An alternative to prompt tapering is **delayed help**, which involves following a brief prompt with a pause (of approximately two seconds) followed by more detail. With this approach, experienced callers tend to respond during the pause. If a caller doesn't respond right away, he will

hear more detailed instructions. Delayed help should be tested carefullyit sometimes causes problems with callers who do not know what to say but jump in and begin speaking during the pause.

12.3.3 Use Barge-In

The use of barge-in can significantly improve the trade-off between efficiency and clarity and make it easier for systems to accommodate naive as well as experienced callers. Experienced callers can simply interrupt the prompt as soon as they know what they want to say, and naive callers can listen to the detailed prompts before formulating their response.

As part of a larger study of mixed-initiative dialog systems, Nuance examined barge-in behavior with respect to two things: (1) level of experience with a system and (2) prompt length. The study used the TickerTalk system, a mixed-initiative brokerage application. Approximately 250 callers used the system for three weeks. Users could call as frequently as they wished and could get quotes, make trades, set up watch lists, and administer their accounts. The system was connected to a live quote feed.

Each caller was given an account with $100,000 of fake money. To inject realism into the use of the system, a contest was held in which the user with the highest net worth at the end of three weeks was awarded $500 in real money. Data for more than 3,000 phone calls, including more than 30,000 utterances to the main menu (which was an open-ended prompt), were gathered. The data were used to study numerous aspects of mixed-initiative dialog, including barge-in behavior.

Analysis of the barge-in data showed increased frequency of barge-in as individual callers made more calls to the system (see Figure 12-1), and increased use of barge-in on longer prompts (see Figure 12-2). The results suggest that experienced callers will use barge-in to navigate the application more efficiently.

Figure 12-1. In repeat calls, users barge in more often.

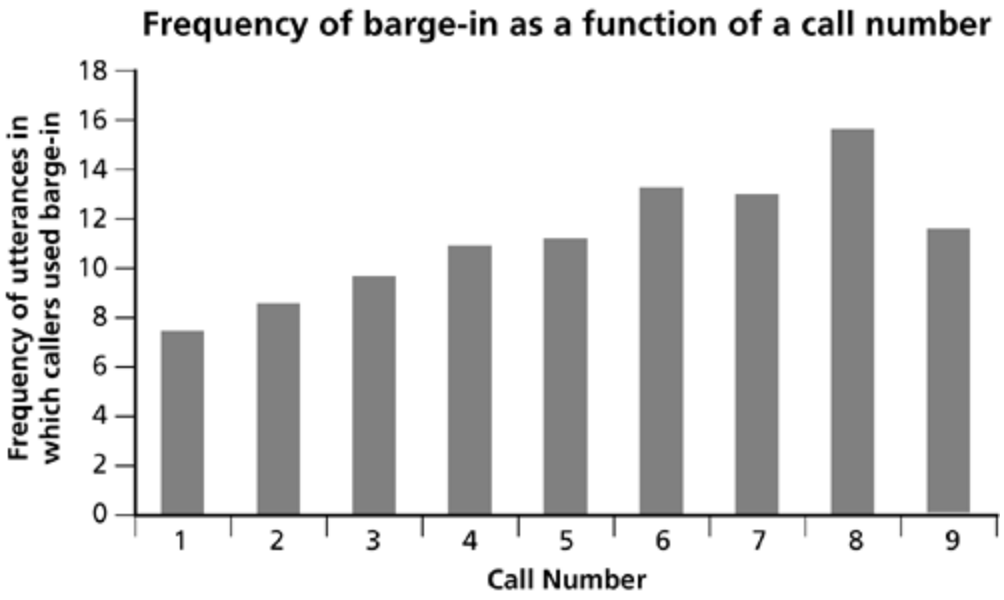
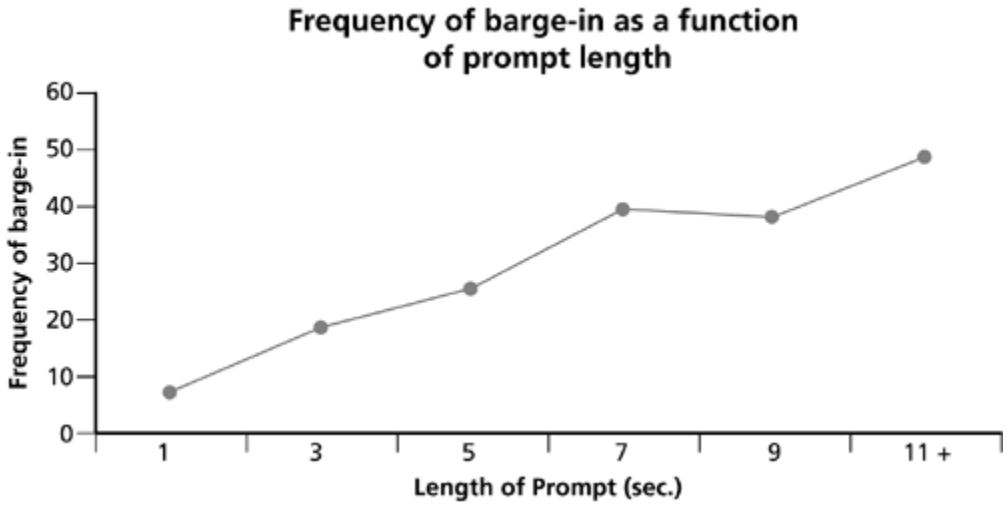


Figure 12-2. The longer the prompt, the more often callers barge in.


Frequency of barge-in as a function of prompt length
Frequency of barge-in
60
50
40
30
20
10
0
1
3
5
7
9
11 +
Length of Prompt (sec.)

12.4 Conclusion

Both efficiency and clarity are key qualities of all user interfaces. To maximize both when you design voice user interfaces, it is crucial to understand the different needs of first-time as opposed to experienced callers. You can then make trade-offs or create the flexibilities that permit each type of user to interact with the system to best advantage.

Chapter 13. Optimizing Accuracy and Recovering from Errors

High recognition accuracy is at the core of speech interface usability. Accuracy is key for all user interfaces; the system must correctly "understand" the user's intent and must respond appropriately. However, accuracy takes on special importance for voice user interfaces, for a number of reasons.

First, the impermanent and dynamic nature of spoken interactions makes it hard for the user even to identify errors when they happen, much less take appropriate action to correct them. Second, the technology is imperfect; there will be errors, especially in difficult acoustic environments, such as those that have background noise or cellular channel distortions. Therefore, it is essential to do whatever is possible in the dialog design to prevent errors and recover effectively when they do happen.

And problems with accuracy are exacerbated by the fact that callers usually can't understand why the system made the error. Although some errors may be obvious (for example, "Boston" and "Austin" sound similar and may get confused), the reasons for most recognition errors are obscure. As a result, errors typically cause confusion for callers as well as reduce their faith in the "rational behavior" of the system. Every error degrades the usability of the system. The best-designed and most graceful error recovery approaches cannot compensate for poor accuracy.

Evidence gathered from deployed systems shows that recognition accuracy is highly correlated with both task

completion and user satisfaction. This is also supported by laboratory data. Using a VUI experimental framework called PARADISE, researchers at AT&T analyzed three applications: two voice-enabled e-mail systems and a train information system. They found that recognition accuracy was the largest contributor to user satisfaction (Kamm, Walker, and Litman 1999).

In an experiment conducted at the Centre for Communication Interface Research (CCIR-1 1999) at the University of Edinburgh, user satisfaction was assessed for the same speech system at four accuracy levels. The researchers found that subjective attitudes toward the system improved as the accuracy increased.

There are a number of contributors to recognition accuracy. Many of them are related to acoustic modeling and search techniques and are independent of the individual application. However, the way you design your dialogs and grammar affects both accuracy and perceived accuracy. This chapter discusses dialog design guidelines that can contribute to accuracy. We then describe approaches for error recovery: handling those cases when accuracy fails.

13.1 Measuring Accuracy

Recognition accuracy is typically described by a number of measurements. The first step is to separate data into two groups: in-grammar and out-of-grammar. The in-grammar data are things callers say that are in the defined grammar for the relevant dialog state. Out-of-grammar data are not in the defined grammar.

In-grammar data fall into one of three categories:

1. **Correct accept:** The recognizer returned the correct answer.

2. **False accept:** The recognizer returned an incorrect answer.

3. **False reject:** The recognizer could not find a good match to any path in the grammar and therefore rejected rather than return an answer.

Out-of-grammar data fall into one of two categories:

1. **Correct reject:** The recognizer correctly rejected the input.

2. **False accept:** The recognizer returned an answer that is, by definition, wrong because the input was not in the grammar.

From the point of view of callers, the distinction between in-grammar and out-of-grammar is irrelevant. All they know is that they said something, and the dialog with the system is not proceeding as they expect. Especially in the case of systems that are mainly single use (rather than called repeatedly by the same person), the user is often unaware

that there has been an error and is confused by how the dialog is progressing.

On the other hand, from the point of view of the designer or developer trying to improve system performance, two thingsthe distinctions between in-grammar and out-of-grammar data and the categories of errorsare important, as you will see.

13.2 Dialog Design Guidelines for Maximizing Accuracy

Many activities during design and development contribute to accuracy. Some we have covered earlier (e.g., the use of acoustic adaptation, mentioned in Chapter 2). Others are covered in Chapter 15 (tuning recognition parameters) and Chapter 16 (tuning grammar coverage). In this chapter, we cover some of the design choices you can make during detailed design that can help optimize the accuracy of difficult recognition tasks.

A number of common recognition tasks are extremely challenging. One example is the alphabet. The names of 25 of the 26 letters in the English alphabet have one syllable and thus provide limited acoustic information. Furthermore, 9 of the letter names rhyme with *E*; these are often referred to as the **eset**. Another 4 have a vowel rhyming with *A*. Another two*F* and *S*are hard to distinguish over the phone. Telephones don't transmit frequencies higher than about 3,500 cycles per second, which is where most of the information to distinguish *F* and *S* lies. Recognition of alphabetics comes up in applications requiring spelling (e.g., of person names or street names) and often as part of account IDs.

Another common and challenging recognition task is digit strings. Most of the names of the digits have one syllable. Even if recognition of individual digits is extremely high, when you string many of them together, the recognition rate on the entire string (getting every digit right) may be low. Many common recognition tasks require digit-string recognitionfor example, account numbers, PINs, telephone numbers, credit card numbers, and social security numbers.

When handling tough recognition problems such as strings of alphabetics, digits, or a combination of both (alphanumerics), you should look for all possible ways to constrain which strings are valid, as well as all possible sources of knowledge that can be brought to bear to limit the possibilities the recognizer must consider.

In general, you can apply these constraints either by building them into the grammar or by postprocessing an N-best list (i.e., choose the first item on the N-best list that fulfills the constraint). It is best to build these constraints into the grammar, if at all possible. This approach will result in higher accuracy.

Here are examples of encoding structure in grammar:

- **Fixed-length digit strings:** If the string will always be the same length, write a grammar that accepts only strings of that length. It is far easier to accurately recognize fixed-length strings than variable-length strings.

- **Combinations of fixed-length strings:** In some cases, you may be working with more than one valid string lengthfor example, phone numbers that may be local (seven-digit) or long distance (ten-digit).

- **Constraints on particular character positions:** Account IDs may have constraints on various positionsfor example, the first two positions may contain one of the 50 U.S. state codes.

- **Constraints on relationships between character positions or groups:** An example of this, for phone numbers, would be a grammar that is constrained to capture only the valid exchanges for each area code.

- **Spelling of valid words:** If the recognition task is spelling and if the vocabulary being spelled is from a known set (e.g., person names, English words), you can train a statistical language model (SLM) so that it learns the probabilities of letter sequences. This is likely to be a useful source of constraint, because in English certain letter combinations are far more common than others.

Here are examples of encoding structure by post-processing an N-best list:

- **Checksums:** Many account numbers are designed with some form of checksum digit. The checksum digit is computed as a function of the other digits in the account number. The checksum computation is usually designed so that there is a roughly uniform distribution of checksum values (over the ten digits) given the expected distribution of account numbers. As a result, nine times out of ten, if there are any recognition errors in the account number, the checksum will not be valid. You can perform the checksum test on each item in the N-best list, choosing the first one that matches.

- **Database of valid items:** Often, only a small subset of the possible account numbers is actually in use. If you have access to the database of existing account numbers, you can go down the N-best list, testing each item for validity.

Another difficult recognition problem is recognition from an extremely long list, such as street names. One approach for maximizing accuracy is to constrain the grammar dynamically (as the application is running) based on information previously supplied by the caller. For example, if the zip code is known, you can dynamically load a

grammar with only the streets for that zip code. Dynamic grammars are discussed in Chapter 16. In general, recognition from long lists can be improved by incorporating probabilities into the grammar based on in-service data from the application.

13.3 Recovering from Errors

One of the fundamental challenges for speech technology (and therefore a challenge for the VUI designer) is the capability to know when there is a problem. When a false accept occurs, the system proceeds, given what it understood the caller to be saying, without realizing that something is wrong. To deal with this, the application must use a variety of techniques to confirm results with callers, when necessary and appropriate, to avoid taking undesired actions.

When the system does know there is a problem (e.g., the recognizer returns a reject), a gap often still exists between what the system knows and the root cause of the problem. In the case of a reject, the system knows only that there was no path through the recognition model that was a good match for the input. However, the reject may be caused by out-of-grammar input from the caller, a noisy cellular connection, a rock 'n' roll band playing in the caller's living room, or a variety of other problems. Despite the uncertainty about the underlying cause, the application must get the dialog back on track.

This section covers confirmation strategies to detect and fix recognition errors when they occur, and recovery strategies to use when the recognizer returns an error condition such as a reject or timeout.

13.3.1 Confirmation Strategies

False accepts may cause serious problems. The system may take an action (e.g., buy 1,000 shares of IBM) or transition to an unintended subdialog. In general, detecting

false accepts requires some sort of confirmation dialog in which the system validates its hypothesis with the caller (e.g., "Going from New York to San Francisco. Is that correct?"). There are a number of issues concerning the design of confirmation dialogs, including

- When to confirm
- How to confirm
- Avoiding repeat errors

When to Confirm

Confirmation is necessary if the cost of an error is high, if it would be hard to get back on track, or if recognition confidence is low. For example, before executing a transaction (e.g., transferring money, booking a flight), you would certainly want to confirm the parameters of the transaction with the caller. Failing to confirm in such cases, even if the system is highly confident that it has the right information, not only risks a costly mistake but also tends to reduce caller confidence.

Keep in mind that there are two roles of confirmation: first, to allow the system to validate its recognition hypotheses, and second, to communicate correct recognition to the caller, thereby buttressing user confidence and making transparent the current state of system knowledge.

On the other hand, there are cases when confirmation is not necessary. Many stock quote systems don't explicitly confirm company names. If the company name is misrecognized, the caller loses a few seconds hearing the wrong quote but then can proceed without problem. In fact,

callers can barge in as soon as they hear the wrong company name at the beginning of the quote and ask again for the desired company. The cost of the error is low in terms of time lost and in terms of user confusion. It is clear how to proceed. Furthermore, stock quote systems typically have very high accuracy, further reducing the need for confirmation.

Another approach is to use the confidence level returned by the recognizer to decide whether or not confirmation is needed. This is sometimes referred to as **three-tiered-confidence**: If the confidence is high, don't confirm; if it is low but above the reject threshold, confirm; otherwise, reject.

How to Confirm

When confirmation is clearly needed, there are a number of ways to handle it. You can confirm each item as it is spoken, or you can group related items for confirmation before completing a transaction or other application task. In general, the latter approach makes sense: Confirm a group of related items at the same time.

Let's compare the following two confirmation approaches for a travel application. The first one immediately confirms each item:

(1)

SYSTEM: What's the departure city?

CALLER: San Francisco.

SYSTEM: San Francisco. Is that correct?

CALLER: Yes.

SYSTEM: To what city?

CALLER: New York.

SYSTEM: New York. Is that right?

CALLER: Yes.

SYSTEM: Great. What day do you want to leave?

CALLER: I wanna leave March fifth.

SYSTEM: March fifth. Is that correct?

CALLER: Yes.

This approach can lead to a rather tedious and stilted dialog. In contrast, the following approach groups items for confirmation:

(2)

SYSTEM: What's the departure city?

CALLER: San Francisco.

SYSTEM: To what city?

CALLER: To New York.

SYSTEM: And on what date?

CALLER: I'm leaving on March fifth.

SYSTEM: Now let me confirm that: San Francisco to New York on March fifth. Is that correct?

CALLER: Yes.

This approach leads to a much more fluid and efficient dialog. In general, when you group items for confirmation, they should be logically related (e.g., the three parameters for a leg of a trip). Furthermore, a small number should be grouped. A good rule of thumb is to group no more than four items for one confirmation.

Avoiding Repeat Errors

When there is a negative response to a confirmation prompt, the system has some valuable information. It knows what the caller does *not* want. Systems should make sure not to repeat the same mistake.

Here's an example of this type of mistake:

(3)

SYSTEM: What's the flight number?

CALLER: Sixteen fifty-five.

SYSTEM: Now is this for arrival or departure information?

CALLER: Arrival.

SYSTEM: Let me confirm that: Flight 6355 arriving on

CALLER: No.

SYSTEM: Sorry, which part should I change, the flight number or the date?

CALLER: The flight number.

. . .

SYSTEM: What's the flight number?

CALLER: Sixteen fifty-five.

SYSTEM: Let me confirm that: Flight 6355 arriving on

CALLER: No! <hang up>

In addition to frustrating the caller, risking endless loops through the same error (until the caller hangs up in anger), and wasting time, this mistake hurts the image of the system persona. It appears to lack basic intelligence, memory, and the ability to track the conversation.

To avoid repeating mistakes, disconfirmed items should be put in a **skip list**. The ensuing recognition should be configured to return an N-best list. If the first item in the N-best list is on the skip list, then the application should skip (ignore) this item because it already knows that the information is incorrect. The next item on the N-best list that is not on the skip list should be the hypothesis.

13.3.2 Recovering from Rejects and Timeouts

Chapter 5 reviews basic error recovery strategies, including escalating detail and rapid reprompt. It also discusses the use of maximum error thresholds, the number of successive retries before transferring to a customer service agent. Here we discuss the design of the error prompts themselves, specific to each dialog state.

When you're designing error recovery prompts, the challenge is that you do not know the root cause of the problem. If the recognizer returns a reject, there can be many underlying reasons. If the reason is that the caller spoke out-of-grammar, that, in turn, may be the result of a number of different problems.

The best approach to designing error prompts is to try to understand the most likely reasons for the problem, considering the context of the dialog state. Given the limits described in Chapter 5 on the number of retries, you get

only a few opportunities (typically no more than three) to fix the problem before a caller gives up. Therefore, error recovery prompts should be designed to optimize the chance of successful recovery for the most likely problems. After the application is in pilot and real in-service data become available for analysis, you will have concrete data on the problems that callers are experiencing and may decide to further refine some of the error messages at that point.

Let's look at an example of failed error recovery:

(4)

CALLER: Order status.

SYSTEM: To better assist you in checking the status of your order, we need your order number. Please say your order number.

CALLER: I don't know what it is. <reject>

SYSTEM: I'm sorry, I didn't understand. Please say or key in your eight-digit order number. You can say, "Help" for more details.

CALLER: Help.

SYSTEM: Okay, here's some help. In order to assist you, we need your order number. Please speak naturally and say the eight digits of your order number. You can also enter them on your touchtone keypad.

CALLER: <hang up>

In this case, the caller hears the initial prompt, the first error recovery prompt, and the help prompt. As he transitions through the various prompts, he receives more instruction about *how* to say his order number. He even hears that he can key it in.

One implicit assumption behind this design (an assumption that turns out to be false) is that the caller knows his order number or has his invoice and knows where to find it. Other assumptions (which could easily have been false) are that the caller landed in this dialog state for the right reasons (e.g., not because of a recognition error or caller confusion) and that he never changed his mind about what he wanted to do.

Assuming an escalating error handling strategy, here is an alternative set of prompts for that dialog state:

INITIAL PROMPT:	Okay, order status. Please say your order number.
REJECT 1:	I'm sorry, I didn't understand. To check the status, I'll need your order number. It appears in the upper left-hand corner of your invoice. Please say your eight-digit order number now. You can also say, "Help" for more information.
REJECT 2:	I'm sorry, I still didn't understand. If you know your order number, please key it in. Otherwise, say, "I don't know" and I'll connect you to someone who can help you.
REJECT 3:	I'm sorry, I'm still having trouble. Please hold while I transfer you to a customer service representative who can help you.
HELP:	Okay, here's some help. I need your order number so I can check the status. If you know it, you can say it or key it in now. Otherwise, say, "I don't know," and I'll connect you to someone who can help you. At any time, you can go back to the beginning by saying, "Main menu."

This approach accounts for a number of possible problems that the earlier strategy did not. First, we are handling the case in which the caller does not know his order number (assume that this grammar includes phrases such as "I don't know," "I don't know what it is," "I don't have it," etc.). Second, the help prompt lets the caller know about the main menu universal, so there is a way to recover if the caller got to the order status part of the application by mistake or has changed his mind. Third, given that for tasks like this the caller will often have his invoice in hand (assume we learned this from researching usage profiles during requirements definition), he is told how to find his order number.

When there is more than one reject in a state, it is often advisable to try an alternative approach, if one is available, to help the caller complete the task, as when the preceding example allows touchtone input of the order number. The following example, from a flight information system, shows another approach:

(5)

SYSTEM: What's the flight number?

CALLER: . . . <reject>

SYSTEM: What was that?

CALLER: . . . <reject>

SYSTEM: I still didn't get it. Let's try another approach. What's the origin city of the flight?

CALLER: Boston.

SYSTEM: And what's the destination?

CALLER: New York, Kennedy Airport.

SYSTEM: Okay. And what time is it due?

CALLER: Around noon.

SYSTEM: Got it. American Airlines flight 135 will be arriving at JFK on time at 12:15 p.m. at gate 3.

You can apply the following guidelines when crafting error recovery and help prompts. These guidelines are based on an analysis of errors observed in deployment data across many applications.

- When appropriate, provide more specific detail about what can be said (what is in-grammar).

- Always provide an escape (the ability to navigate elsewhere, such as back to the main menu). Callers may be in the dialog state because of a recognition error or confusion, or they may change their minds about what to do.

- Teach callers about universals.

- After a second error, present an alternative means to accomplish the goal, if there is one.

- When requesting specific pieces of information (e.g., account numbers, ID numbers, medical group numbers), accommodate callers who do not have the information. If the information is easily available on something callers are likely to have handy (e.g., an invoice, a medical card), tell them how to find it.

- When appropriate, provide examples. Callers often pattern their answers after examples. Therefore, examples can help with types of information that can have many formats (such as dates) and when callers are using unexpected fillers. For example:

(6)

SYSTEM: What's the date of travel?

CALLER: Um, I gotta get there by next Saturday. <reject>

SYSTEM: I'm sorry, I didn't understand. Please say the date of travelfor example, "January eighth" or "April twenty-third."

- When there are rejects after confirmation prompts, tell callers to answer "Yes" or "No." Often, when callers respond negatively to a confirmation prompt, they try to correct the error in the same response (e.g., "No, not BostonAustin"). Because it is hard to cover the great variety of ways callers may combine disconfirmation and the correction in a grammar, it is best to constrain them to a yes/no grammar and then step them through the correction process.

- If lots of no-speech timeouts are observed for a state (either in usability data or pilot data), consider a delayed help approach (see Chapter 12) for the initial prompt.

13.4 Conclusion

Using the approaches in section 13.3 should allow you to design strategies for helping the caller and the system to reestablish a productive dialog even after one or two errors or misunderstandings have occurred. Never forget, though, that error prevention is always preferable to error recovery. The image of the company deploying the system will be best enhanced if you prevent errors in the first place through the use of highly accurate recognition technology coupled with accuracy-enhancing dialog design that includes graceful error recovery strategies.

Clear and consistent dialogs, coupled with successfully constraining interface design, should minimize errors. When errors do occur, tasks can still be completed and callers can still finish their interactions with a sense of having dealt with an intelligent, efficient, and considerate agent.

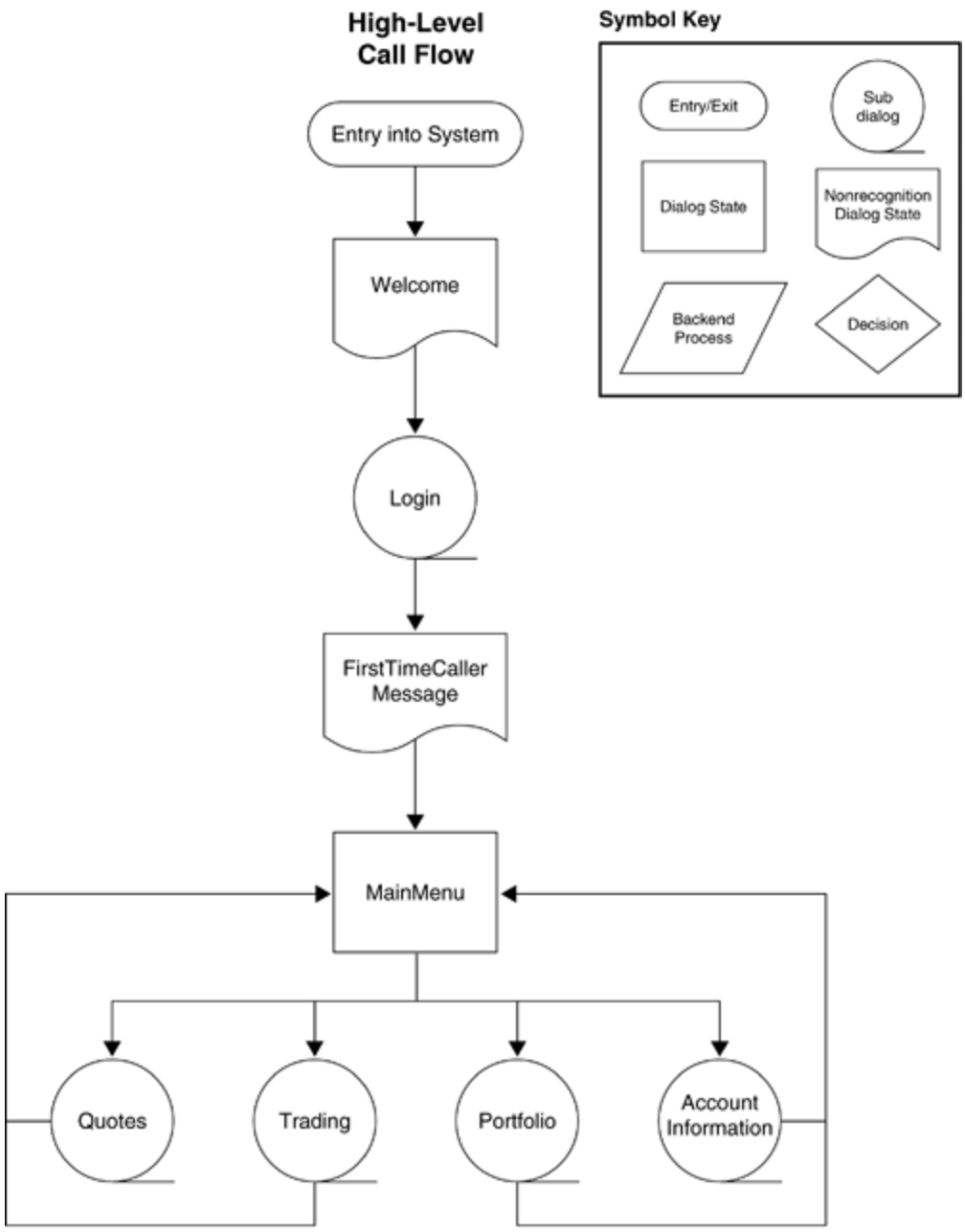
Chapter 14. Sample Application: Detailed Design

We are now ready to begin the detailed design of the Lexington Brokerage application (introduced in Chapter 7). To get early user feedback to validate our basic design decisions, we decide to do an initial design of the most critical parts of the system and to do a usability test using a Wizard of Oz setup. We plan to create enough of the design so that callers can log in, get quotes, and make trades. Once the initial design is validated, we will flesh out the remainder of the design in complete detail.

14.1 Call Flow Design

We start by creating a high-level call flow of the entire system (see Figure 14-1). The call flow shows the basic architecture of the dialog. The major components of the system are encapsulated in subdialogs. Callers first enter the `Welcome` state, where they hear the branding earcon followed by the welcome prompt. Then they enter the Login subdialog, which we will flesh out later.

Figure 14-1. The high-level call flow for Lexington Brokerage shows the basic architecture of the application.



After successfully logging in to the system, new callers will hear the first-time caller message, which introduces the customer to the application. Rather than use an extremely long first-time caller message or offer a tutorial, we decide to keep the first-time caller message short and use just-in-time instruction to gradually teach users about system capabilities. Just-in-time instructions will be triggered occasionally to teach callers about functionality they are

not using and to help them the first time they exercise a feature.

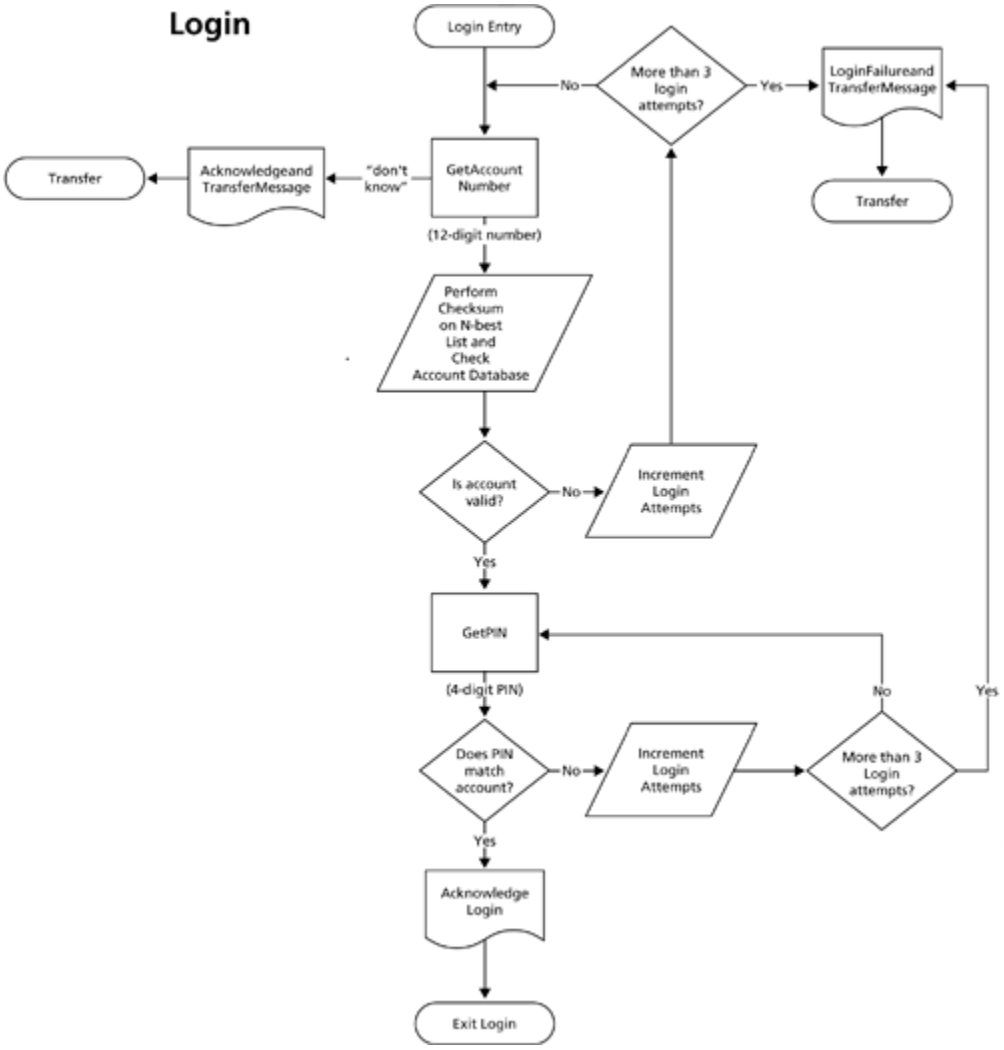
Following login and the first-time caller message, callers reach the main menu. Here they will be given the flexibility to make almost any request. For example, they can say, "Give me a quote on Cisco," "I wanna make a trade," "Buy 100 shares of Apple," "Buy 100 shares of Apple at the market," and so on. The details of each type of operation are then handled in the subdialogs that follow the main menu.

To prepare for our WOZ usability test, we must specify the designs for the Login, Quotes, and Trading subdialogs. Ultimately, all the subdialogs will be fleshed out so that we have call flows that show every dialog state in the system.

14.1.1 The Login Subdialog

Figure 14-2 shows the call flow for the Login subdialog. A number of important dialog strategies are used in this subdialog. First, this is where we execute the approach discussed in Chapter 7 to optimize the accuracy of account number recognition. We do N-best recognition in the `GetAccountNumber` state, and in the ensuing state we test the checksum digits of the items on the N-best list to see which ones are valid account numbers. For valid account numbers, we query the account number database to find out whether they are actually in use. We choose the account number with the highest confidence (earliest on the N-best list) that has a valid checksum and exists in the database. If no items on the N-best list are valid, we query the caller again.

Figure 14-2. The call flow for the Login subdialog shows how we optimize accuracy and handle login problems.



Another design choice shown in the Login subdialog is what we do when callers do not know their account number. We learned in our observational study that some callers do not remember their account number and do not have an account statement handy. In those cases, the brokers can access their accounts by asking for some personal information (e.g., name, address, mother's maiden name). Rather than terminate calls to our system if the account

number is unknown, we decide to transfer callers to an agent who can help them. Furthermore, we include phrases such as "I don't know" in the grammar for the GetAccountNumber state, and when there are rejects or requests for help, we tell callers that they can say, "I don't know" and get assistance.

If callers are having trouble logging in with their account number, we want to avoid hang-ups by transferring them to an agent who can help them. To strike a balance between maximizing automation rates and avoiding hang-ups, we choose to transfer them after the third failure in the `GetAccountNumber` state. Whenever we transfer callers to live agents, we first play a message clarifying what is happening (e.g., see states `LoginFailureandTransferMessage` and `AcknowledgeandTransferMessage` in Figure 14-2). This accords with results reported in CCIR-2 (1999), in which callers expressed a strong preference to be told when and why they were being transferred.

14.1.2 The Quotes Subdialog

Two dialog strategy issues come up in the design of the Quotes subdialog: handling ambiguous company names and the use of just-in-time instruction to teach users about watch lists.

Many company names are ambiguous. For example, Cisco (Systems) and Sysco (Foods) are pronounced the same way. There are five different companies that can be referred to as Genesys. When a quote is requested on an ambiguous company name, the system must figure out which company is intended.

There are a number of possible strategies. One approach is to simply tell callers that there are a number of companies with that name and ask them to use the *full* company name. This approach has the advantage of efficiency. Another technique is to provide a list of the full names of all the companies and ask callers to repeat the name when they hear the one they want. This approach is less efficient than the first one, but it has the advantage that the caller is told the full names and thus gains greater clarity about what to say.

For the Lexington application we decide on a hybrid approach that maximizes efficiency but ultimately provides a list if necessary. Our approach has three stages (specific prompt wordings for these are worked out later in section 14.2):

1. If there are two or three companies, say the full names and ask which one is desired (e.g., "Do you want Cisco Systems or Sysco Foods?").

2. If there are more than three companies, tell callers that there are multiple companies with that name and ask them to say the full name.

3. If there are more than three companies and if the caller does not provide a successful response to the system's request for the full name (e.g., there is a <reject> or a request for help), then list the full names of the companies. Before presenting the list, ask callers to repeat the name of the company of interest as soon as they hear it. Supporting caller barge-in as soon as they hear the name is effective because it makes the task a recognition problem rather than a recall problemcallers respond as soon as they recognize the company rather than try to recall it after hearing the entire listthereby lowering cognitive load. Because of possible timing problems, it works better to have callers repeat the

name rather than say, "That one"; by the time the caller says "That one," the system may be listing the next company.

We decide to use just-in-time instruction to teach callers about the watch list feature. From our observational study and from usage statistics of the touchtone system, we know that callers often ask for numerous quotes, so we believe they will find the watch list feature useful. We also want to encourage the use of a watch list, because it personalizes the system to customers' needs and makes the system more sticky.

We want to be careful not to trigger the watch list instruction too often. We set the following criteria:

- Do not trigger the watch list instruction on a user's first call to the system. The first-time caller already has a lot to learn.
- On the user's second call to the system, if she has not yet set up a watch list and if she has asked for at least two quotes on this call, trigger the watch list instruction.
- On the user's fourth call to the system, if he has not yet set up a watch list and if he has asked for at least two quotes on this call, trigger the watch list instruction.

14.1.3 The Trading Subdialog

If a caller specifies a trade while at the main menu, the system transitions to the Trading subdialog. There are two important dialog strategy issues to consider for the Trading

subdialog: transitioning between a user-initiative and a system-initiative dialog; and performing careful confirmation before completing any transactions.

At the main menu, callers may provide different amounts of detail about a trade. For example, all the following would be valid requests: "Buy Intel," "Buy a hundred shares of Intel," "Sell all my shares of Intel," "Buy Intel at twenty-eight," "I wanna make a trade," "Sell," and "Buy Intel at the market, good for the day." If the request is incompletethat is, if any information is missingthe system must then take over the initiative and query the caller for the missing information.

We decide to query for one item at a time, in a directed fashion, in order to complete the request. For example, if the caller says "Buy Intel," the system will first ask, "How many shares do you want to buy?" and then will ask, "At what price?" We decide to accept complex input even during directed querying for single items; for example, after asking, "How many shares do you want to buy?" our grammar will accept responses such as, "Buy fifty shares at twenty-two point three three." In this way, we are not forcing callers to change their mental model from one that allows them to use complex sentences with multiple information items, even though we have transitioned into a directed prompting style.

Before completing the trade, the system will carefully confirm the details with the caller. We will slow the pace a little and play a prompt such as, "You want to buy two hundred shares of Intel, at the market, good for the day. Is that correct?" We will move ahead with the transaction only if we are highly confident that the caller has confirmed it. Therefore, we use a strict yes/no grammar: one that accepts only "Yes" or "No" and does not accept input such as, "Yeah, that's right," "Okay," and so on. Furthermore, we raise the reject threshold a bit to reduce the chance of false

accepts of "Yes." If we do need to go back and correct anything, we use skip lists to make sure we do not make the same mistake twice.

We complete the designs of the Quotes and Trading subdialogs, producing call flow diagrams describing every dialog state.

14.2 Prompt Design

After we complete the high-level call flow and the Login, Quotes, and Trading subdialogs, we begin to work on the prompting. We do this by specifying sample dialogs to cover the various dialog paths.

Here's an initial sample dialog, starting with the welcome prompt and the Login subdialog:

(1)

SYSTEM: <audio brand> Thanks for calling Lexington Brokerage. What's your account number?

CALLER: 837620868247.

SYSTEM: Okay. Now tell me your PIN. <pause> You can also enter it with touchtones.

CALLER: 1743.

SYSTEM: Great.

We then continue the sample dialog in (2), assuming that this is a first-time caller:

(2)

SYSTEM: With this system you can get quotes, place trades, access your account, and get information about your portfolio. If you ever feel lost or want more information, just say, "Help." So what can I do for you?

CALLER: I wanna buy a hundred shares of Apple.

SYSTEM: Okay, at what price?

CALLER: At the market.

SYSTEM: Confirming: You want to buy one hundred shares of Apple Computer, at the market. Good for the day. Is that right?

CALLER: Yes.

SYSTEM: Your order's been placed. I'll let you know when it's been filled.

. . .

Example (2) shows the system transitioning from a wide-open prompt ("So what can I do for you?") to a directed prompt ("Okay, at what price?"). This dialog also uses careful confirmation of the trade details before it is placed.

We work through a number of sample dialogs to flesh out the company-name disambiguation approach in the Quotes subdialog. First, we handle a case of two companies having the same name:

(3)

CALLER: Sysco.

SYSTEM: Do you want Cisco Systems or Sysco Foods?

CALLER: Sysco Foods.

Next, we handle a case of five companies having the same name:

(4)

CALLER: Genesys.

SYSTEM: There are five companies with that name. Please say the company's *full* name.

CALLER: Um. I'm not sure what the full name is.

SYSTEM: <reject> Sorry, I didn't understand. There are five companies with the name you requested. When you hear the one you want, please repeat the *full* company name. Genesys Bioventures. <pause> Genesys Microchip. <pause> Genes…

CALLER: Genesys Microchip.

As we work through sample dialogs for the Trading subdialog, we realize that some states will need multiple possible initial prompts, depending on the context. In (5), we enter the state that queries for the number of shares twice; the second entry is to correct a misrecognition. As you can see, the prompt wording and prosody are different the second time we enter.

(5)

CALLER: Buy Intel.

SYSTEM: How many shares?

CALLER: Two hundred fifty.

SYSTEM: And at what price?

CALLER: At twenty-eight.

SYSTEM: Confirming: You want to buy two hundred fifteen shares of Intel at twenty-eight. Is that right?

CALLER: No.

SYSTEM: Do you want to change the company, the number of shares, the price

CALLER: The number of shares.

SYSTEM: Okay. How *many* shares do you want to buy?

In a similar fashion, we create sample dialogs for many possible paths. In some sample dialogs, we will get quotes and then make a trade. In others, we will flesh out some of the error handling.

When we have a full set of sample dialogs, we get together with a colleague and read through them out loud. Upon hearing them in this conversational context, we gain some new perspective and make a few adjustments.

We then flesh out the dialog specification for the high-level call flow, Login, Quotes, and Trading, including all prompts and error handling. For each state, we try to imagine the most likely causes of problems and design error messages to recover effectively. For example, the first time there is a reject or timeout in the `GetAccountNumber` state, we remind callers that they can say, "I don't know"; the second time, we suggest that they key in the account number.

When this part of the design is complete, we are ready to prepare for the WOZ test.

14.3 User Testing

To prepare for the WOZ test, we start by establishing the goals. First, as described earlier, we want to test the main paths of the design to catch problems early. We also identify two high-risk dialog strategies we want to be sure to test: company name disambiguation and recovery from recognition errors when users specify trades. Additionally, we want feedback on the persona design.

We record a set of preliminary prompts covering the dialogs we have designed, as well as a set of company names that will be used in the test. Although audio productionwhich includes choosing a voice actor, recording, postprocessing prompt files, and creation of NVAis considered part of the development phase of the project, it often begins early.

Choosing the voice actor early gives us an opportunity to get early feedback about the persona design. For the Lexington project, it is important to begin the recording process early to avoid delays, because we will ultimately need audio files to cover approximately 15,000 names of companies, mutual funds, and indexes (handling the NASDAQ, New York Exchange, and American Exchange). The recording of company names can begin early, before final wording for other prompts is determined. For this project, we have already chosen the voice actor, and therefore we can create the WOZ recordings with his voice (the details of choosing the voice actor are covered in Chapter 18).

The designer will act as wizard, using a tool that allows control of the call flow and rapid selection of prompts to play. We also design a number of task scenarios for participants to perform, including logging in, getting stock

quotes, and making trades. We make sure to include some companies with ambiguous names in the test of the Quotes subdialog. We also intentionally introduce some "recognition errors" in order to test the confirmation and correction approaches in the Trading subdialog.

We formulate a questionnaire for each participant to complete after talking with the system. In this way, we can measure subjective preferences consistently across participants and get an average score that we can use as a baseline for future usability tests. Our survey includes all the relevant questions in the standard questionnaire Lexington has used to assess its touchtone system. Figure 14-3 shows some of the questions we included in the survey.

Figure 14-3. You can use a questionnaire like this one in early user testing.

1. Overall, the system was easy to use

Strongly Disagree	Disagree	Disagree Somewhat	Neither Agree nor Disagree	Agree Somewhat	Agree	Strongly Agree
1 ❑	2 ❑	3 ❑	4 ❑	5 ❑	6 ❑	7 ❑

2. The system was quick and efficient.

Strongly Disagree	Disagree	Disagree Somewhat	Neither Agree nor Disagree	Agree Somewhat	Agree	Strongly Agree
1 ❑	2 ❑	3 ❑	4 ❑	5 ❑	6 ❑	7 ❑

3. The system had the capabilities I expected.

Strongly Disagree	Disagree	Disagree Somewhat	Neither Agree nor Disagree	Agree Somewhat	Agree	Strongly Agree
1 ❑	2 ❑	3 ❑	4 ❑	5 ❑	6 ❑	7 ❑

4. What are the top two suggestions you could make to improve the system?

   a. _____

   b. _____

5. What are the top two things you liked about the system?

   a. _____

   b. _____

Questions about ease of use, efficiency, and flexibility are included because these are important metrics we chose in the requirements phase. We omit questions about accuracy until a later usability test, when callers are interacting with the real speech system. In this test, given that we are purposely introducing recognition errors, measures of callers' perceptions of accuracy would be meaningless.

In addition to the questions shown in Figure 14-3, we include a few questions about the persona. Two are open-ended: "What did you like or dislike about the voice?" and "Please describe the qualities that the voice conveys, as if the voice were a person." A few others are Likert-scaled, to be rated from "strongly disagree" to "strongly agree": "The character behind the voice seemed knowledgeable," " . . . energetic," and so on.

Next, we work with Lexington to get the names of customers who might be willing to participate in the test. The plan is to run the test over the phone; we will call the participants at home. We identify 12 participants.

We ask our usability engineer to run the study because she has not been working directly on the project and will remain an objective evaluator. We get everything scheduled. Before the first day of testing, we run some pilot tests with a few people internally to make sure the WOZ system is functioning properly and also to give our wizard some practice in responding to callers' utterances.

In the course of two days, we run all the test participants. We learn a number of things that help us improve some prompt wording and error messages. The participants complete most of the tasks they are assigned, and they find the system easy to use. They respond positively to the persona.

However, the subjective ratings on question 2 ("The system was quick and efficient") are lower than we had hoped; the average score is 4 (neutral). Upon review of the data, we conclude that most subjects do not take advantage of the mixed-initiative dialog strategy. When placing a trade, most subjects say things such as "Place a trade" at the main menu and then rely on the system to lead them through the steps.

As we review the dialog, it becomes clear that we have not adequately taught users about the advanced capabilities of the system. There are no instructions or examples that encourage them to place trades or get quotes directly from the main menu state. Similarly, users have not been shown that they can describe all the parameters of a trade in a single sentence.

Rather than add elaborate instructions to the beginning of the application, we decide to use additional just-in-time instructions to gradually teach callers about the advanced capabilities. For example, immediately after callers complete a trade step-by-step, we will play the message, "By the way, in the future you can save time by describing your trade in a single sentence. For example, you can say, 'Buy one hundred shares of Cisco at eighteen dollars' or 'Sell all of my shares of Apple at the market.'" A follow-up usability test will be run to verify that the just-in-time instruction approach has worked.

The follow-up usability test will also use a WOZ approach, also conducted over the telephone. Before the follow-up usability test, we complete the designs of the Portfolio and Account Information subdialogs. We create three tasks for each participant. All three tasks will include trading. If they complete their trades step-by-step, they will hear the just-in-time instruction. One of the tasks will include quotes, the second will include accessing portfolio information, and the third will include account information. We use the same questionnaire as in the first usability test.

With the help of Lexington, we line up 10 participants. In the course of two days, we run all the tests. We analyze the data, and the result is positive. The just-in-time instruction approach seems to be working. On the first task, most participants place their trades step-by-step and hear the just-in-time instruction. By the second task, seven out of

the ten use more complex and efficient approaches (e.g., "Buy fifty shares of Intel," "I wanna buy fifty shares of Intel at the market"), and by the third task, all of them do. The average score on question 2 (about efficiency) goes up from 4 to 6.2.

After the usability test, we make a few minor changes based on what we have learned, and we complete the design of the system. Our ultimate design is informed by the work we have done thus far in defining call flow, crafting prompts, and testing the design on end users.

14.4 Conclusion

Our design work is now complete. The design has been driven by our understanding of the business priorities and user needs. Design decisions have been refined and validated based on user input. Our prompts have been crafted to flow well and communicate clearly, based on sample dialogs and a clear definition of a persona.

We are now ready to shift our primary focus to implementation. As discussed in Chapter 3, the project phases often overlap to some degree. On this project, we began both audio production and backend integration early, while design was still ongoing.

The next part of the book discusses development, testing, and tuning. We describe the entire process, with emphasis on those tasks that the VUI designer often has primary responsibility for.

Part IV: Realization Phase: Development, Testing, and Tuning

Chapter 15. Development, Testing, and Tuning Methodology

Part IV covers the **realization phase** of the project: everything it takes to turn a design into a working, deployed, high-performance system. The realization phase involves development (of software, backend integrations, grammars, and audio recordings), testing of all components before release for pilots and deployment, and tuning the performance of the system during pilots and in early deployment.

Entire books could be written about each of these steps. Given our focus on design, our goals in this part of the book are twofold:

- Covering in detail those steps that are often designers' direct responsibility (e.g., voice coaching and grammar development)

- Covering all other steps in sufficient detail so that designers can understand the complete process of deploying a system and comprehend how their role relates to all other roles on the project

15.1 Development

The development phase includes the creation of the application software, the development of the grammars to be used by the recognizer, and the production of the audio for the prompts and any nonverbal audio that is part of the application.

15.1.1 Application Development

Application software development entails the coding of the dialogs as well as integration with backend databases and other software systems.

Application logic that implements the dialog can be written in a number of languages. Occasionally it is written in standard programming languages such as C, C++, or Java. More commonly it is coded in a proprietary language associated with an interactive voice response (IVR) platform that is designed to run touchtone and speech applications over the telephone network. Many IVR platforms have their own proprietary tools and languages that developers must use to implement applications.

A growing trend, which is likely to become the dominant implementation approach in the not-too-distant future, is to code the dialog in VoiceXML, a markup language for building speech interfaces. It is similar in spirit to HTML, the markup language often used for creating Web sites.

One reason for interest in VoiceXML is that it is becoming a widely used standard controlled by the World Wide Web Consortium (W3C). Furthermore, it is a speech-centric language, with basic elements such as menus

(presentations to the caller of a set of choices) and forms (used to collect information from the caller, often handling multiple pieces of information in a single form). VoiceXML is an XML-based language and thus leverages the Web development paradigm already familiar to many programmers. There are a number of excellent references on VoiceXML, including Larson (2003), Sharma and Kunins (2002), and Abbott (2002).

In addition to the dialog code, application development involves the integration of your application with other software systems such as backend databases, computer telephony integration (CTI) software, and Web services. In some casesfor example, if the speech system is replacing a touchtone systemthe integration code may already exist. Most IVR platforms support integration with databases and CTI systems.

15.1.2 Grammar Development

The development of effective grammarsthose that have high coverage of things callers are likely to say to the systemis key to the development of a usable system. Grammar development is often a task of the VUI designer. Even if someone else develops the grammar, there is an intimate relationship between the grammar, the prompt wording, and the chosen dialog strategies. Chapter 16 covers the development and tuning of grammars in detail.

15.1.3 Audio Production

Audio production involves the recording of all the prompts by an appropriate voice actor. The designer often acts as the voice coach at the recording sessions, working with the

voice actor to create the persona. The coach must also make sure that the voice actor understands how all the prompts function in context. Prompts that are to be concatenated must be recorded with appropriate prosody so that concatenation splices can be seamless. Chapter 17 covers the details of preparation and coaching of recording sessions with voice actors.

If the application uses nonverbal audio, final recordings must be prepared and processed. Final recordings should be tested over the phone.

15.2 Testing

Individual components of the application (e.g., dialog code, integration code) are **unit-tested** (tested in isolation) after they are developed. The testing we discuss in this section involves the fully integrated system. There are a number of tests to perform before release of the system for pilots or deployment, including application testing, recognition testing, and evaluative usability testing. All these tests are performed with the complete working system.

15.2.1 Application Testing

The application must be tested to make sure it meets the dialog design precisely as specified, does not have critical bugs, and is adequately provisioned to meet the expected call volumes. The tests performed at this stage include the dialog traversal test, the system QA test, and the load test.

Dialog Traversal Test

The purpose of **dialog traversal testing** is to make sure that the system accurately implements the dialog specification in complete detail. You perform the test with the live system, over the telephone, exercising a test script that thoroughly traverses the dialog. The correct actions must be taken at each step, and the correct prompts must be played.

Every dialog state must be visited during the test. Within each dialog state, every universal and every error condition must be tested. For example, you should try an out-of-

grammar utterance to test the behavior in response to a recognition reject. You should try silence to test no-speech timeouts. You should impose multiple successive errors within dialog states to ensure proper behavior. If your design associates global behaviors with error countsfor example, automatically transferring callers to an operator if they exceed a fixed number of errors in a single phone callyou must also exercise that behavior.

Dialog states that have multiple entry prompts should be tested from each possible entry point. For example, if a state has an alternative prompt if callers enter it after a disconfirmation, the test script should include a path covering the disconfirmation. You should test every transition out of a dialog state to make sure that the system transitions to the correct next dialog state. While traversing the dialog, in addition to noting the correctness of system behavior, the tester should note any prompt (or other audio) recordings that are distorted or have other audio problems.

System QA Test

The **system QA test** is similar to other integration tests of large software systems. A test suite is executed that exercises all integrations, all conditions, and all failure modes.

Load Test

The purpose of a **load test** is to make sure that the system can efficiently handle the peak loads expected during the busiest hour of usage. The load test is typically run after the software is installed on the final system in the call

center or at the service provider. You can run load tests by having numerous people call in to the system simultaneously. More typically, you use a software system that simulates the load by placing multiple calls to the system.

15.2.2 Recognition Testing

In general, reliable testing of recognition accuracy and tuning of recognition parameters can be accomplished only with a substantial amount of data collected from an in-service system. The purpose of the prepilot **recognition test** is to make sure that recognition accuracy is in the ballparkroughly what you would expect given the complexity of the recognition problem in each dialog stateand that the initial values of the recognition parameters are reasonable.

You will fine-tune recognition parameters and accuracy during pilots and early deployment. However, the recognition test assures you that you will not expose pilot callers to unnecessarily poor performance. It is also important not to waste the early pilot fixing bugs and poorly chosen parameters that should have been better adjusted at installation time. The recognition test also ensures that the results of evaluative usability testing (see the next section) will not be tainted by poor recognition.

The recognition test is based on a relatively small number of callers (1020). To make sure that the grammars are thoroughly exercised, you give each caller a script.

As an example of the value of the prepilot recognition test, occasionally issues with endpointing parameters will surface. One of the endpointing parameters controls how long, after speech has been detected, the recognizer will

listen to a silence before deciding that caller speech has ended. From time to time, that parameter will be too short for certain dialog states. In particular, certain types of caller speech may have natural pause points that lead to longer than usual pauses in the middle of an utterance. For example, North American phone numbers are usually spoken with a pause after the three-digit area code and another pause after the three-digit exchange.

15.2.3 Evaluative Usability Testing

Evaluative usability testing is performed before the system is released for pilot testing. The general approach is the same as outlined for the iterative usability testing in Chapter 8 (in the detailed design phase) except that evaluative usability testing is run with the complete working system.

This approach allows you to detect problems with the pacing of the system. For example, you can find out whether there are delays or latencies that cause callers frustration or confusion. At this time you will evaluate the appropriate success criteria (those that can be measured on usability data) that were specified during requirements definition. This will help to determine whether the system is ready for pilot.

15.3 Tuning

After testing is complete, you are ready to deploy the system for your target user population. This is typically done in two phases. The first phase is referred to as the **pilot phase**, in which the system is released to a limited number of users (a few hundred to a few thousand) in the actual caller population. Data are collected, and the system is tuned using these data. Often, as many as three iterations of data collection and tuning may happen during the pilot phase. When the pilot phase is complete, the system is rolled out to the entire user base. Tuning may continue with data collected following the full rollout.

The pilot is the first opportunity to evaluate the system with real in-service datadata from real callers performing real tasks. Such data are essential for tuning recognition performance and grammar coverage. There is no way to evaluate the great variety of ways people will talk to the system until you collect data from users engaged in real, task-oriented behavior, performing tasks that matter to them.

The same is true for evaluation of the dialog design. Users will not respond in exactly the same way in an experimental situation (such as when performing an assigned task in a usability study) as they will when they are dealing with their own money or booking a real flight. In the latter, the difference between flying to Boston or Austin matters a great deal! In general, users do not exhibit the same impatience with obvious inefficiencies when they are experimental subjects as they do when they call a company trying to accomplish a specific goal.

On the other hand, there are important advantages to usability studies. In particular, the ability to talk with participants about their experience and the reasons for any problems they might have had will help you identify the root cause behind observed problems. When evaluating in-service data, despite the advantage of the authenticity of the data, you can only guess at what was going on in the callers' minds when they experienced problems. As a result, we recommend a combination of usability studies (during design and just before pilot) and analysis of in-service data, as described here.

To support the tuning process, you collect data from all calls to the pilot system, every input utterance from every caller. The data are transcribed; the actual word strings spoken by the callers are notated. These transcriptions can then be used to measure recognition accuracy and to tune grammars. In later phases, such as during tuning after full deployment, you may decide to sample a subset of call data for tuning.

Three aspects of the system are tuned: dialog, recognition, and grammar. You must analyze observed problems to figure out the root cause. For example, if a high out-of-grammar rate is observed in one particular dialog state, the cause may be missing items in the grammar, or it may be a poorly worded prompt that is leading to confusion or a poorly designed dialog strategy that leads callers into the wrong state. Poor recognition may be caused by pronunciations that are missing from the dictionary, poorly tuned endpointing parameters, or a variety of other problems.

This chapter covers dialog tuning and recognition tuning, and Chapter 16 covers grammar tuning.

15.3.1 Dialog Tuning

There are three primary approaches to dialog tuning: call monitoring, call log analysis, and user experience research. Let's look at each approach.

Call Monitoring

An essential step in understanding the performance of a system is to monitor calls. It can be extremely instructive to take the time to listen to 100 randomly selected calls. There are a number of ways to do that, depending on the platform and tools that are available.

In some cases, you can actually listen while calls are taking place. In other cases, you can make **whole-call recordings**: digital recordings of both sides of a call (system prompts and caller utterances) that can be played back later. If neither of these approaches is available, you can use software to re-create calls by playing back all the collected utterances from a call, interspersed with the appropriate system prompts.

The advantage of monitoring or whole-call recording is that, given the ability to hear both sides of the conversation continuously, you can more easily diagnose problems with barge-in or endpointing. However, you can also identify many dialog issues from re-created calls. In the early days of Nuance, one of the authors (MC) spent all his commutes to and from work listening to recordings of calls to deployed systems. As the traffic problems in Silicon Valley worsened, our performance improved.

When listening to re-creations of calls or recordings, you can be selective about which calls to review based on

various criteria. As mentioned, you should start by listening to randomly selected calls to get a sense of how the system is doing. You can then select calls with lots of problems, calls with problems in particular areas, or calls that exercised particular features. Be careful not to choose simply "the 100 calls with the most problems" because you will likely get only the calls with babies screaming and dogs barking in the background.

Earlier, we discussed the trade-offs between in-service data and usability data: the representative nature of real use versus the ability to probe the subject to gain insight into the reasons for problems. There is a call monitoring approach that combines the advantages of both. It consists of **callbacks** to users right after they have made a call to the application. Using a callback technique, you can combine the realism of in-service data with the ability to interview the caller. This approach can be very valuable, especially if it can be focused on callers who experience a particular problem that you are trying to understand.

Call Log Analysis

Most platforms allow the collection of **call logs**: data records about the performance of each call to the system. You can make a number of performance measurements from call logs that are useful in identifying dialog problem areas.

A **task completion analysis** looks at all tasks that callers attempt to perform, measuring how often they successfully complete the task and noting the reason for failure when they do not complete the task. Tasks with high noncompletion rates are candidates for more detailed analysis.

A **dropout analysis** looks at where in the dialog calls end, whether by hang-ups or by requests to transfer to a live agent. States with a high dropout rate that are not normally associated with logical places to end a call are candidates for further analysis.

A **hotspot analysis** identifies those dialog states with a high rate of problems. A "problem" can be defined in a number of ways. For example, you may define a problem as all cases when the recognizer returned something other than a recognition result (e.g., rejects or timeouts) plus requests for help. You can even run a number of hotspot analyses using different criteria.

Once a problem area is identified, you need to find the root cause. Listening to the interaction in only that dialog state for a number of calls may shed light on the problem. It may also be useful to begin listening at a point a couple of interchanges before the problematic state. You can also look at a long list of transcriptions of things callers said when in that state.

Looking at misrecognitions and out-of-grammar rates may also be useful. If the out-of-grammar rate is high, see whether it is dominated by in-domain (reasonable answers that were not covered by the grammar) or out-of-domain (inappropriate answers) instances of out-of-grammar utterances. A high rate of out-of-domain out-of-grammars may indicate a confusing prompt or dialog strategy, whereas many in-domain out-of-grammars may be a stronger indicator that certain items are missing from the grammar.

You can use pilot data to tune error recovery approaches. Review every state with more than a minimal number of rejects or timeouts. For a number of the problem calls, listen to the relevant dialog state (or start a couple of states earlier for context). Often, the predominant causes of

problems in that state will become clear, and you can tune the error recovery messages (or even the main prompt for the state) accordingly.

User Experience Research

User experience research is focused on getting qualitative input about the performance of the system. A common approach is to send a survey to callers who used the system during the pilot period. On a number of projects, we have tacked a telephone survey onto the end of the application. When callers completed their task, they were asked whether they were willing to participate in a brief survey. The survey was automated. They were asked a few questions in a directed dialog that used the speech system to understand and tabulate their answers and then asked for open-ended feedback, which was recorded for later transcription. Interviews, typically over the telephone, can also be used to get qualitative input about the experience callers are having with the system.

Longitudinal studies are sometimes performed for systems that expect lots of repeat callers. These studies are designed to track the usage experience of individual users over time. Longitudinal studies may use periodic surveys or interviews, correlated with usage and performance data tracked over time for the participating users. You should use performance data to help choose appropriate participants in a longitudinal study. For example, you can purposely include callers who have used particular features, have neglected particular features, or have stopped using the system after one or two tries.

15.3.2 Recognition Tuning

The goal of recognition tuning is to optimize the accuracy of recognition for every grammar. Typically, each dialog state has its own grammar, although some grammars (e.g., yes/no grammars) may be shared among a number of dialog states.

The first step in recognition tuning is to measure the performance for every grammar. You measure recognition accuracy by comparing the transcriptions to the recognizer output for each utterance. The data are typically divided into in-grammar and out-of-grammar sets. For the in-grammar data, you measure the correct-accept, false-accept, and false-reject rates (these are defined in Chapter 13). For the out-of-grammar data, you measure the correct-reject and false-accept rates. Additionally, you measure recognition latency for each grammar. Latency is measured as the time between the end of the caller's utterance and the beginning of the next prompt played by the system. Excessive latency may indicate that the system is underprovisioned; you may need to add more servers.

Efforts to improve recognition performance are typically focused on a number of parameters that control the recognition process. Most commercially deployed recognition engines have some recognition parameters the developer can set, including the following:

- **Rejection threshold:** This is the confidence level below which the recognizer will return a <reject> rather than a recognition result. As the rejection threshold is lowered, the recognizer will reject less often, thereby lowering the false-reject rate (the rejection of utterances that should have been accepted). However, at the same time the false-reject rate is lowered, the false-accept rate is raised (the choice, by the recognizer, of an incorrect recognition

hypothesis). You must choose an appropriate trade-off between false accept and false reject.

- **Start-of-speech timeout:** This is the maximum length of time the endpointer will listen for the beginning of speech before timing out. The longer the start-of-speech timeout, the longer the endpointer will listen. As the start-of-speech timeout gets longer, the system is less likely to miss a caller's response, but the amount of time waited gets longer before the system realizes the caller is not responding and takes appropriate action.

- **End-of-speech timeout:** This is the length of silence the endpointer will listen to, after detecting speech, to decide that the caller has finished speaking. As the end-of-speech timeout gets longer, the system is less likely to mistakenly believe callers are finished speaking when, in fact, they are only pausing. However, the longer the end-of-speech timeout, the greater the delay between the end of caller speech and the system response.

- **Pruning threshold:** Most commercial recognizers use some sort of pruning approach to limit the recognition search as it progresses so that it stops pursuing paths through the recognition model that are very unlikely to end up as the best-matching path when the search is finished. This speeds recognition, at the cost of occasionally introducing an error. (An error can be introduced if a path that seemed unlikely early in the search is actually the best path when the entire utterance is considered.) As the pruning threshold gets higher (i.e., fewer paths are pruned out), recognition accuracy improves, at the cost of more time (greater computational cost for recognition).

There may be other parameters associated with the recognition engines from different vendors, such as those used to optimize the choice of acoustic models or the recognition search. We do not cover such vendor-specific details here. They are covered in each vendor's documentation.

To optimize the value of recognition parameters, you run a series of recognition experiments in batch mode. Most commercial recognition vendors provide a batch mode in which you can supply the system with a set of utterances, along with their transcriptions, plus a set of parameter settings (e.g., the reject threshold). The system then runs the entire set of utterances through the recognizer, compares the recognized word strings to the transcriptions, and reports the results. By running a number of experiments for each parameter to be tuned, testing a series of values for the parameter, you can choose the best trade-off for recognition performance and behavior of your application.

When recognition performance is lower than expected, you can make a number of adjustments in addition to the tuning of recognition parameters. Let's discuss the two most common: dictionary tuning and grammar probabilities.

Dictionary Tuning

If specific words are consistently missed by the recognizer, it is worth checking the dictionary pronunciation and listening to the data to make sure the pronunciations in the dictionary cover the pronunciations callers are using. Most vendors provide a means of adding pronunciations to the dictionary. Their documentation should describe the

phonetic alphabet they use for specifying dictionary pronunciations.

Some application grammars include very long lists of itemsfor example, lists of company names, person names, street names, city names, and so on. Some words, especially company names, may be made-up words (e.g., "Microsoft" was not a word before the company was founded) and have nonstandard pronunciations. Many person names are foreign and therefore use different pronunciation rules than those of standard North American English. In both cases, it may be necessary to take the time to listen to the data and update the dictionary to accurately reflect the most common pronunciations being used.

Grammar Probabilities

Recognition accuracy for grammars that include long lists, such as the examples in the preceding section, can often be improved by embedding probabilities in the grammar. The probabilities represent the a priori probability that callers will say each of the items in the list.

For example, suppose a stock quote grammar includes 15,000 names of companies, indexes, and mutual funds. Users ask about some companies far more often than others. Embedding appropriate probabilities in the grammar can significantly improve recognition performance. Of course, if the probabilities are wrong, they will do more harm than good. The best way to obtain accurate probabilities is to estimate them based on a large set of data, such as the data collected during a pilot or from a deployed system. Probabilities for certain types of grammars (e.g., stocks) may change over time and therefore must be updated periodically. Most commercial

recognizers facilitate probabilities, even for rule-based grammars.

15.4 Conclusion

We have examined the standard approaches used for development, testing, and tuning of applications. Some of these steps (e.g., dialog tuning) are typically performed by the VUI designer. Other steps are often performed by other team members, especially in large organizations.

In Chapters 16 and 17, we discuss grammar development and voice coaching in depth. These two activities are unique to the design of voice user interfaces. In both, the designer often plays the lead role.

Chapter 16. Creating Grammars

The development of grammars is a key part of the creation of effective voice user interfaces. Grammar development goes hand-in-hand with the choice of dialog strategies and the crafting of prompts. The interacting goals of grammar development and detailed design are as follows:

- To create a grammar that accommodates, as much as possible, what real callers will say
- To craft the call flow, dialog strategies, and prompts in order to guide the caller to stay within the grammar

Clearly, these goals are interrelated. The grammar definition step described in Chapter 8 (definition of the slots and sample phrases for the grammars for each dialog state) is one part of achieving these goals. In general, the grammar writer and the dialog designer must collaborate. A lack of communication can lead to poor solutions to design challenges. For example, if you observe, during the tuning phase, a high out-of-grammar rate, it may imply the need to extend the grammar. Alternatively, it may imply the need to write a prompt that provides the caller greater clarity about what can be said.

A key skill for grammar writing is the ability to imagine the variety of ways callers will express themselves. This skill improves with experience and benefits from linguistic knowledge. However, neither knowledge nor experience, nor even great skill at introspection about your own language use, is enough to ensure high coverage. Grammar development must include the opportunity to extend and

refine grammars based on data from real calls to a working system.

Keep in mind that adding new paths to grammars so that they accommodate a greater range of caller input creates new recognition challenges; for example, there will be new opportunities for ambiguities and confusions during the recognition search. Therefore, the goal is to maximize coverage and minimize overcoveragein other words, to maximize inclusion in the grammar of word strings (words, phrases, and sentences) callers are likely to say, while minimizing inclusion of word strings rarely or never spoken. Only analysis of real caller data can ensure that you will accomplish this goal.

Earlier sections of the book discuss two primary approaches to grammar: rule-based and statistical. Rule-based grammars involve an explicit definition of all word strings, whereas statistical grammars involve learning the probabilities of the various possible word strings from data. Keep in mind that grammars, as discussed here, include both a **syntactic** component (a description of possible word strings) and a semantic component (a description of the meaning associated with each word string). In rule-based grammars, the syntactic and semantic definitions are combined. Natural language interpretation directives are embedded within the grammar rules. In contrast, with statistical grammars, the definition of the syntax is usually separate from the definition of the semantics. One (or more than one) file defines the statistics of possible word strings. This is the grammar that defines the search space for recognition. A separate file, used for natural language understanding, defines the mapping from word strings to meanings. The technique for defining the semantics may itself be either rule-based or statistical. These variations are discussed later.

Before we dive into the details of grammar creation, there is one point worth reiterating. As you create a voice user interface, there are many times throughout the entire process when you may refine prompt wording, sometimes in subtle ways. Beyond detailed design, you may refine prompts during tuning. Sometimes you will even refine prompts during the recording session, when the voice coach hears something that doesn't quite work. Whenever a prompt is changed, there are two constituencies who must be notified: the grammar developer (because even subtle changes in wording may imply grammar changes or additions) and the prompt designer (to make sure that the change accords with the larger context in which the design was created, such as considerations of consistency in terminology and persona).

The following sections cover the development, testing, and tuning phases of grammar creation. In each section, we discuss the process for the rule-based and the statistical grammar approach.

16.1 Grammar Development

The process of grammar development is substantially different for rule-based than for statistical approaches. This section covers the development of rule-based grammars, statistical language models for recognition, and two approaches for developing natural language grammars for systems that use SLMs for recognition.

16.1.1 Developing Rule-Based Grammars

The primary challenge for the developer of a rule-based grammar is to anticipate what callers will say in each dialog state. To begin, you look at the grammar definition for the dialog state you are ready to work on. The designer will have specified two things: the set of information items, or **slots**, to be returned by the grammar; and a set of sample expressions (described in Chapter 8).

The slots will tell you about the **core** of the grammar: items to include that actually carry the salient information. For example, one slot for a travel application might be `destination-city`. This tells you that city names will be included in the grammar and that they represent part of the core, or meaning-bearing, elements in the grammar. It also tells you that the information about the city chosen by the caller will be communicated to the application in a slot named `destination-city`.

The sample expressions supplied by the designer will give you information about both the core and the fillers. **Fillers** are the words and phrases that may surround the core. If

one sample expression is "I want to go to the Big Apple," you know that "the Big Apple," as a synonym for New York, is part of the core grammar and that "I want to go to" is an example of the kind of filler that can be expected. Obviously, all the sample expressions should be covered by the grammar. However, they are just meant to be examples. You should flesh out the grammar with appropriate related core and filler items.

Another important source of information about the expected caller utterances is the prompt wording itself. Callers often choose the same wording as the prompt. For example, if the prompt says, "Where are you going?" callers are likely to give an answer such as, "I am going to San Francisco," whereas if the prompt says, "What's the destination city?" an answer such as, "My destination is San Francisco" is more likely.

A number of languages have been developed for specifying grammars, and all of them have roughly the same capabilities. For the examples presented here, we use Grammar Specification Language (GSL), a language created by Nuance and available on a number of platforms.

Table 16-1. Grammar Operators for GSL

OPERATOR	EXPRESSION	MEANING
( ) concatenation	(A B C ... D)	A followed by B followed by C followed by ... D

OPERATOR	EXPRESSION	MEANING
[ ] disjunction	[A B C … D]	One of A or B or C or … D
? optional	?A	A is optional
+ positive closure	+A	A occurs one or more times
* Kleene closure	*A	A occurs zero or more times

A GSL grammar is a set of rules of the form:

GrammarName *GrammarDescription*

A grammar description uses the operators defined in Table 16-1 to combine basic grammar elements. The elements, or **operands**, are either lowercase strings (representing the actual words in the grammar) or strings that include uppercase characters, which represent subgrammars. Subgrammars are themselves grammar rules defined elsewhere.

To make it more concrete, let's look at an example of a simple grammar for the travel application mentioned earlierspecifically, the grammar for the `GetDestination` dialog state. The first grammar name is `.GETDESTINATION`. The dot operator (".") at the beginning indicates that it is

the top-level grammar, the grammar name referenced in the application:

```
.GETDESTINATION (?PREFILLER CITY ?POSTFILLER)


PREFILLER [(i want to go to)
           (i am going to)
           (i need a flight to)
           (?i'm going to)
          ]


CITY      [[(new york) (the big apple)]
           (san francisco)
           boston
          ]


POSTFILLER [please]
```

The top-level grammar, `.GETDESTINATION`, is defined as the concatenation of three subgrammars: `PREFILLER`, `CITY`, and `POSTFILLER`. Two of the subgrammars`PREFILLER` and `POSTFILLER`are defined as optional; the `?` operator means

that grammatical word strings may or may not have such fillers.

This grammar allows inputs such as, "I want to go to New York, please," "I'm going to San Francisco," "Going to Boston," and "Boston." The CITY subgrammar is the core. Isolating it in a separate module as a subgrammar makes it easy to updatefor example, to add more cities. As a separate module, it is also easy to reuse. You could, for example, reuse it in another dialog state as part of a grammar for getting the traveler's origin city.

One thing missing from the example is the semantic specification: the instructions for filling slots given the caller's path through the grammar. A simple approach is shown in the following example:

```
.GETDESTINATION (?PREFILLER CITY ?POSTFILLER)



PREFILLER [(i want to go to)

           (i am going to)

           (i need a flight to)

           (?i'm going to)

          ]



CITY      [[(new york) (the big apple)]
{<destination-city ny>}

           (san francisco)
```

```
{<destination-city sf>}

           boston                    {<destination-
city boston>}

          ]


POSTFILLER [please]
```

The slot-filling commands (e.g., `<destination-city ny>`) are executed if the preceding grammar construct is traversed. In this example, if the caller said, "I want to go to New York," "Going to the Big Apple," "I need a flight to New York," or "New York," the `destination-city` slot will be filled with the value ny, indicating to the application the intended meaning, despite the variety of ways the caller may have expressed that meaning.

An alternative approach, for a different but related grammar, is shown in the following example. In this case, the subgrammar returns values to the higher-level grammar that referenced it, rather than directly filling slots. This approach is useful if the subgrammar is to be used multiple times by the higher-level grammar to fill multiple slots, as is the case in this example.

```
.GETCITIES  (?PREFILLER

             [(from CITY:orig to CITY:dest)

              (to CITY:dest from CITY:orig)

             ] {<origin-city $orig> <destination-
city $dest>}
```

```
             ?POSTFILLER

            )


PREFILLER   [(i want to go)

             (i am going)

             (i need a flight)

             (?i'm going)

            ]


CITY        [[(new york) (the big apple)]
{return(ny)}

             (san francisco)              {return
(sf)}

             boston                       {return
(boston)}

            ]


POSTFILLER  [please]
```

This grammar accepts inputs such as, "I want to go from New York to San Francisco" or "I'm going to Boston from

New York." Rather than directly fill slots, the `CITY` subgrammar returns values. Those values get assigned to the variables `dest` and `orig` (by the expressions `CITY:dest` and `CITY:orig`), which are then referenced, as appropriate, by the slot-filling commands. For example, `<origin-city $orig>` causes the `origin-city` slot to be filled with the value of the variable `orig`. Using this approach makes it possible for the higher-level grammar `.GETCITIES` to use the `CITY` subgrammar in two places, returning values to be used in two different slots.

In general, grammars are tuned and refined iteratively, and therefore they should be organized for easy maintenance. Here are some guidelines:

- Break the grammar into a logical, modular structure of subgrammars.
- Place the core and the fillers in separate subgrammars.
- Choose descriptive names for subgrammars, slots, and variables.
- Format the grammars to make the structure obvious. For example, use a clear indenting scheme to offset logical groupings within the grammar.

In some cases, the grammar cannot be fully specified before runtime. For example, imagine an application for paying bills. Each subscriber to the service will select the companies to be paid. When subscribers call the system, their specific list of companies must get loaded into a grammar. This is called a **dynamic grammar**. Each speech technology vendor has its own approach for handling

dynamic grammars. Therefore, we do not cover it here; see the vendor-specific documentation for details.

16.1.2 Developing Grammars for Statistical Language Models

Statistical language models (SLMs) are used when the amount of expected variation in spoken inputs is hard to capture with explicit grammar rules. The basic approach is to automatically learn what word strings occur, and with what likelihood, from real caller data. In this way, the grammar developer no longer needs to imagine all the variations. Instead, you need only collect a data set, transcribe it, and feed it to the software utility that creates the SLM.

The creation of the statistical language model is referred to as **training** the language model. The data set used by the training utility, consisting of a list of transcriptions of caller utterances (the actual word strings spoken), is called the **training set**. The basic approach used by the training utility is to estimate the probability of the occurrence of each word in the vocabulary, given its context. The context used for these estimates is the most recent few words spoken.

The **order** of the model determines how much context is considered. An Nth order model (called an **N-gram**) considers the N 1 (N minus 1) predecessor words as context. In other words, the model provides an estimate of the probabilities of what the next word to be spoken may be, given the previous N 1 words. A first-order N-gram, called a **unigram**, consists of estimates that a word will occur, without regard to context. A second-order N-gram is called a **bigram**. It consists of estimates of word

occurrence, given the most recent predecessor word. A **trigram**, or third-order N-gram, provides estimates given the most recent two preceding words.

Theoretically, the higher the order of the model, the more predictive power you will have about which word will occur next. However, the higher the order of the model, the more training data we need to come up with reliable estimates, given the larger number of probabilities that we must estimate. Assuming a vocabulary size of 1,000 words, a unigram model consists of 1,000 probabilities. By contrast, a bigram model consists of $1{,}000^2$ probabilities (i.e., the probability of each of the 1,000 words in the vocabulary, given each of the 1,000 possible predecessor words).

In typical applications that use SLMs, a vocabulary size of a few thousand words is common. Trigram models are often used. A rule of thumb about the size of the training set for such a model is a minimum of 20,000 transcribed utterances.

After an application goes to pilot, it is easy to collect and transcribe data, thereby increasing the size of the training set. The challenge with SLM-based systems is **bootstrapping**: creating an initial model so that you can achieve reasonable performance at the start of the pilot. There are a number of approaches you can use to create the initial model. One approach is to use a Wizard of Oz system to collect data (see Chapter 8). If the wizard is equipped to help callers complete their task, you can handle real callers. Otherwise, you need to solicit callers to use the system and give them tasks to complete.

Alternatively, if there are live agents currently handling the task, you can use them as wizards. The caller will hear the recorded prompts played by the system, but the

"recognition" and "understanding" will be performed by the live agents.

Another possibility is to develop a GSL-based grammar for the first phase of the pilot. In that case, you should still use the prompting targeted for the SLM system, even though the wording of the prompts may encourage language from the callers that leads to a high out-of-grammar rate for the GSL. When a recognition reject occurs, the back-off prompting should be more appropriately constraining, to help the caller succeed with the GSL grammars. In that way, you will be able to collect data, especially from the first interchange with the system, that will be useful for training the SLM. However, when there is a problem, the system will quickly back off to a directed dialog, and in this way the caller is unlikely to experience more than one reject due to the data collection setup. As soon as enough data have been collected, the GSL grammar can be replaced with an SLM.

The SLMs we have discussed in this section fulfill the role of the syntactic side of the grammar. They are used to create the search space for the recognizer (the recognition model described in Chapter 2), resulting in the recognition of word strings. The next two sections describe the two approaches commonly used for the semantic role of the grammar. These are the methods for assigning a meaning to the word strings recognized by an SLM-based recognizer.

16.1.3 Developing Robust Natural Language Grammars

In the late 1980s and early 1990s, researchers at a number of sites worked on projects that combined speech recognition technology and natural language understanding

technology, resulting in the early spoken language understanding systems (Cohen, Rivlin, and Bratt 1995). Before those projects, most natural language understanding research was applied to text rather than speech.

One of the lessons of these first applications of natural language technology to spoken language was that spoken and written language differ dramatically. Beyond the structural and word-choice differences discussed in Chapter 10, spoken language is different from written language in that it is often "ungrammatical," includes disfluencies (e.g., "Um, I want the secno, the third flight"), and often includes extraneous information that is not directly needed to answer a particular question ("I have a meeting in the afternoon, so I want to arrive by eleven a.m.").

Robust parsing approaches were developed to deal with these problems (Jackson et al. 1991). The basic idea is to search for meaning-bearing words and phrases without trying to parse and understand the entire word string that was spoken. For example, consider the following dialog:

SYSTEM: What time do you want to arrive?

CALLER: I have a meeting in the afternoon, so I want to arrive by eleven a.m.

Here, a robust grammar might search only for phrases that specify a time, disregarding everything else. The grammar specification would include phrase-grammars with slot-filling commands, but no grammars to cover fillers. In this example, the grammar need only cover "eleven AM." There is no need to write a grammar that can cover all the other, hard-to-predict material.

As the language you must interpret becomes more variable and flexible (as with applications using SLMs), the ability to write only the slot-filling phrase-grammars, disregarding everything else, is a tremendous simplification of what would otherwise be a daunting grammar-writing task. When applications accommodate the expression of multiple slot values in a single utterance, not only the surrounding fillers but also the actual order in which slot values get expressed may vary. Luckily, when writing a robust natural language grammar, you need not specify the order in which the slot-filling phrases happen. As a result, a very simple grammar could cover variations such as the following:

> *"I want to get a flight from Boston to San Francisco."*
>
> *"I need to get to San Francisco right away, starting from Boston."*
>
> *"I have a very important meeting, and I need the next flight from Boston to San Francisco."*
>
> *"Tomorrow is my aunt's birthday party in San Francisco. When's the next flight from Boston?"*

The method for developing phrase grammars for robust parsing is the same as that described earlier for rule-based grammars, but simpler. The specification syntax is usually

similar, although you will have to look at vendor-specific documentation for the details.

16.1.4 Developing Statistical Natural Language Grammars

Sometimes, SLM-based grammars are used to fill a single slot. One example is call routing, discussed in earlier chapters. In call routing, a tremendous variation is expected in the ways callers express their needs, but the result is the filling of a single slot: the identity of the service to which the system must forward the call.

Some speech technology vendors let you use a statistical approach to create the natural language grammar (the slot-filling grammar) for such single-slot applications. The result is that there is no need to develop handwritten grammar rules. Instead, you simply collect data for a training set. Then a training utility automatically learns, from the data, appropriate mappings from words, phrases, and combinations of words and phrases to the appropriate slot value. This capability makes it far easier to develop call routing applications, given the difficulty of writing the slot-filling rules by hand for input that has so much natural variation. In effect, all the knowledge to be encapsulated in the grammar is learned automatically.

To use a utility for statistical natural language training, you need a training set. Typically, the same training set used for the SLM is used for training the natural language grammar. However, in addition to transcriptions, each utterance must be labeled with the appropriate slot value (e.g., the appropriate route, given the caller request). You can collect data appropriate for natural language training by using the

same methods for collecting data discussed in section 16.1.2.

The specific details of developing such systems are vendor-specific. You should consult the vendor directly.

16.2 Grammar Testing

Grammars can be quite complex. In the same way that complex software can have bugs and must be tested, grammars should be tested before release for a pilot. The purpose of the testing is both to fix bugs and to ensure reasonable performance out of the box.

16.2.1 Testing Rule-Based Grammars

There are six standard tests to run on rule-based grammars:

1. Coverage
2. Overcoverage
3. Natural language
4. Ambiguity
5. Spelling
6. Pronunciation

Speech technology vendors provide a variety of tools to accommodate these tests. For the specific details, see the vendor documentation. Here, we describe each test in a general way and explain how to use it.

Coverage Testing

A **coverage test** uses a test suite consisting of phrases and sentences the grammar is designed to cover and makes sure that all of them are covered. Typically, you build up the test suite while developing the grammar. Start with the sample grammar expressions supplied for each dialog state during detailed design. Before writing any grammar rules, supplement the test suite with items you know you intend to cover in the grammar. As you write the grammar, add more items to the test suite to make sure that the various grammar paths are adequately exercised.

When the grammar is complete, run the test suite and make sure all items are parsed by the grammar. Whenever the grammar is changed, you can rerun the test suite to make sure that bugs have not been introduced. When enhancing the grammar, such as during the tuning phase, you can add items to the test suite to cover the grammar paths you are adding.

The primary purpose of the coverage test is to find bugs as well as to ensure a certain minimum level of coverage going into the pilot. The ultimate refinement of the grammar to maximize coverage will happen in the tuning phase, based on actual spoken input from real callers.

Overcoverage Testing

Overcoverage refers to sentences or phrases the grammar covers unintentionally, such as nonsense phrases that the grammar parses either because of a bug in the grammar or as a side effect of combining various partial intended paths through the grammar. Overcoverage may, in some cases, be an indication of a bug in a written grammar. Even if it is not due to bugs, excessive overcoverage makes recognition harder. There are more

possible grammar paths that can lead to ambiguities and confusions during the recognition search.

In some cases, a certain amount of overcoverage either is unavoidable or actually makes sense when traded off against increased grammar-writing complexity. For example, a sentence such as, "Please get me a flight to San Francisco, please" may be very unlikely, but it may end up as a legal sentence in a grammar because the prefiller and postfiller subgrammars are separate. They cannot therefore restrict their paths based on the path that was taken in the other subgrammar. Although it is possible to write the grammar in a way that will avoid this problem, it might add significant complexity and might not be worthwhile.

The standard way to test for overcoverage in a grammar is to generate strings from the grammar using a utility that simply enumerates valid word strings that the grammar parses. In relatively small, finite grammars, you can generate all possible strings. Most grammars are too complex to generate all possible strings. In fact, many grammars can represent an infinite number of possible stringsfor example, those using the + or * operator or recursive grammars in which a grammar can refer to itself as a subgrammar or to other subgrammars that ultimately refer back to it. For large or infinite grammars, you can randomly generate a set of strings from the grammar (around 100 is a reasonable number). In either case, after the strings are generated, you read through them. When you find strings that are unintended (constitute overcoverage), you should either fix them or make the decision to accept them.

Natural Language Testing

The purpose of the natural language test is to make sure that the correct meaning is assigned to input utterances (e.g., in the case of GSL, that each slot is assigned the correct value). Typically, you use the same test suite for natural language testing that you used for coverage testing. You run a utility that returns the meaning for each input string in the test set. The first time the test is run, you must check the results by eye to make sure that the meanings are correct. If they are, you should save a copy of the utility's output so that in the future, when rerunning the test (e.g., after a change to the grammar), you can automatically compare the new test result to the old one. In this way, you can make sure that nothing has changed, meaning that the change has not introduced natural language understanding bugs.

Ambiguity Testing

An ambiguity is a case in which there are multiple possible natural language interpretations of an input. Vendors provide utilities that search for ambiguities in grammars. Often, an ambiguity is caused by a bug in the grammar, which you must fix. In some cases, the ambiguity reflects actual ambiguity in the application domain (e.g., the same input utterance may be spoken whether the caller wants a stock quote for Cisco Systems or for Sysco Foods). Any ambiguity that is not fixed in the grammar must be explicitly handled in the application (e.g., adding the prompt, "Did you want Cisco Systems or Sysco Foods?").

Spelling Testing

A misspelled word in a grammar can cause degradation in recognition performance that can be very hard to track

down. For example, a misspelled word can lead to an automatically generated pronunciation that is wrong, making recognition difficult. This is especially problematic in applications that have grammars that include long lists, such as a travel application that handles travel between 2,000 cities. If five of those cities are misspelled, leading to poor pronunciation models that cause a slight degradation in recognition performance, the problem will be very hard to detect and will not be detected until you are tuning pilot data, if at all. Related problems can also be caused by misspellings in values used in slot-filling commands.

A simple check with a standard spell checker can detect many of these problems. Of course, some words in grammars will not be in the spell check dictionary (e.g., company names that are not standard words). Therefore, the list of misspellings should be reviewed, and only those that are mistakes should be fixed.

Pronunciation Testing

Pronunciation testing is useful for applications that have long lists of itemssuch as company names, city names, street names, and so onthat may not be in the initial dictionary and are added either by hand or with automatic pronunciation facilities. A pronunciation test uses a facility that lets you test a grammar by speaking to it. Simply go down the list saying each item. For any misrecognitions, take a look at the dictionary pronunciation. An alternative approach is to feed the pronunciations of each word in the list to a text-to-speech engine and listen to the result.

16.2.2 Testing Statistical Language Models

Given that statistical language models are automatically trained from data, they are far less prone to bugs than rule-based models. However, before pilot, they should be tested to make sure they provide reasonable out-of-the-box recognition accuracy.

When you test statistical language models, the most important concept is that the data used for the test must not be part of the training set. To perform a valid test you must use separate data. Otherwise, you are likely to end up with unrealistically optimistic results that will not reflect real-world performance.

When you collect data to train the initial SLM, some of the data (e.g., 2,000 utterances) should be held out of the training set and used as a **test set** to measure the performance of the model. You can assess the performance of the model either by running recognition on the test set or by measuring **perplexity**: a measure of the predictive power of a model. The lower the perplexity, the greater the predictive power. Greater predictive power is likely to lead to more accurate recognition performance. (Utilities to measure perplexity, given a test set, are provided by speech recognition vendors that offer SLMs.) It is often most straightforward to assess SLMs based on recognition tests. Most vendors provide utilities for running recognition in batch modethat is, from a list of prerecorded utterances.

16.2.3 Testing Robust Natural Language Grammars

Testing robust natural language grammars is very similar to testing standard rule-based grammars. The only difference is that the former are simpler; they include only the slot-filling phrase-grammars. Therefore, you need not worry

about coverage of the many possible filler and other extraneous words and phrases that callers may use. All the tests listed for rule-based grammars can be used for robust natural language grammars.

16.2.4 Testing Statistical Natural Language Grammars

A statistical natural language grammar is generally tested for accuracy. You want to know how accurately it assigns the correct slot value (e.g., the correct destination service for a call router) to input word strings. To perform a valid test, you need a test set that is not part of the training set. The same test set used for testing an SLM can be used for testing the statistical natural language grammar. In addition to transcriptions, each item in the test set should be labeled with the correct slot value. You measure performance by running the statistical natural language engine on the test set.

16.3 Grammar Tuning

Chapter 15 discusses the importance of pilot data in assessing and tuning the performance of the system. The pilot is the first opportunity to collect real, in-service data. Such data are just as important for tuning grammar coverage as for tuning recognition accuracy and dialog performance. People will choose different words to express themselves when they are engaged in real, task-oriented behavior.

16.3.1 Tuning Rule-Based Grammars

The same data that are collected and transcribed for tuning recognition accuracy and dialog performance can be used to tune grammar coverage. The utility described earlier, for running the coverage test of a rule-based grammar, can be used on the transcriptions of the pilot (or deployment) data set. The result will be a list of those utterances that are handled by the grammar (in-grammar) and those that are not handled (out-of-grammar).

You can evaluate the out-of-grammar utterances to find candidates for addition to the grammar, as well as indications of general refinements that are needed. Each out-of-grammar utterance should be evaluated. In general, those phrases and sentences that happen often and are missing from the grammar are the most important candidates for addition. Even a single observation, though, may be enough to warrant inclusion if it makes obvious sense for the grammar.

Out-of-grammar examples should be considered as candidates for addition to the grammar if they are in-

domainthat is, the caller is appropriately responding to the system prompt. They should not contain a substantial amount of extraneous material that is unrelated to the application. For example, in response to a prompt asking for a desired travel date, "My aunt Gertrude is having her sixtieth birthday party next Tuesday, so I would like to fly on Monday" would not be reasonable to try to cover in the grammar, although it is in-domain (it answers the question asked in the prompt). On the other hand, it would be reasonable to add the phrase "I would like to fly on Monday," especially if it happens numerous times.

Each example that is added to the grammar should be considered for generalization. For example, if you add "I want to go in the afternoon," you should also consider "morning" and "evening."

If out-of-domain examples happen repeatedly, it may be an indication that the prompt is unclear, that callers are ending up in that dialog state by mistake, or that the notion of what is in-domain needs to be reconsidered. In general, there is a gray area between clearly in-domain and clearly out-of-domain utterances. If you have out-of-domain utterances in that gray area that happen repeatedly, you should reconsider what is appropriate to consider in-domain and cover in the grammar.

After the grammar is rewritten to expand the coverage, you should rerun the original set of tests, using the original test suite, to make sure that bugs have not been introduced. This is referred to as **regression testing**. Once the tests are passed, the test suite can be fleshed out to cover new paths that have been added to the grammar. The new test suite should then be run and, after it passes, the results saved for future regression testing.

16.3.2 Tuning Statistical Language Models

With each phase of pilot testing and early deployment, the new data sets of transcribed utterances should be added to the training set used for SLM training. In general, the larger the training set, the better the probability estimates. Furthermore, data from the working system may be more representative of speech from the real caller population than the data used for bootstrapping the SLM.

Hold out some of the new data to add to the test set. If the original bootstrap data were less realistic than the new data set (e.g., based on laboratory-style data collection rather than real in-service data), you may want to replace the old test set. After a new SLM is trained, you can measure recognition accuracy with both the old and the new models to make sure performance is improving.

16.3.3 Tuning Robust Natural Language Grammars

Tuning a robust natural language grammar is similar to tuning a standard rule-based grammar, except that it is simpler because you need add only new core items (e.g., slot-filling phrases). There is no need to worry about extraneous words and phrases. Otherwise, the same approaches and tests apply.

16.3.4 Tuning Statistical Natural Language Grammars

Tuning a statistical natural language grammar is similar to tuning an SLM: You add the new data to the training set. In this case, the data will need to be both transcribed and labeled with the appropriate slot value.

As with the SLM tuning, hold out some of the new data to add to the test set. Be sure to test performance of both the old and the new models to make sure performance is improving.

It can be informative to look at false accepts, false rejects, and even correct rejects. Multiple rejects can indicate that there is a missing class (a route or service that people are asking for) that should be added to the system. Rejects and false accepts may also indicate that there is possibly a better classification scheme. Perhaps the way the business has divided its services doesn't match the callers' needs or mental models. A card sort approach (see Chapter 8) can be used to test whether there is a better way to organize the set of available services.

16.4 Conclusion

The development, testing, and tuning of grammars, both rule-based and statistical, is a key part of creating a voice user interface. In Chapter 17 we turn to the other development task that often requires the hands-on involvement of VUI designers: coaching recording sessions with voice actors.

Chapter 17. Working with Voice Actors

We deliberately use the term "voice actor" in the title instead of "voice talent" because these two terms have different connotations. When we think of a voice talent, we think of someone with a full and resonant voicesomeone who could be a radio announcer, or perhaps someone who can do different foreign accents and celebrity impersonations. "Talent" connotes a raw, innate gift. In contrast, an actor does more than just perform announcements and vocal tricks. An actor assumes the role of a characterin the world of speech interfaces, perhaps a customer service representative, an executive assistant, a tour guide, or a stockbroker. Actors breathe life into characters, and those who voice speech applications, at least the really good ones, seek to create interactions that engage users.

The recording session is really a performance, and the actor is the star of the show. Accordingly, the primary concern of this chapter is the central role that the voice actor plays in making your application a success. The first topic is scripting for success, because the importance of preparing scripts for actors is vastly underrated. We then discuss considerations for selecting your voice actor and explain how to get the best performance in the recording studio. We also include coaching tips as well as procedural tips for managing the recording session.

17.1 Scripting for Success

If one of your design goals is to create an engaging experience for the user, one that is linguistically and socially familiar, then an essential step is to prepare a script that will enable the voice actor to deliver messages that flow smoothly in the context of an actual interaction between the user and the system. A well-devised recording script can prevent prompts from being delivered with the wrong prosody or from sounding fatigued, insecure, or confused. The success of one of your design goals therefore depends on two general areas of concern:

1. Voice actors should have access to the information they need in order to deliver recordings that capture the right persona and prosody. That is, they should know the intonation, stress patterns, phrasing, and rate of speech that are appropriate to the prompt's function in the context of an actual interaction between the system and a user.

2. The voice actor's script should be formatted for ease of use. It should be clear and legible and should not be visually or mentally fatiguing to the actor.

17.1.1 An Introductory Case Study

As we've mentioned, the two main ideas in this chapter on script preparation are to provide adequate context for the voice actor and to format the material with a view to the actor's comfort and convenience. These concerns are requisites for high-quality, good-sounding results, but they are frequently ignored in practice, as we see in the script

excerpt shown in Figure 17-1. (The actual prompt text has been changed in the interest of anonymity.)

Figure 17-1. This excerpt from an actual script reflects more concern for technicians than the actor.

1.	PR-310	a Chrysler LeBaron	ADD
2.	PR-319	a Chrysler LeBaron	ADD (copy from PR-310)
3.	PR-320	a Chrysler LeBaron	UPDT (copy from PR-310)
4.	PR-321	a Chrysler LeBaron	ADD (copy from PR-310)
5.	PR-330	a Chrysler LeBaron	UPDT (copy from PR-310)
6.	PR-332	a Chrysler LeBaron	ADD (copy from PR-310)
7.	PR-340	a Chrysler LeBaron	UPDT (copy from PR-310)
8.	PR-343	a Chrysler LeBaron	ADD (copy from PR-310)
	...		
31.	PR-AB6	a Chrysler LeBaron	UPDT (copy from PR-310)

The script does not explicitly say so, but (from left to right) the first column refers to the item number on the script. The second is the name of the audio file. The third is the text to be recorded. The last column is reserved for notes to the engineer about individual audio files.

The excerpt in Figure 17-1 reflects more concern for technicians than actors. In general, the script supplies no information illuminating the function of any of the recordings in context. As it happens in this particular application, the item in question should be recorded as in (1).

(1)

Your vehicle, a Chrysler LeBaron, will be ready for pick-up on . . . "

That is, the make and model information should be recorded as a breath group all its own, standing in nonfinal position in the sentence. Another possibility would be to record the make and model as the second half of what is intended to pass as a single, continuous breath group that stands in final position, as in (2).

(2)

Your vehicle is a Chrysler LeBaron.

In either case, the intonation of "a Chrysler LeBaron" is dictated by its contextually determined function as well as placement in the utterance.

This basic lack of concern for contextualization manifests itself in a number of other ways throughout this script. For example, another section lists small function words that are generally unstressedsuch as "a," "an," "the," "or," "and," and "if"in isolation, out of context. In addition, the script seems to have been generated automatically from another document and sorted according to strict alphabetical order by filename. Consequently, portions of multipart messages, such as the vehicle type fragment, are dispersed

throughout the script, often separated by dozens of pages. But the user, of course, is going to experience multipart messages as a unit, so they are best grouped in the same vicinity on the script.

Another oddity is that the type of vehicle in this excerpt occupies nine lines of this script, eight of which are consecutive. Does this mean that all of them are to be inflected identically? Or do the nine occurrences suggest that they are somehow different? Why is so much space and ink, not to mention time and effort, devoted to redundancy? (As it turns out, only one recording was intended; the others were to be copies of the audio file.) Where we would hope to find some contextual clues or direction notes for the voice actor, who should be the center of attention during the recording session, we instead find notes ordering file copying and naming operations for the sound engineer. Presumably, the voice actor or director is charged with reading these file management notes to find out whether the item to the left should be recorded.

The format of this script presents additional problems. For example, it lists 493 prompts printed in 10-point type. This is more likely to give the voice actor a headache, eyestrain, and fatigue than inspiration to voice an interface that sounds upbeat and engaging.

There are many ways that this script can be improved to provide the voice actor with the information needed to provide a prosodically accurate, natural-sounding read. This is the subject of the following section.

17.1.2 Scripting Tips

What can you do to build the voice actor's contextual knowledge and ensure the best delivery possible? The

following sections describe techniques and protocols for helping the voice actor understand the context and therefore the prosody requirements of the messages to be recorded.

The voice actor should also receive direction regarding persona, as well as information about the target users of the application, its areas of functionality, high-level business goals, and so on. This is discussed in section 17.3.1.

Create Useful Direction Notes

Direction notes can be placed in a special "Comments" column. Direction notes explain contextual cuesfor example, what the preceding prompt was, what the user has just said, any information that will help the voice actor deliver the prompt with appropriate prosody, and so on. See Figure 17-2.

Figure 17-2. Good direction notes (shown in the "Comments" column) give the actor and director the context for each prompt.

Item no.	Text	Comments	Filename
18	What's the account number?	First time request (falling intonation)	get_acct_num_ini
19	What was that account number?	Request for repetition (rising intonation)	get_acct_num_err1
20	Tell me the account number one more time.	Preceded by "Sorry, but I *still* didn't get that."	get_acct_num_err2
. . .			
73	Okay, main menu.	This "okay" is a bright and upbeat. Caller has asked to go back to Main.	acknow_ mainmenu.wav
74	Okay then, let's go back to the main menu.	"Okay then" = "no problem." Context: SYS: Do you want to recharge your account? CALLER: No. SYS: Okay then...	acknow_no.wav

Many voice actors prefer that you not write out numbers as words (e.g., "twenty-seven"), especially for digit strings whose formats adhere to familiar conventions, such as phone numbers and times of day. In such cases, rely on direction notes to help the voice actor interpret potentially ambiguous forms. For example, does the numeral "0" mean "oh" or "zero"? Does "1200" mean "one two zero zero" or "twelve hundred"? Does "727" mean "seven two seven" or a "seven twenty-seven"? These are important questions, especially considering that VUI users adopt certain linguistic forms and behaviors that the application itself evidences, and presumably reinforces, through prompting.

Another use of the comments column is for pronunciation tips, as in (3). The "comments" column in Figure 17-3 uses an informal system to show how some unusual city names should be pronounced.

CONTEXT: "Sure, here's the weather for Bangor, Maine."

Figure 17-3. A "Comments" column might include pronunciations of unfamiliar terms.

Item no.	Text	Comments	Filename
44	Bangor, | Maine	say “BANG-gore”	bangor.wav
45	Coeur d’Alene, | Idaho	“kerr-duh-LENN”	coeurdalene.wav
46	Nacogdoches, | Texas	“nag-uh-DOE-chiz”	nagodoches.wav
47	Pawtucket, | Rhode Island	“puh-TUCK-it”	pawtucket.wav
48	Worcester, | Massachusetts	“WOOS-ter” (first syll. rhymes w/ “puss”)	worcester.wav

The phonetic transcriptions in this example are informal, so make sure that your transcriptions do not yield ambiguous or confusing interpretations. You may want to try out your phonetic transcriptions on a few colleagues before finalizing the script.

Group Related Items and Contextualize

Keep prompts together that are similar in function. Order them according to how the user might hear them in an actual interaction with the system. That is, the initial or top-

level prompt should be presented first, error prompts in the order they will be played, and so on. Provide notes for context where appropriate (see Figure 17-4).

Figure 17-4. Group like items, and order them according to the order callers will hear them.

Item no.	Text	Comments	Filename
17	Thanks, now what time does it start?		get_time_ini
18	Sorry about that. Tell me when your appointment starts one more time.	Preceded by: Sys: "Did I get that [the appointment details] right?" Caller: No.	get_time_reentry
19	Sorry, what time was that?	Rising intonation, follows caller's first error	get_time_err1
20	I still didn't get that. Tell me what time your appointment starts, for example, "eight a.m."		get_time_err2
21	Sure, here's some help. To put this appointment in your calendar, I need to know what time it starts, for example, "eight a.m." You can also say "start over" or "Main Menu."		get_time_help

It is typical to find such items, along with everything else to be recorded that day, arranged on scripts in strict alphabetical order by filename. Because of alphabetical sorting, we have seen many scripts in which error prompts actually *precede* the initial or top-level prompt. (To make matters worse, on many of these scripts the filenames themselves fail to reveal the prompt's function, such as "initial" versus "first-time error" versus "help.") Instead, decide on a more intuitive ordering of prompts by state, and be consistent.

It also helps to give prompts names that voice actors will find intuitive, such as `get_time_err2` instead of `20.wav`. The more intuitive and consistent the presentation of material in the script, the less coaching will be needed in the long run.

Especially where concatenation items are concerned, keep prosodically related items together in the script. For example, if your application requires recordings of digits with a falling, final intonation contour (%1see Chapter 11), record these digits in sequence. Similarly, record all digits with rising intonation (%3) as a separate group, and all digits with prepausal, nonfinal intonation (%2) as yet another group.

A popular alternative, which we do not endorse, is to interleave these three intonation typesfor example, by recording "one" (rising), "one" (prepausal, nonfinal), and "one" (falling, final), "two," "two," "two," and so on. This approach is highly error-prone because the interleaved sequence makes most voice actors automatically revert to list intonation, and this means that recordings intended as nonfinal/prepausal (%2) end up sounding identical or at least very similar to the rising versions (%3). In this case, list intonation is an inadvertent and undesirable artifact of the material's presentation.

Consider an application that reads back cardinal numbers in dollar amounts. The cardinal numbers are zero through 99, 100 to 900 by hundreds, 1,000 to 99,000 by thousands, and so on. (As it happens, the application also reads back dates consisting of names of months followed by ordinal numbers, but we will deal with this case separately because it constitutes a distinct prosodic and semantic module.)

In any case, there are several ways that all this material can be scripted. As always, keep prosodically and

semantically related items together, and provide adequate context, as in Figure 17-5.

CONTEXT: "Here's your account balance: five hundred eighty-two dollars and sixty-five cents."

Figure 17-5. Keep prosodically and semantically related items together, and provide adequate context.

Item no.	Text	Filename
157	... 100 | dollars | and 1 cent.	100.wav dollars_nonfin.wav and1cent.wav
158	... 200 | dollars | and 2 cents.	200.wav and2cents.wav
159	... 300 | dollars | and 3 cents.	300.wav and3cents.wav
160	... 400 | dollars | and 4 cents.	400.wav and4cents.wav

The underlined words in the context header are the recording targets for this portion of the script. The pipe symbol (|) indicates a concatenation break; at these points, the voice actor should allow the slightest of pauses

so that the sound engineer has just enough "room" to crop out and save the desired take. There are two reasons for preferring the pipe symbol over the more usual indicator of concatenation breaks, which are ellipses (. . .). First, ellipses are often used to suggest a more prominent, more drawn-out pause, which in turn has a significant effect on the prosody of the preceding material. Second, the pipe symbol simply takes up less horizontal space.

In Figure 17-5, each item yields at least two files to be cropped and saved. Although it may seem time-consuming to organize a script in this way, this example shows that scripts can be prepared with efficiency in mind and still provide all the benefits of recording in context. Of course, it is not always practical to massage a script to this degree of leanness and economy, especially under time pressure. The excerpt in Figure 17-6 is a simpler version of Figure 17-5 but will consequently require a longer completed script.

CONTEXT: "Here's your account balance: five hundred dollars and sixty-five cents."

Figure 17-6. This simpler version of Figure 17-5 will require a longer completed script.

Item no.	Text	Filename
157	... 100 ...	100.wav
158	... 200 ...	200.wav
159	... 300 ...	300.wav
160	... 400 ...	400.wav

Most developers seem to prefer the quicker, more straightforward preparation of Figure 17-6 than the more time-consuming, linguistically strategized script of Figure 17-5.

The same scripting techniques hold for dates. For example, "first" to "thirty-first" can be scripted and captured along with the months, as in Figure 17-7. To facilitate cropping, make sure the voice actor leaves a little space between the month and the ordinal. Alternatively, the ordinals and months can be captured more simply in separate lists (not shown here). Either way, you should always provide context to ensure natural-sounding results.

CONTEXT: ". . . And in interest, you've earned: one dollar and sixty-nine cents on October seventh."

Figure 17-7. Recording months and ordinals together will make for more natural-sounding concatenation.

Item no.	Text	Filename
24	... | on January | 1st.	on_january.wav 1st_fin.wav
25	... | on February | 2nd.	on_february.wav 2nd_fin.wav
26	... | on March | 3rd.	on_march.wav 3rd_fin.wav

Figure 17-8 shows how you might script a set of numbers that will figure as winning scores in a sports application.

CONTEXT: The Mets defeated the Padres thirteen to seven.

Figure 17-8. Here is how you might script sports scores.

Item no.	Text	Filename
117	... nine	to eight.	9_rising.wav to_8.wav
118	... eight	to seven.	8_rising.wav to_7.wav
119	... seven	to six.	7_rising.wav to_6.wav

Winning-losing score sequences more or less conform to a rising-falling intonation contour, assuming they fall at the end of declarative sentences (statements). By capturing these pairs in their naturally occurring context, you can attain more natural results.

As an aside, we recommend recording "to" with the number that follows rather than concatenating it as needed. When the preposition "to" is recorded by itself and concatenated with a number, it often sounds like the number "two," which is confusing in this context. This is because the preposition "to" is conversationally pronounced with a reduced vowel (schwa), whereas "two" is never reduced.

Again, many developers prefer to record the high and low scores in separate recording groups. In this case, make sure that the voice actor is familiar with the context to deliver the target files with the appropriate prosody. This scripting technique applies to a number of other scenarios, such as high-and-low temperature sequencesfor example, ". . . with a high of 72 and a low of 63."

Indicate Contrastive Stress

If the context calls for special emphasis on a particular word (contrastive stress), this should be indicated on the script. Figure 17-9 shows how contrastive stress can be indicated with italics. (These prompts have been culled from various applications.)

Figure 17-9. The script indicates contrastive stress using italics.

Item no.	Text to be recorded	Comments	Filename
15	I *think* you said goodbye, but I'm not sure. So, *are* we done for now?	System needs to verify before signing off.	confirm_goodbye_ini
16	Okay, now how *much* do you want to pay?	Previous prompt was "What bill do you want to pay?"	get_amount_ini
17	You don't *have* any messages.	Caller just said "Get my messages."	no_messages
18	Is *that* right?	To confirm 2nd or 3rd item on N-best list: "What about 'Texas'? Is *that* right?"	

Special or unusual stress patterns can also be indicated with accent marks, underlining, or capital letters, as in (3).

(3)

Is thát right?

Is that right?

Is THAT right?

Use Punctuation Wisely

Most voice actors take their punctuation fairly seriously. Observing details such as punctuation, along with phonetic tips and direction notes, is emblematic of professionalism in this industry.

About Alphabetization

Scripts that are organized according to alphabetical order by filename obscure context, which is essential for delivering appropriate prosody. In theory, alphabetization does not necessarily imply undesirable context insensitivity, but it always seems to be the case in practice. The director and actor thus are faced with having to figure out the context *in spite of the script*, and that imposes an undue cognitive burden on both.

Consider the excerpt in example (4), which was taken from an actual script.

(4)

Eighty-three thousand

Eighty-two

Eighty-two thousand

Eleven

Eleventh

eleven thousand

February

Fifteen

fifteenth

fifteen thousand

fifty

fifty thousand

fifty-eight

fifty-eight thousand

fifty-five

fifty-five thousand

This hodgepodge of cardinal numbers, ordinal numbers, and months is mentally taxing if the goal is to ensure prosodic appropriateness. It is highly unlikely that these items are to be recorded with identical intonation contours, so the voice director will have to direct and redirect as the actor progresses from item to item. The voice actor, in turn, will have to internalize a different context for each item to produce a context-sensitive, prosodically appropriate delivery.

In other words, the presentation of material in (4) requires lot of unnecessary mental gear-shifting. And in case you're wondering, there appears to be no obvious reason why some items should begin with a capital letter, whereas others do not, but this is a formatting issue. Compare this unwieldy list with the more manageable arrangements of this *same* material recommended in Figures 17-5, 17-6, and 17-7.

Often, scripts are alphabetized because of the way certain tools are built. These tools are designed to extract filenames and the associated text from a document, such as a dialog specification document, and then sort the results. Alphabetization, however, does not always promote the voice actor's contextual knowledge of how items relate to each other.

First, let's consider **commas**. For example, the prosody of the message in (5) is different from that of (6).

(5)

Sorry, I'm having trouble understanding. Let's start over.

(6)

Sorry I'm having trouble understanding. Let's start over.

Specifically, in (5) the statement, "I'm having trouble understanding" is prefaced with a distinct transitional element, "sorry," similar in use to sentence-initial disjunct adverbs such as "unfortunately" or "regretfully." So there are three prosodic groupings here: "Sorry," "I'm having trouble understanding," and "Let's start over." In contrast, (6) consists of only two such groupings; the first means "I apologize for this difficulty," and the second is "Let's start over."

This point may seem minor, but it is only an example to illustrate how the comma can influence delivery and the listener's perception of **thought groups**, which are an essential aspect of prosody. Omitting commas, or inserting them where they don't belong, can also render a prompt unintelligible or ungrammatical to the reader or listener. For example, the prompt, "Tell me what's the registration number?" seems to be ungrammatical, as if it were written or spoken by a nonnative student of English. What the writer actually intended was, "Tell me, what's the registration number?" Without the comma, the question is ungrammatical.

Periods suggest "final" intonationa melodic fall to a relatively low pitch levelso make sure you are using them appropriately. For example, each of the items in (7) is followed by a period and thus is delivered with the falling-final contour.

(7)

. . . 2001.

. . . 2002.

. . . a Boeing 727.

. . . an MD-11.

Perhaps owing to force of habit, there seems to be a tendency to put a period at the end of every line of a script, even when the final contour is not intended. We recommend against that practice. Use periods only where appropriate.

Colons, too, are prosodically meaningful. Compare the effect of the colon in (9) and (11) with the lack of a colon in (8) and (10), respectively. Square brackets mark text-to-speech material in (10) and (11).

(8)

Your choices are The Pimco Low Duration Fund, The Pimco Stable Value Fund, or The Pimco Foreign Fund.

(9)

Your choices are: The Pimco Low Duration Fund, The Pimco Stable Value Fund, or The Pimco Foreign Fund.

(10)

The street address is [1313 Mockingbird Lane].

(11)

The street address is: [1313 Mockingbird Lane].

In these examples, the colon at the end of each opening phrase ensures a more graceful hand-off from a piece of naturally recorded audio to a concatenated list or to TTS. This colon cues the actor to treat the item as a single breath group all its own and invites a natural pause, which in turn will coincide with the first concatenation break.

Perhaps it seems obvious, but **parentheses** are the best way to indicate information that serves a parenthetical function and therefore calls for a parenthetical delivery, as in (12), (13), and (14).

(12)

I need to record your voice a few times so I'll be able to recognize your voice next time you call. (This is just for today.)

(13)

Got it. Here's your financial report . . . (Remember, all market data is delayed by at least twenty minutes.)

(14)

Whenever you like, you can also say "main menu" or "customer service." (Or if you're finished, feel free to hang up.) So, what would you like?

Parentheses should be used consistently, so don't use them to bracket direction notesfor example, "(Friendly)." To avoid confusion, you can place these notes in, for example, square brackets. Alternatively, you can separate them from the prompt text altogether and into their own column, as we have done throughout this chapter.

Avoid symbols that can be read aloud in more than one wayfor example, the dashes ( ) in "Our business hours are MondayFriday, 8:30 a.m.8:30 p.m., EST." This message can be more clearly scripted as, for example, "Our business hours are Monday through Friday, 8:30 a.m. to 8:30 p.m., Eastern Standard Time," where the first dash has been replaced with "through" and the second with "to." Note also that "EST" has been rewritten as "Eastern Standard Time," in the interest of clarity.

We sometimes find prompts that make sense in print but do not lend themselves to being read aloud. For example, "To be connected to a representative/agent, press 0." How does the author of this prompt intend the voice actor to read "representative/agent"? Is the voice actor supposed to say "representative 'slash' agent"? "Representative and/or agent"? Or was this simply some sort of mental Post-it that accidentally escaped revision? In any case, we recommend something like, "To speak with a customer service representative, press zero."

Follow Practical Guidelines

Finally, there are some practical guidelines to keep in mind when you prepare the script:

- Use a large font.

- Use 1.5 or double spacing.

- Print the script in landscape mode (wider than it is long). This will accommodate the columns we have recommended. In particular, try to make the prompt text column, which is the voice actor's center of attention, as wide as possible so that the actor's eyes don't tire from constantly having to jump from line to line after every few words.

- Break pages between prompts, not interrupting them.

- Number the items on the script as well as the pages.

- Don't staple. Most voice actors prefer to work with one page at a time. Stapled scripts are unwieldy.

- Keep scripts current and accurate. This will help avoid version-control problems later on.

17.2 Choosing Your Voice Actor

Careful selection of a voice actor is a necessary ingredient in producing a successful, branded user experience. The following are important considerations for choosing an actor who will be appropriate to your application.

17.2.1 Professionalism and Experience

The voice-over industry is vast, comprising areas as diverse as radio, animation, CD-ROMs, documentary films, industrial training videos, airport announcements, and informational videos, to name a few. VUIs, however, present unique and significant challenges. For example, when a voice actor records an instructional video on in-flight safety and emergency procedures, the meaning and context of each utterance are self-evident, and so most actors will not need much direction. In voicing a speech application, however, it is often not obvious how a given recording will fit into the larger interaction between the system and the user. The voice actor must make a conscious effort to internalize the context of each utterance, always thinking, "How will this recording sound in context?"

In addition, VUIs are notorious for long, tortuous help prompts that contain many disparate bits of information. The voice actor should be able to communicate such gnarly bundles of information to the user with an air of effortlessness. Voice actors who are experienced with speech systems are able to quickly grasp the meaning of these long prompts and deliver them with just the right

inflections to guide the listener smoothly through the content.

The ability to manipulate prosodic variables, such as pitch and rhythm, is highly relevant to the success of concatenation-intensive VUIs. The voice actor should have an ear that not only can distinguish different intonation contours but can also produce them without sacrificing the desired energy, tone, mood, or persona. An experienced voice actor should also be able to perform stress shifts in isolationfor example, "SEVenteen" versus "sevenTEEN."

Trained voice actors are sensitive to different kinds of variation in pronunciation. Having received proper direction, the trained actor should stay on phonetic course. The persona design may dictate pronunciation differences based on socio economic variation, regional variation, ethnic variation, or stylistic variation.

17.2.2 Coachability

Some musicians have a technical understanding of the music they play and of the instrument they play it on, but others do not, although they can still, say, "bang out a tune" on the piano. A similar dichotomy exists among voice actors. It is worth your while to find a voice actor who has technical knowledge and control of his or her instrument and who takes technical direction well, especially if your VUI relies on concatenation. A voice actor who has poor voice control, who cannot match an intonation contour, or who does not take direction well can turn a simple three- or four-hour recording session into an all-day affair, including the time it will take for extra editing on fumbled takes. Of course, you can find out how coachable an actor is by simply holding an audition (discussed in a moment).

17.2.3 Fit with Persona

Choose a voice actor who captures the desired persona for your application and who will be able to project that character consistently and effectively over the phone. An actor's suitability for a particular persona can be determined in a brief audition.

17.2.4 Demo Tapes (or CDs) and Auditions

Demo tapes and CDs are essential for narrowing the selection of voice, but they are no substitute for auditions. Auditioning a few key performance areas should provide an accurate picture of the actor's skill level and suitability for your speech application.

- Determine the actor's ability to interpret the persona by recording a few sample dialogs, including "best path" and error scenarios. In recording these persona samples, you may even find that the actor will bring greater depth to the character than you may have anticipated.

- Determine the actor's ability to take direction for concatenation-intensive messages, which include dates, times of day, money amounts, travel itineraries, driving directions, and so on.

- Whenever anyone reads a passage out loud, listeners can tell how well the reader understands the content. When someone, even a voice actor, reads a passage that he or she comprehends poorly or not at all, this too

is obvious. It is therefore important to test the actor's ability to convey meaning *in the persona* under more difficult linguistic circumstances. Have the actor run through some longer informational messages, such as help prompts or a tutorial.

- Listen carefully for any impediments that will be noticeable over the telephone, excessive "mouth noise," "wobble," lack of vocal control, or any other vocal characteristics that might be received unfavorably.

17.3 Running a Recording Session

A successful recording session depends on a certain familiarity with procedure, as well as knowing *what* to communicate to a voice actor and *how*.

17.3.1 Procedural Considerations

There are a number of procedural considerations that will make the session run more smoothly and keep the voice actor in good form. Remember that the actor is not a machine but a person, and people sometimes have special personal and environmental needs to perform at their best.

Prepare the Voice Actor

Don't expect your voice actor to show up, receive a script, and immediately start recording prompts without receiving an overview of the application's purpose, goals, or areas of functionality. In addition to furnishing *local* context for individual prompts, as recommended earlier, it is crucial that you inform the voice actor of the same *global* issues that were deemed relevant for creating an appropriate, application-specific persona, as outlined in the persona design checklist given in Chapter 6.

Make sure that the voice actor is aware not only of relevant business concerns but also of any information about typical users, such as demographics, whether they will be calling frequently, their likely mind-set, and so on. Get a copy of the recording script and a description of the interface's

persona to the voice actor beforehandif possible, a day before the session.

Recording Procedure

Decide on your recording procedure before the session begins, and inform both the voice actor and the sound engineer of your plan. You should decide in advance how many times the voice actor should say each item in the script.

In planning your recording procedure, think about the nature of the content you are recording and how it functions in the application. Suppose you are recording "one minute" up to "fifty-nine minutes," all with the same intonation contour, or a list of more than 100 relatively simple city-and-state names (e.g., "Providence, Rhode Island"). In that case, instruct the actor to simply do one take of each item, adding something like, "Just move on to the next city, unless I cue you for a retake." This procedure will ensure that the recordings are elicited as quickly and efficiently as possible.

This approach is most appropriate when the items to be recorded will not be heard frequently and when the list is lengthy and repetitive, but simple. In this case, it is not worth eliciting more than one token of each city and state unless it is necessary. On the other hand, if you are recording messages that are more complex or more informative, or that will be heard rather frequently, have the voice actor do three to five takes of each one.

It is not advisable to start the recording session with, for example, the welcome prompt, or system greeting. This is the one message that all users will hear and will be the basis for all users' very first impression. You want to make

sure the voice actor is adequately warmed up before reading such critical prompts. The same holds true for other high-traffic prompts, such as "What's your account number?" and "Sorry, I didn't catch that," as well as any difficult or tricky messages, including concatenation pieces.

Planning Tips

Keep these guidelines in mind while planning your session:

- **Duration:** The session should not exceed four hours in a given day. Vocal production is muscle activity, and muscles get fatigued with repetitive use.
- **Time of day:** Voices change throughout the day. If recording will take place over the course of a few days, schedule consistently.
- **Beverages:** A note about beverages may seem trivial, but we raise this issue because they directly affect the vocal tract and therefore the sound of your recordings. Make a beverage available, but avoid ice-cold beverages, which constrict the vocal cords. Also avoid dairy and thick juices, which have a tendency to coat the throat and make the voice sound thick.
- **Breaks:** Take a short break every hour, unless you and the voice actor feel you're on a roll and don't want to break the momentum.
- **Sitting versus standing:** Although this is a matter of the voice actor's personal preference, the recording area should be equipped with a comfortable seat with decent back support. Many voice actors prefer to stand

because it facilitates a more energetic delivery and fuller, more controlled involvement of the diaphragm.

17.3.2 Voice Coaching

A smooth-flowing, comfortable dialog between the user and the system depends in part on what the voice actor both knows and feels in the recording studio, and both knowledge and feelings are mediated by the voice coach. The role of the voice coach is therefore critical in the production of the VUI and should not be underestimated.

First, it is the coach who communicates the application's objectives and functionality, user demographics and psychographics, and the context of the prompts to be recorded. In addition, the coach must also be knowledgeable about language in general. For example, the coach must know whether a Wh- questionsuch as "What's your account number?"is intended as a first-time request or as a request for repetition, and must know (at least implicitly) that such a difference determines the question's intonation contour. Much of what the actor feels in the recording studio will depend on the voice coach's social intelligence, interpersonal skills, and emotional sensitivity.

In general, the coach should avoid editing prompts in the studio. If there is a compelling reason to make a changeperhaps some wording is clearly wrong when you hear the voice actor create itmake sure to document the change and notify both the grammar writer and the dialog designer.

It is unlikely that the prompts will be recorded in the order that users will hear them, so it is important to maintain consistency throughout the recording session. This means consistency in volume, distance from the microphone,

energy, and interpretation of the persona. You can prevent most inconsistencies in vocal quality by being mindful of the duration of the session, the time of day it is scheduled, the consumption of certain beverages, and so on, as indicated earlier.

As early in the session as possible, identify one or two key recordings that you feel are especially successful in capturing the desired mood, energy level, and persona. Keep these recordings handy so that you can revisit them throughout the session. If there are any additional recording sessions, start by having the voice actor listen to these key recordings so that he or she can "find the voice" from the earlier session.

The voice coach should try to create a relaxed, pleasant, warm, and supportive environment. Give the voice actor plenty of positive feedback. Communicate praise and encouragement through your own positive body language. Gestures as subtle as a nod or thumbs up communicate reassurance effectively without disrupting the flow of recording.

Be aware of the type and amount of direction you are furnishing the voice actor. Do not overcoach. In other words, try not to give overly explicit, overly technical instructions about how you want things to sound. Otherwise, you run the risk of making the voice actor overly self-conscious, and that can result in awkward or unnatural-sounding prompts. If you are not making progress on a particular item on the script, a good rule of thumb is to move on and return to the problematic item later in the session.

Voice coaches should be aware of the difference between sloppy diction and colloquial pronunciation. Sloppy diction refers to mumbled, slurred, or garbled speech caused either by a habitual way of speaking or an isolated, unintentional

speech production slip-up. Sloppy diction should be avoided in speech interfaces because it detracts from clarity and can thus lead to listening comprehension problems. It also gives the impression of low production values and can therefore be damaging to your branding efforts. In contrast, colloquial pronunciations, such as "gonna," "wanna," and "lemme," are everyday, conversational alternatives to the more formal, more careful "going to," "want to," and "let me." Remember that the voice actor's pronunciation should evoke a linguistically and socially familiar persona. Use the pronunciation variant that is appropriate in the context of the dialog and persona.

More than anything else, a good voice coach is a good listener. The voice coach must be listening attentively while the voice actor is recording. The coach should be paying attention to prosody, the quality of vowels and consonants, undesirable mouth noise, and consistency of volume, energy, mood, and persona. In sum, a good voice coach listens on many levels at once and directs the voice actor to produce recordings that will work in context and satisfy the business and user requirements of the application.

17.4 Conclusion

In the same way that a successful VUI is designed with an overarching concern for its potential users, so should your recording script be designed with *its* user in mind, the voice actor. In other words, think of script preparation fundamentally as a usability task. Consider what the voice actor needs to know to deliver successful recordings. Consider how each recording will be heard by callers when they encounter each one in context. Scripts should therefore build the voice actor's contextual knowledge, and they should also be formatted for readability and the actor's comfort.

In addition to being familiar with the script, a good director must be familiar with certain procedural details of session management. More importantly, the director needs to communicate to the voice actor the business goals and user requirements that underlie the persona and dialog design. The director must also know how the application works overall and how each prompt to be recorded figures in context. After all, the order in which the voice actor records the prompts will certainly not be the order in which callers will hear them.

Merely "understanding" business, user, and application requirements is not enough; the director must also have an ability to communicate this information to the voice actor efficiently and effectively. A successful director knows how to set the tone of the session, creating a supportive, nonthreatening atmosphere in order to bring out the best in a voice actor. Finally, a successful director possesses a keen range of listening skills, able to attend to the many auditory nuances that, when taken together in context, users will

perceive as a persona who is optimally comprehensible and socially appropriate.

Chapter 18. Sample Application: Development, Testing, and Tuning

We are now at the realization phase of the Lexington Brokerage project. After implementation and integration, the application will go through testing and tuning to ensure that the design meets the established requirements, that accuracy is optimal, and that the dialog is tuned for an effective user experience.

18.1 Development

The development phase takes all the thinking that has gone into definition and design and transforms these ideas into concrete deliverables, including code, grammars, and audio recordings.

18.1.1 Application Development

Lexington decides that the application should be developed in VoiceXML. The executives want to avoid proprietary languages to ensure flexibility in the future, while taking advantage of the speech-centric nature of VoiceXML. Ultimately, they want to be able to update the application themselves, so they are training some members of their development team on VoiceXML.

Based on the definitions of grammar slots and sample phrases for each dialog state, the developers create simple **stub grammars** that they can use for unit-testing of modules before the completion of grammar development. They also create stubs for database return values to accommodate testing before completion of all the backend integrations.

The backend and CTI integrations are straightforward. They are implemented by Lexington. The firm decides to reuse the integrations to its touchtone system to the greatest extent possible. The customer account database is updated to accommodate personalization features, such as the tracking of calls to the system, so that the necessary information is available for just-in-time instruction. To accommodate future additions for greater personalization, they make the changes in an easily extensible framework.

From time to time during development, members of the development team meet with the VUI designer to make sure that they fully understand the dialog spec. One of the developers suggests eliminating the need for multiple entry prompts to dialog states by rewording those prompts to be more generic. He estimates such a change will save two days of development time when you count the development and testing effort. This idea is rejected because the change would significantly diminish the naturalness of the dialog flow and would risk reducing clarity for callers.

In general, the application development proceeds smoothly, according to schedule.

18.1.2 Grammar Development

A number of types of grammars must be developed. Rule-based grammars are needed for those dialog states that are part of a directed dialog. SLMs are needed for states that accept more flexible input. Robust natural language grammars are needed for semantic interpretation for those states that use SLMs for a recognition grammar.

As we begin working on rule-based grammars for specific dialog states, we look at the slot definitions and sample phrases defined for that state. We also look at the prompt wording. These give us a starting point for imagining the variety of ways a caller may respond to our prompt.

Before beginning to flesh out the grammar, we create an initial test suite, which is simply a list of phrases and sentences we want to make sure the grammar covers. We then flesh out the grammar definition and add items to the test suite. The following is our first version of the grammar for the `GetAccountNumber` state:

```
.GetAccountNumber

     (?Uh

      [(?Prefiller AccountNumber:number)

                        {<account_number $number>}

       Unknown {<unknown true>}

      ]

     )


Uh [uh hm um]


Prefiller [( [my the] ?account number is )

           [ it's (it is) ]

          ]


AccountNumber

     (Digit:d1 Digit:d2 Digit:d3 Digit:d4 Digit:d5
Digit:d6

      Digit:d7 Digit:d8 Digit:d9 Digit:d10
Digit:d11 Digit:d12)

     {return (strcat($d1 strcat($d2 strcat($d3
```

```
strcat($d4

              strcat($d5 strcat($d6 strcat($d7
strcat($d8

              strcat($d9 strcat($d10 strcat($d11
$d12)))))))


)))))}



Unknown [( ?i [ (do not) don't ] know ?[it (what
it is) ] )

         ( ?i dunno )

         ( ?i [ (do not) don't ] remember ?it )

         ( ?i [ (do not) don't ] have it ?(with
me) )

         ( [i'm (i am)] not sure )

         ( i have no idea )

        ]



Digit [  [oh zero]  {return(0)}

         one        {return(1)}

         two        {return(2)}
```

```
        three       {return(3)}

        four        {return(4)}

        five        {return(5)}

        six         {return(6)}

        seven       {return(7)}

        eight       {return(8)}

        nine        {return(9)}

    ]
```

(Note that `strcat` is a function in GSL that concatenates strings. The `AccountNumber` subgrammar demonstrates a way to return a continuous string of 12 digits.)

In a similar fashion, we create grammars for all the states using directed dialogs.

The next issue is the bootstrapping of the SLM so that we have a good starting point. Because we have done a number of similar projects in the past, with the same functionality available at the main menu, we have data we can use for training an initial SLM. Although we expect improved performance after we train with application-specific data, this initial model should be good enough to get us to pilot, at which point we will collect a lot of data with the working application. We split off a portion of the data to use as a test set.

Next, we create the robust natural language grammars for the states that use SLMs. In addition to looking at the definitions of slots and sample phrases for these states, we

look at the transcriptions used for training the SLM; this gives us an idea of the phrases we should cover in our grammars. The development of these grammars is straightforward, given that we need to cover only the core slot-filling phrases and not the fillers. For a test suite for the robust grammars, we develop one by hand, in the same way we did for rule-based grammars. We also add the transcriptions from the SLM test set.

18.1.3 Audio Production

Audio production includes the selection of the voice actor, coaching recording sessions, postprocessing prompt recordings, and creating nonverbal audio. As discussed in Chapter 14, for this project we selected the voice actor early so that we could get feedback on the persona from the iterative usability tests we ran during the detailed design phase. We also began recording company name prompts as soon as the voice actor was chosen. In this way, we had time to record all 15,000 names in a series of recording sessions over the course of a few weeks.

To select the voice actor, we get a CD from a talent agency we often work with containing examples from 10 male voice actors. With the persona definition in mind, we narrow the choice to two possible voices. Lexington listens to the CD and agrees with these choices. We decide to bring in both actors for live auditions.

Both auditions follow the same process. When the voice actor arrives, we give him a copy of the persona definition and discuss with him the application and persona. We then go through a brief recording session. The session begins by recording the prompts for a few of the sample dialogs, including login, trades, and disambiguation of company names. We include a number of complex help prompts and

just-in-time instructions. We then test each actor's coachability on prosody for complex concatenated prompts, focusing on confirmation prompts for trades. We have them record a few company names, numbers to be used for the number of shares, numbers for stock prices, and the other pieces of trade confirmations (e.g., "Confirming: You want to buy"). After the session, we use the recordings to splice together some confirmation prompts. Finally, we listen to all the material over the telephone.

After the auditions, one candidate stands out as the clear choice. The Lexington team listens to the audition recordings and agrees. We sign a contract and set up a schedule of recording sessions. The first session covers the material needed for the first usability test. We then begin a series of recording sessions, gradually covering the 15,000 company names over the following few weeks. At the beginning of each recording session, we have the voice actor listen to recordings from previous sessions to get back into character and match the voice.

18.2 Testing

The code for the application is finally complete and goes through several rounds of bug fixes. As a part of this process, our dialog designer performs a dialog traversal test, calling into the system and exhaustively traversing every dialog state and identifying any areas where the application behavior does not match the logic defined in the dialog specification. The application also undergoes QA tests and load tests to ensure that there are no problems and that the fully integrated system is provisioned correctly.

After application testing, we have a handful of employees call into the system to check speech recognition performance. We supply the volunteers with a list of things to say to the system and also let them talk to it without a structured script. To test the performance of recognition for the states using SLMs, we use the test set we earlier set aside. We decide that recognition performance is in the ballpark; no special problems are noted.

After we have written grammars for all the dialog states in the specification, we are ready to test them. First, we use the test suites we created to ensure that we have covered everything we intended to have in the grammar. We also make sure that each utterance returns the correct slot values. Using other grammar testing tools, we check to see that the grammars do not unintentionally include utterances that the caller would never say, such as, "I want quotes for my the holdings." We also eliminate any ambiguity in the grammar that the system is not designed to handle.

Finally, we do a pronunciation test on company names to make sure that the dictionary entries are reasonable. Given

the large number of company names, we decide to use the company name prompt recordings for this test. We feed them to the recognizer, running in batch mode, using the company-name grammar. A few companies are rejected. A quick fix to the dictionary solves the problem.

18.2.1 Evaluative Usability Testing

We run an evaluative usability test. This time, rather than use a Wizard of Oz approach, we run the tests using the real, fully integrated system. This is our first opportunity to observe people use the actual system.

The format of the tests is similar to that of the earlier iterative usability tests. We design a set of task scenarios for participants to run. This time, we have them exercise additional functionality, including setting up a watch list and changing an existing open order. We create some fake customer accounts for them to use. We will use the same debriefing questions used in the earlier tests, with the addition of questions on accuracy. Again, we will run the test over the phone, calling subjects at home.

On the first day of testing, we run seven participants. We discover a timing issue that causes problems for five of the seven participants. It turns out that there is significant latency in the system response after an account number is entered. This latency is a result of the checksum test on the items in the N-best list as well as the database latency. The delay is often close to five seconds. Our participants are confused by the delay; they are not sure the system has heard them. After a few seconds of silence, some of them say "Hello?" and others repeat their account number. In general, this kind of timing problem could not have been found until we were testing with the actual system.

We decide to add the following prompt, which will play immediately after a caller finishes saying the account number: "Please hold while I access your account." The voice actor comes in the next day to record it, and we quickly make the change to the system. Two days later, we run seven more participants. None of them has the problem; the extra prompt has been successful in clarifying for them what the system is doing. Furthermore, all subjects successfully complete all tasks, and we have very positive responses to our survey questions.

The application is now in good shape. Implementation bugs have been fixed, recognition accuracy is in the expected range, and the system has achieved good scores on the evaluative usability test. We agree with the client that the system is ready for the pilot test.

18.3 Tuning

The goal of the tuning phase is to have real customers interact with the system and provide enough data to help us tune the grammars (both rule-based and SLM), recognition parameters, dictionary, and of course the dialog flow. Often, pilot studies can introduce a Catch-22 dilemma for clients. They are reluctant to let customers interact with an untuned system, but to tune the system, you need real customer data. The Lexington executives have been comfortable with the results of the recognition test and evaluative usability, so they are ready to move forward with the pilot. We do manage their expectations and explain that the accuracy of the SLM will not be as high in the early pilot as we will ultimately achieve after we have application-specific training data.

The pilot is scheduled to run for eight weeks. For the first four weeks, Lexington will redirect 10 percent of its touchtone traffic to the speech system, resulting in approximately 20,000 calls per week. If all goes well, we will increase it to 20 percent for the next four weeks. We will send out user surveys at the end of week 3 and again, to different users, at the end of week 8. At that point, if both our quantitative measures of system performance and the subjective results of our user survey achieve our success metrics, the system will be fully rolled out.

All the pilot data is logged and the waveforms saved for all user utterances. Our transcription team is geared up, so there is only a one-day lag from the time a phone call comes in until it is transcribed and ready for our analysis.

18.3.1 Dialog Tuning

We begin the tuning process by evaluating the performance of the dialog. We start by using a call monitoring tool to listen to about 100 randomly selected calls. Next, we use a tool that allows us to set criteria to determine which calls we listen to and reconstructs calls from logs by playing the recorded utterances interspersed with the recordings of the prompts.

We select calls in which multiple trades were placed. The result of our usability test is corroborated on real call data: First-time callers often place their first trade step-by-step, allowing the system to lead them through the series of requests for the number of shares, the price, and so on. Then, after hearing the just-in-time instruction describing the more efficient way to specify their trades, they place their next trade more efficiently using a more complex sentence. We listen to 30 such calls. Of these, 28 go step-by-step on their first trade. Of those 28, 22 of them use more complex sentences on their second trade.

By the third week of the pilot, we have 8,142 repeat callers (customers who have called at least twice). Of these, 863 do trades on both their first and second call. Again, the just-in-time instruction has worked; 748 of those callers use complex sentences for their trades on their second call (whereas only 41 did on the first trade of their first call). We conclude that the just-in-time instruction is working both within and across calls.[1]

[1] These and other results on the Lexington Brokerage sample application have been fashioned to illustrate the application design and tuning process, along with ways of thinking about how to solve problems when they are discovered. These results should not be interpreted as specific results of actual valid scientific studies or a specific real deployment.

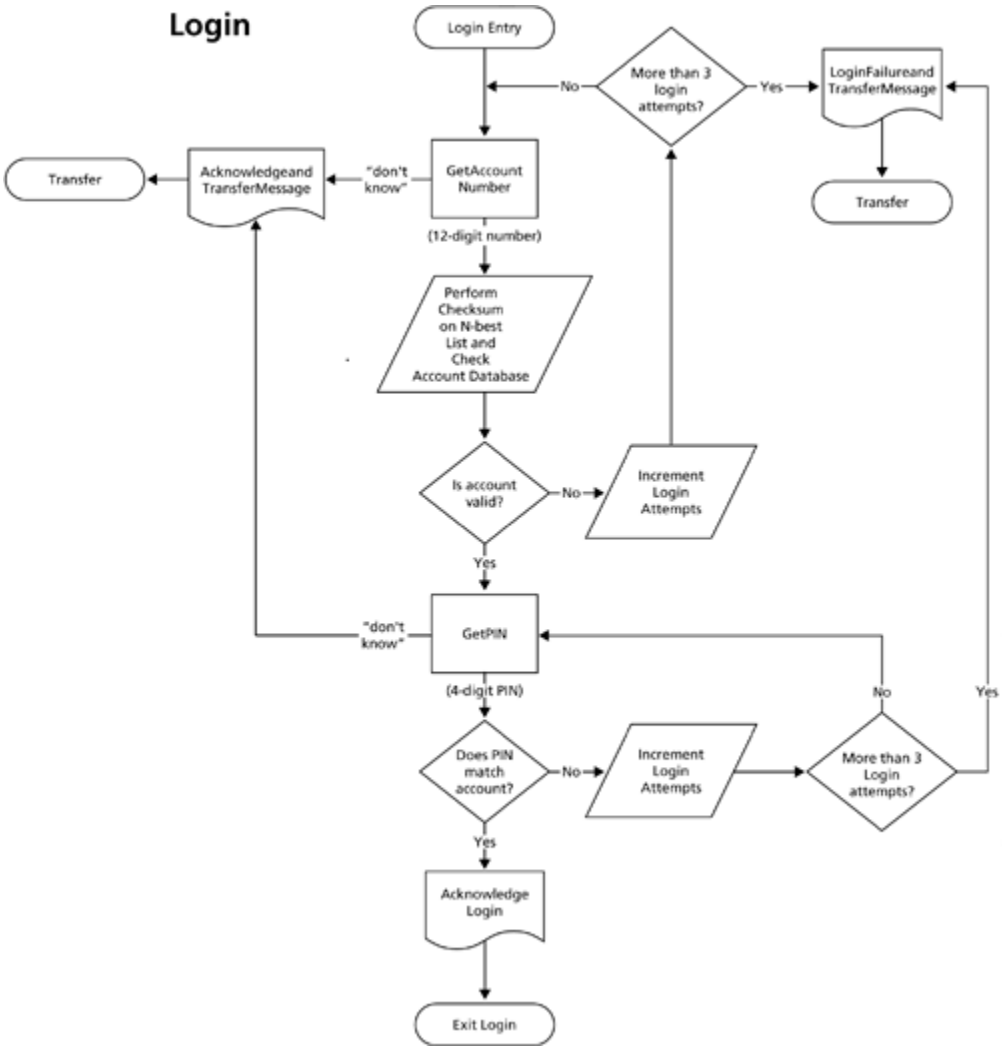
Next, we do a hotspot analysis. We look for dialog states with a high rate of rejects, timeouts, or requests for help. It turns out that the `GetPIN` state has a high reject rate. In fact, many callers are hanging up after one or two rejects in

the `GetPIN` state. We listen to the login dialog for 25 calls that have rejects followed by hang-ups in this state. It turns out that 21 of them seem to be cases in which the caller does not remember the PIN. For example, some of the inputs include, "I don't know," "Um, I can't think of it," and "I don't remember." Some others seem to be guessing; they try one or two numbers that do not match and then hang up.

You can see the problem with our design by looking back at the call flow design for the Login subdialog (Figure 14-2 in Chapter 14). We accommodated callers who did not know their account numbers by recognizing inputs such as "I don't know" and transferring them to an agent who could provide an alternative means to get into their account, but we did not provide the same capability for those who did not know their PIN. Our pilot data show that this is a problem for PINs as well.

Therefore, we change the call flow for the Login subdialog as shown in Figure 18-1, adding a path to transfer those who do not know their PIN to an agent who can help them. We add the Unknown subgrammar, shown earlier in the grammar for the `GetAccountNumber` state, to the `GetPIN` state. Additionally, in the first reject message in the `GetPIN` state, or in response to a request for help, or if the caller is reentering the state after a nonmatching PIN was tested, we tell callers they can say, "I don't know" and get connected to someone who can help them. After the change is implemented and deployed in the pilot system, we collect more data and repeat the hotspot analysis. The data show that the problem is fixed.

Figure 18-1. The revised call flow for the Login subdialog shows a new way of handling unknown PINs.



We are surprised to learn how large a percentage of callers know their account number but do not remember their PIN. The Lexington team is also surprised. We agree to make speaker verification a high-priority feature for the next version of the system. In that way, customers will not have to remember a PIN; instead, their voiceprint will provide security. It is interesting to note that the lesson about the shortcomings of the `GetPIN` state could have been learned only from real usage by real callers. A usability study with made-up tasks would not have uncovered such a problem.

The next step in our dialog tuning process is an evaluation of our error recovery and help messages. We use a hotspot analysis to find all states with more than a 10 percent reject

rate. For each such state, we listen to a number of the interactions that had rejects. One thing we learn is that in the directed-dialog portion of trading, when callers are answering the question about the number of shares they want to buy, they often try to check their balance or say things like "Oh, I'd better see how much I have in my account . . . " Therefore, we add phrases for balance queries to the grammar for that state. Additionally, we add to the help and reject messages the sentence, "You can also ask, 'What is my balance?'" Logic is added to the call flow to respond to such a request with the amount of money in the caller's default account, followed by a reprompt for the number of shares.

18.3.2 Recognition Tuning

For each of the states that use rule-based grammars, we divide the data into in-grammar and out-of-grammar sets. On the in-grammar sets, we measure the rates of correct accepts, false accepts, and false rejects. On the out-of-grammar sets, we measure correct rejects and false accepts.

To adjust the recognition parameters, we run **parameter sweeps**: a series of recognition tests, in batch mode, varying the parameter we are tuning. For example, we run a series of recognition tests for the `GetNumberofShares` state in the Trading subdialog, each with a different pruning threshold. We look at the resultin terms of the rate of correct accepts and the average time for recognitionin order to choose a value that optimizes the trade-off between accuracy and time. We use a parameter sweep of the reject threshold to adjust the trade confirmation state, choosing a threshold that makes it extremely unlikely we will have a false accept of "yes."

We find that company name recognition is not as high as we expected. Further analysis shows that there are 12 companies that we almost never get right. They are all company names based on foreign words. We listen to a bunch of the utterances and adjust the dictionary entries. Further testing shows that the solution has worked.

18.3.3 Grammar Tuning

We begin our grammar tuning by looking at the out-of-grammar (OOG) utterances for each of our rule-based grammars. The `GetAccountNumber` state has quite a few OOGs. Analysis of the data shows that the OOGs are a result of three problems:

1. Many callers say "dash" between digit positions 4 and 5, and again between digit positions 8 and 9. A quick check reveals that callers are simply replicating the format they see on their account statements. The 12-digit account numbers are printed in three groups of four digits each, separated by dashes.

2. Callers have a tendency to pause after every group of four digits. In some cases, the pauses are long enough to trigger the end-of-speech detector, causing the recognizer to stop listening and therefore missing the last four or last eight digits.

3. Some callers are using natural numbers. For example, for the four-digit sequence 2347 some callers say "twenty-three forty-seven," for the sequence 3400 some callers say "thirty-four hundred," and for 5000 some say "five thousand." Our grammar accommodates only digits.

To solve the problem of OOGs in which callers say "dash," we add the word "dash" optionally to the grammar between digit positions 4 and 5 and between digit positions 8 and 9. Here is the new version of the `AccountNumber` subgrammar for the `.GetAccountNumber` grammar:

```
AccountNumber

     (Digit:d1 Digit:d2 Digit:d3 Digit:d4 ?dash
Digit:d5

      Digit:d6 Digit:d7 Digit:d8 ?dash Digit:d9
Digit:d10

      Digit:d11 Digit:d12)

     {return (strcat ($d1 strcat($d2 strcat($d3
strcat($d4

               strcat ($d5 strcat($d6 strcat($d7
strcat($d8

               strcat ($d9 strcat($d10 strcat($d11
$d12)))))))


)))))}
```

To solve the problem of pauses causing the endpointer to detect end-of-speech before the caller is actually finished, we adjust the end-of-speech timeout parameter to listen for a longer pause before assuming that the caller has finished speaking.

We consider two solutions to the problem that some callers use natural numbers. The first possible solution is to

augment the grammar to include natural numbers. The second solution is to leave the grammar as is and make it clear in the first reject prompt that callers should "say the digits" of their account number. We decide to use the second approach. Given our desire to achieve very high recognition accuracy for account numbers, we don't want to add the natural numbers to the grammar, something that would make recognition more challenging.

After the changes are implemented and installed in the pilot system, we collect more data. We are able to verify that our three fixes for the account number OOG problems are effective.

The handling of the OOG problems for the `GetAccountNumber` state is a good illustration of the variety of approaches that can be brought to bear to fix what initially appear to be "grammar" problems: In one case we extended the grammar, in another we adjusted a recognition parameter, and in the final case we changed the wording of an error prompt. In general, when OOGs are observed, you must first determine the root cause of the problem. In some cases, once you understand the cause, the solution will be obvious. In other cases, you may need to consider trade-offs between solutions.

The remainder of our grammar tuning goes smoothly. We discover a few alternative ways to refer to certain companies. For example, some callers refer to IBM as "Big Blue." That gets added to the grammar.

When we have the data from the first week of the pilot, we measure recognition accuracy for states that use the SLM. It is roughly what we expect. For the remainder of the pilot, we train a new SLM every two weeks, with increasing amounts of pilot data, and install it on the system. In the first month we see substantial improvements given the

increased training data for the SLMs. In the second month, the improvements continue, although they are smaller.

18.3.4 User Survey

Lexington plans to use a survey to gather subjective data from its customers at two phases. The first set of surveys is sent at the end of week 3 of the pilot. The second set, with identical questions, is sent at the end of week 8 of the pilot to a different set of customers.

The surveys include a set of Likert-scaled questions related to the success criteria, as well as a few questions designed to gather open-ended feedback. As discussed in Chapter 7, this survey is based on the survey the firm has used to evaluate its touchtone system, augmented with questions about accuracy. The goal is to meet all success criteria by the end of the pilot and final survey and then proceed with the full deployment.

At the end of pilot week 3, Lexington sends the survey to 5,000 of its customers who called the system for the first time during that week. In that way, they will have experienced the system after the first iteration of improvements and retraining of the SLM. Two weeks later, responses have been received from 1,254 customers.

The responses, in general, are quite positive, with one exception: The subjective impression of accuracy is not as high as we expected. This is surprising given the high accuracy we are measuring on the data we are gathering from the system. Many of the respondents who rated accuracy low commented on it in the open-ended questions. Their comments help uncover the problem. The following example illustrates the issue:

> *I got a quote on a company and then placed a market order and realized that the market price had changed by a whopping 8%. The quote was delayed, but the trade was based on real time prices.*

The quotes the system is providing are delayed as much as 20 minutes. As a result, customers are not placing trades at the market price they expect. The important lesson from this result is that, to the end user, accuracy is a whole-service concept. A system seems accurate when it consistently does what they expect. Measurements we make in the lab, such as the correct-accept rate on in-grammar utterances, are only part of the story. All the other pieces that contribute to the system's ability to faithfully fulfill a caller's request contribute to the caller's perception of system accuracy and reliability.

We discuss the problem with Lexington. It turns out that the company from which it licenses its quote feeds offers two pricing schemes: one for real-time quotes, and a cheaper one for delayed quotes. Lexington has been using real-time quotes at its brokers' desks and using delayed quotes for the touchtone system. It has never been a problem in the past because trades are not available from the touchtone system. Luckily, the integration is the same whether you license real-time or delayed quotes. Within a week, the speech system is running with real-time quotes.

At the end of week 8 of the pilot, Lexington sends the survey to another 5,000 customers. This time, it chooses customers who used the system for the first time in week 8. Therefore, these respondents have experienced the fully tuned system. Two weeks later, responses have been received from 1,139 customers. The results are very positive.

While waiting for the survey results, we transcribe the data collected during the final week of the pilot and run all our tuning tests. The final tuning reports are submitted to Lexington. At a follow-up meeting, we all agree that both the objective measures in our tuning reports and the subjective measures in the survey show that the system is performing well. We have met all the success criteria established during requirements definition. Lexington feels ready to fully deploy the system. We all go out to dinner to celebrate and discuss future projects!

Chapter 19. Conclusion

The most interesting thing about Radio Rex is that he was a commercial success, a feat that was not repeated in the world of speech technology for many decades. The truth is, you can glean many of the principles in this book just by examining the success of Radio Rex. First, he got the business case right. In 1911, television had not yet been invented, and Radio Rex could keep children happily occupied. That is a major value proposition.

Second, based on a deep understanding of his userstheir capabilities, limitations, linguistic behaviors, and interestshe got the user interface right. It was simple to learn to use: "Just say 'Rex.'" And the persona design was rightchildren loved the little dog.

Third, Radio Rex got the technology right. Simple, one-word recognition was all that was needed (no feature creep here!). It's true that his rejection technology was limited. Rex often responds to any loud sound, not just the word "Rex." For a dog, however, that is probably natural, realistic behavior. In any case, just as the patience of the Schwab system appealed to our first usability subject, the dog's eager, if indiscriminate, responsiveness was a hit with young users.

Rex notwithstanding, VUI design is still a young discipline. We have tried to present the fundamental underpinnings of the discipline and derive best practices in a principled way, supported wherever possible by concrete data and based in all cases on lots of experience. The methodology we present is driven by a number of principles:

- Inform design decisions with end-user input.

- Find solutions that combine business and user goals.
- Avoid expensive changes downstream by focusing on thorough early work.
- Move the design experience close to the user experience so that the designer can conceive of design elements as conversation.
- Make all design decisions with appropriate consideration of context.

The approach to making design decisions is also driven by principles:

- Minimize cognitive load.
- Accommodate conversational expectations.
- Maximize efficiency.
- Maximize clarity.
- Ensure high accuracy.
- Gracefully recover from errors.

It is our hope that you now have a set of valuable best practices that can be applied within an effective methodology. Perhaps more importantly, you should also have an understanding of the underlying fundamentals that you can apply to new situations as they arise.

If we all do our job well, someday we will be able to look back at today's systems the way today we fondly look back at Radio Rex: as a great beginning hinting at vast potentialin terms of technology, business value, and, most importantly, user value.

Appendix APPENDIX

The Computer Phonetic Alphabet (CPA) is a set of symbols for defining the phonetic segments of a language. It parallels the International Phonetic Alphabet (IPA), with changes to make it easy to enter symbols with a standard keyboard. Table A-1 defines the set of CPA symbols used by Nuance for American English. These symbols are used to define pronunciations in this book.[1]

[1] This table is reproduced from the *Nuance Grammar Developer's Guide*, Version 8.0.

Table A-1. CPA Symbols for American English

CATEGORY	PHONEME	EXAMPLE	PHONEME	EXAMPLE
Vowels	i	fl**ee**t	u	bl**ue**
	I	d**i**mple	U	b**oo**k
	e	d**a**te	o	sh**ow**
	E	b**e**t	O	c**augh**t
	a	c**a**t	A	f**a**ther, c**o**t

CATEGORY	PHONEME	EXAMPLE	PHONEME	EXAMPLE
	aj	s**i**de	aw	c**ou**ch
	Oj	t**oy**	*r	b**ir**d
	^	c**u**t	*	**a**live
Stops	p	**p**ut	b	**b**all
	t	**t**ake	d	**d**ice
	k	**c**atch	g	**g**ate
Flap	!	bu**tt**er		
Nasals	m	**m**ile	g~	runni**ng**
	n	**n**ap		
Fricatives	f	**f**riend	v	**v**oice
	T	pa**th**	D	**th**em
	s	**s**it	z	**z**ebra

CATEGORY	PHONEME	EXAMPLE	PHONEME	EXAMPLE
	S	**sh**ield	Z	vi**s**ion
	h	**h**ave		
Affricates	tS	**ch**ur**ch**	dZ	**j**u**dg**e
Approximants	j	**y**es	w	**w**in, **wh**ich
	r	**r**ow	l	**l**ame

Bibliography

Works Cited

Abbott, K. 2002. *Voice enabling Web applications: VoiceXML and beyond*. Berkeley, CA: Apress.

Allen, J. 1995. *Natural language understanding*. Redwood City, CA: Benjamin Cummings.

Baber, C. 1997. *Beyond the Desktop*. San Diego, CA: Academic Press.

Bailey, C.J. 1999. "You and your English grammar." In M. Celce-Murcia and D. Larsen-Freeman, eds., *The grammar book*. 2nd edition. Boston: Heinle and Heinle Publishers.

Balentine, B. 1999. Re-engineering the speech menu: A "Device" approach to interactive list-selection. In D. Gardner-Bonneau, ed., *Human factors and voice interface systems*, 213215. Norwell, MA: Kluwer Academic Publishers.

Balogh, J. 2002. Methodologies and best practices: An overview. Usability Workshop. SpeechTek 2002, New York.

Balogh, J., N. LeDuc, and M. Cohen. 2001. "Navigating the voice Web." Proceedings of UAHCI (Universal Access for Human Computer Interaction) 2001.

Boyce, S. 1999. Spoken natural language dialogue systems: User interface issues for the future. In D. Gardner-Bonnea, ed., *Human factors and voice interface systems*, 205235. Norwell, MA: Kluwer Academic Publishers.

Bransford, J.D., and M.K. Johnson. 1973. Considerations of some problems of comprehension. In W.G. Chase, ed., *Visual information processing*. Orlando, FL: Academic Press.

Broadbent, D.E. 1975. The magic number seven after fifteen years. In A. Kennedy and A. Wilkes, eds., *Studies in long term memory*. London: Wiley.

Campbell, J. 1997. Speaker recognition: A tutorial. *Proceedings of the IEEE* 85: 14371462.

CCIR-1. 1999. Attitudes to recognition accuracy. Technology Report, Dialogues Spotlight Research, Centre for Communication Interface Research, University of Edinburgh. http://spotlight.ccir.ed.ac.uk/

CCIR-2. 1999. Dialogues for speaker verification and operator hand-over. Technology Report, Dialogues Spotlight Research, Centre for Communication Interface Research, University of Edinburgh. http://spotlight.ccir.ed.ac.uk/

CCIR-3. 1999. The effects of speaker state (tone of voice) and speaker style (fast track) in dialogue prompts. Technology Report, Dialogues Spotlight Research, Centre for Communication Interface Research, University of Edinburgh. http://spotlight.ccir.ed.ac.uk/

CCIR-4. 1999. The priming effects of telephone tutorials. Technology Report, Dialogues Spotlight Research, Centre for Communication Interface Research, University of Edinburgh. http://spotlight.ccir.ed.ac.uk/

CCIR-5. 1999. User attitudes towards real and synthetic speech. Technology Report, Dialogues Spotlight Research, Centre for Communication Interface Research, University of Edinburgh. http://spotlight.ccir.ed.ac.uk/

Celce-Murcia, M., and D. Larsen-Freeman. 1999. *The grammar book*. 2nd edition. Boston: Heinle and Heinle Publishers.

Chu-Carroll, J., and J. Nickerson. 2000. Evaluating automatic dialogue strategy adaptation for a spoken dialogue system. Proceedings of NAACL (North American Chapter of the Association for Computational Linguistics) 2000, Seattle, WA.

Chu-Carroll, J., and M. Brown. 1998. User modeling and user adapted interaction. *Special Issue on Computational Models of Mixed Initiative Interaction* 8(3+4): 215253. Also appeared in S. Haller, S. McRoy, and A. Kobsa, eds., *Computational models of mixed-initiative interaction*, 4987. Boston: Kluwer Academic Publishers, 1999.

Cohen, M. 1991. Combining linguistic knowledge with statistical pattern recognition techniques for speech recognition. Keynote talk and Proceedings of Expert Systems and Their Microcomputer Applications, Ankara, Turkey.

Cohen, M. 2000. Surfing the voice Web: Issues in the design of a voice browser. Keynote talk, ASR2000, Paris.

Cohen, M. 2001. VUI design under the microscope: User centered design methodology. V-World 2001, San Diego, CA.

Cohen, M., Z. Rivlin, and H. Bratt. 1995. Speech recognition in the ATIS domain using multiple knowledge sources. Proceedings of the Spoken Language Systems Technology Workshop, ARPA, Austin, TX.

Crystal, D. 1992. *The Cambridge encyclopedia of language*. Cambridge, UK: The Cambridge University Press.

Crystal, David. 1995. *The Cambridge encyclopedia of the English language*. Cambridge, UK: The Cambridge University Press.

Daneman, M., and P.A. Carpenter. 1980. Individual differences in working memory and reading. *Journal of Verbal Learning & Verbal Behavior* 19(4): 450466.

Dumas, J., and J. Redish. 1999. *A practical guide to usability testing*. Revised edition. Exeter, UK: Intellect.

Dutoit, T. 1997. An introduction to text-to-speech synthesis. Boston: Kluwer Academic Publishers.

Dutton, R., J. Foster, and M. Jack. 1999. Please mind the doors: Do interface metaphors improve the usability of voice response services? *BT Technological Journal* 17(1): 172177.

Edgington M., A. Lowry, P. Jackson, A. Breen, and S. Minnis. 1998a. Overview of current text-to-speech techniques, part I: Text and linguistic analysis. In F.A. Westall, D. Johnston, and A. Lewis, eds., *Speech technology for telecommunications*. New York: Chapman & Hall.

Edgington M., A. Lowry, P. Jackson, A. Breen, and S. Minnis. 1998b. Overview of current text-to-speech techniques, part II: Prosody and speech generation. In F.A. Westall, D. Johnston, and A. Lewis, eds., *Speech technology for telecommunications*. New York: Chapman & Hall.

ETSI (European Telecommunications Standards Institute). 2002. *Generic spoken command vocabulary for ICT devices and services*. ETSI DES/HF-00021 v 0.0.40 (2002-05-27). www.etsi.org.

Fraser, J., and G. Gilbret. 1991. Simulating speech systems. *Computer, Speech, and Language* 5: 8199.

Furui, S. 1996. An overview of speaker recognition technology. In C.F. Lee and F. Soong, eds., *Automatic speech and speaker recognition*, 3166. Boston: Kluwer Academic.

Gardner-Bonneau, D.J. 1992. Human factors in interactive voice response applications: "Common sense" is an uncommon commodity. *Journal of the American Voice I/O Society* 12: 112.

Giangola, J. 2000. Building naturalness into prompt design. V-World 2000, Scottsdale, AZ.

Giles, H., and P. Powesland. 1975. *Speech style and social evaluation*. (European Monographs in Social Psychology). New York: Harcourt Brace.

Giles, H., and P. Smith. 1979. Accommodation theory: Optimal levels of convergence. In H. Giles and R. St Clair, eds., *Language and social psychology*, 4565. Oxford: Blackwell.

Gold, B., and N. Morgan. 2000. *Speech and audio signal processing: Processing and perception of speech and music*. New York: John Wiley & Sons.

Grice, H. P. 1975. Logic and conversation. In P. Cole and J.L. Morgan, eds., *Speech acts*, 4158. New York: Academic Press.

Halliday, M.A.K. 1994. *An introduction to functional grammar*. 2nd edition. London: Edward Arnold.

Ishihara, R. 2003. Enhancing TTS performance. Proceedings of Telephony Voice User Interface Conference, San Diego.

Jackson, E., D. Appelt, J. Bear, R. Moore, and A. Podlozny. 1991. A template matcher for robust NL interpretation. Proceedings of the Fourth DARPA Workshop on Speech and Natural Language, Pacific Grove, CA.

Jelinek, F. 1997. *Statistical methods for speech recognition*. Cambridge, MA: MIT Press.

Jurafsky, D., and J. Martin. 2000. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, NJ: Prentice Hall.

Kamm, C., D. Litman, and M.A. Walker. 1998. From novice to expert: The effect of tutorials on user expertise with spoken dialogue systems. Proceedings of ICSLP (International Conference on Spoken Language Processing) 1998, Sydney, Australia.

Kamm, C., M.A. Walker, and D. Litman. 1999. Evaluating spoken language systems. Proceedings of AVIOS (Applied Voice Input/Output Association) 1999, San Jose, CA.

Kramer, G., ed. 1994. *Auditory display: Sonification, audification, and auditory interfaces*. Proceedings Volume XVIII, Santa Fe Institute, Studies in the Sciences of Complexity. Reading, MA: Addison-Wesley.

Labov, W. 1966. *The social stratification of English in New York City*. Washington, DC: Center for Applied Linguistics.

Larson, J. 2003. *VoiceXML: Introduction to developing speech applications*. Upper Saddle River, NJ: Prentice Hall.

Manning, C., and H. Schutze. 1999. *Foundations of statistical natural language understanding*. Cambridge, MA: MIT Press.

McClelland, I., and F. Brigham. 1990. Marketing ergonomics: How should ergonomics be packaged? *Ergonomics* 33(5): 519526.

McConnell, S. 1996. *Rapid development: Taming wild software schedules*. Redmond, WA: Microsoft Press.

Melrose, S. L. 1999. *Must* and its periphrastic forms in American English usage. M.A. Thesis, UCLA. In Celce-Murcia and Larsen-Freeman, *The grammar book*. 2nd edition. Boston: Heinle and Heinle Publishers.

Miller, G. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63: 8197.

Nielsen, J. 1993. *Usability engineering*. San Diego, CA: Morgan Kaufman.

Norman, D.A. 2002. *The design of everyday things*. New York: Basic Books.

Norman, D.A., and S.W. Draper, eds. 1986. *User centered system design: New perspectives on human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Nowlin, R. 2001. VUI design under the microscope: Requirements definition. V-World 2001, San Diego, CA.

Nowlin, R. 2001. VUI design under the microscope: Tuning and validation. V-World 2001, San Diego, CA.

Oviatt, S. 1996. User-centered modeling for spoken language and multimodal interfaces. *IEEE Multimedia* 3(4): 2635.

Page, J., and A. Breen. 1998. The Laureate text-to-speech system: Architecture and applications. In F.A. Westall, D. Johnston, and A. Lewis, eds., *Speech technology for telecommunications*. New York: Chapman & Hall.

Pierrehumbert, J. 1980. The phonetics and phonology of English intonation. Doctoral dissertation, MIT.

Preece, J., Y. Rogers, and H. Sharp. 2002. *Interaction design: Beyond human-computer interaction*. New York: John Wiley and Sons.

Quirk, R., and S. Greenbaum. 1973. *A concise grammar of contemporary English*. New York: Harcourt Brace Jovanovich.

Rabiner, L. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77: 257286.

Rabiner, L., and B. Juang. 1993. *Fundamentals of speech recognition*. Englewood Cliffs, NJ: Prentice Hall.

Raman, T. 1997. *Auditory user interfaces*. Boston: Kluwer Academic.

Raskin, J. 2000. *The humane interface*. Boston: Addison-Wesley.

Reeves, B., and C. Nass. 1996. *The media equation*. Stanford, CA: Center for the Study of Language and Information.

Reynolds, D., and L. Heck. 2001. Speaker verification: From research to reality. International Conference on Acoustics, Speech, and Signal Processing. Tutorial. Salt Lake City, Utah.

Richards, J. 1980. Conversation. *TESOL Quarterly* XIV(4).

Rubin, J. 1994. *Handbook of usability testing*. New York: John Wiley and Sons.

Rudnicky, A., and W. Xu. 1999. An agenda-based dialog management architecture for spoken language systems.

IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, CO.

Schiffrin, D. 1987. *Discourse markers*. Cambridge, UK: Cambridge University Press.

Schiffrin, D. 1998. *Approaches to discourse*. Cambridge, MA: Blackwell Publishers.

Schumacher, R.M., Jr., M.L. Hardzinski, and A.L. Schwarz. 1995. Increasing the usability of interactive voice response systems: Research and guidelines for phone-based interfaces. *Human Factors* 37(2): 251264.

Selkirk, E. 1995. Sentence prosody: Intonation, stress, and phrasing. In John A. Goldsmith, ed., *Phonological theory*. Cambridge, MA: Blackwell Publishers.

Seneff, S., and J. Polifroni. 2000. Dialogue management in the Mercury flight reservation system. Presented at Satellite Dialogue Workshop, ANLP-NAACL, Seattle.

Sharma, C., and J. Kunins. 2002. *VoiceXML: Strategies and techniques for effective voice application development with VoiceXML 2.0*. New York: John Wiley and Sons.

Sheeder, T. 2001. VUI design under the microscope: Detailed design. V-World 2001, San Diego, CA.

Sheeder, T. 2001. VUI design under the microscope: High-Level design. V-World 2001, San Diego, CA.

Sheeder, T., and J. Balogh. 2003. Say it like you mean it: Priming for structure in caller responses to a spoken dialog system. *International Journal of Speech Technology* 6(3): 103111.

Shneiderman, B. 1998. Designing the user interface: Strategies for effective human-computer interaction. 3rd Edition. Reading, MA: Addison-Wesley.

Soukup, B. 2000. Y'all come back now, y'hear: Language attitudes in the United States towards Southern American English. MA thesis, University of Vienna.

TSSC (Telephone Speech Standards Committee). 2000. Universal commands for telephony-based spoken language systems. Telephone Speech Standards Committee: Common Dialog Tasks Subcommittee. www.acm.org/sigchi/bulletin/2000.2/telephonepaper.pdf

van Santen, J., R. Sproat, J. Olive, and J. Hirschberg. 1997. *Progress in speech synthesis*. New York: Springer Publishers.

Weinschenk, S., and D. Barker. 2000. *Designing effective speech interfaces*. New York: John Wiley & Sons.

Weintraub, M., H. Murveit, M. Cohen, P. Price, J. Bernstein, G. Baldwin, and D. Bell. 1989. Linguistic constraints in hidden Markov based speech recognition. International Conference on Acoustics, Speech and Signal Processing, Glasgow, Scotland.

Wickelgren, W.A. 1974. Size of rehearsal group and short-term memory. *Journal of Experimental Psychology* 68: 413419.

Yankelovich, N., G.A. Levow, and M. Marx. 1995. Designing SpeechActs: Issues in speech user interfaces. In I.R. Katz, R. Mack, and L. Marks, eds., *Human Factors in Computing Systems*. CHI 1995 Conference Proceedings, 369376.

Works Consulted

Anderson, J.R. 1990. *Cognitive psychology and its implications*. New York, NY: W.H. Freeman, 167170.

Attwater, D.J., and S.J. Whittaker. 1999. Large-vocabulary data-centric dialogues. *BT Technology Journal* 17(1) 149159.

Balentine, B. 2003. The power of the pause. *Speech Recognition Update* 118.

Balogh, J. 2001. Strategies for concatenating recordings in a voice user interface: What we can learn from prosody. Extended Abstracts, CHI (Computer Human Interface) 2001, 249250.

Bowen, J. 1975. *Patterns of English pronunciation*. Rowley, MA: Newbury House Publishers.

Cowley, C., and D. Jones. 1992. Synthesized or digitized? A guide to the use of computer speech. *Applied Ergonomics* Jun. 23 (3): 172176.

Delogu, C., S. Conte, and C. Sementina. 1998. Cognitive factors in the evaluation of synthetic speech. *Speech Communication* 24(2): 153168.

Francis, A., and H. Nusbaum. 1999. The effect of lexical complexity on intelligibility. *International Journal of Speech Technology* 3(1): 1525.

Fucci, D., M. Reynolds, R. Bettagere, and M. Gonzales. 1995. Synthetic speech intelligibility under several experimental conditions. *AAC: Augmentative & Alternative Communication* Jun. 11 (2): 113117.

Gong, L., and J. Lai. 2001. Shall we mix synthetic speech and human speech? Impact on users' performance, perception, and attitude. Proceedings of CHI (Computer-Human Interface) 2001, Seattle, WA. 158165.

Higginbotham, D., A. Drazek, K. Kowarsky, and C. Scally. 1994. Discourse comprehension of synthetic speech delivered at normal and slow presentation rates. *AAC: Augmentative & Alternative Communication* Sept. 10 (3): 191202.

Hoover, J., J. Reichle, D. Van Tasell, and D. Cole. 1987. The intelligibility of synthesized speech: ECHO II versus VOTRAX. *Journal of Speech & Hearing Research* Sep. 30 (3): 425431.

Hudson, R. 1980. *Sociolinguistics*. London: Cambridge University Press.

James, F. 1996. Presenting HTML structure in audio: User satisfaction with audio hypertext. Proceedings of ICAD (International Conference on Auditory Display) 1996, Palo Alto, CA, 97103.

Kangas, K., and G. Allen. 1990. Intelligibility of synthetic speech for normal-hearing and hearing-impaired listeners. *Journal of Speech & Hearing Disorders* Nov. 55(4): 751755.

Lai, J., D. Wood, and M. Considine. 2000. The effect of task conditions on the comprehensibility of synthetic speech. Proceedings of CHI 2000, 321328. The Hague, Netherlands. New York: ACM.

Lamel, L., S. Rosset, J. Gauvain, S. Bennacef, M. Garnier-Rizet, and B. Prouts. 1998. The LIMSI ARISE system. Proceedings of IVTTA (Interactive Voice Technology for Telecommunication Applications) 1998, Turin, Italy, 209214.

Lavelle, C-A., M. de Calmes, and G. Perennou. 1998. A study of users' behaviors in different states of a spontaneous oral dialogue with an automatic inquiry system. IEEE, Proceedings of IVTTA (Interactive Voice Technology for Telecommunication Applications) 1998, Turin, Italy, 118123.

McInnes, F.R., D.J. Attwater, D. Edgington, M.S. Schmidt, and M.A. Jack. 1999. User attitudes to concatenated natural speech and text-to-speech synthesis in an automated information service. Eurospeech 1999, Proceedings of Speech Technology Symposium, Budapest.

Nass, C., Y. Moon, and N. Green. 1997. Are machines gender neutral? Gender-stereotypic responses to computers with voices. *Journal of Applied Social Psychology* May 27 (10): 864876.

Oshrin, S., and J. Siders. 1987. The effect of word predictability on the intelligibility of computer synthesized speech. *Journal of Computer-Based Instruction* 14(3): 8990.

Paris, C., M. Thomas, R. Gilson, and J. Kincaid. 2000. Linguistic cues and memory for synthetic and natural speech. *Human Factors* 42(3): 421431.

Paris, C., R. Gilson, M. Thomas, and N. Silver. 1995. Effect of synthetic voice intelligibility on speech comprehension. *Human Factors* 37(2): 335340.

Pinker, Steven. 1994. *The language instinct*. New York: William Morrow.

Potjer, J., A. Russel, L. Boves, and E. den Os. 1996. Subjective and objective evaluation of two types of dialogues in a call assistance service. Proceedings of IVTTA (Interactive Voice Technology for Telecommunications Applications) 1996, Basking Ridge, NJ, 8992.

Ralston, J., D. Pisoni, S. Lively, and B. G. Greene. 1991. Comprehension of synthetic speech produced by rule: Word monitoring and sentence-by-sentence listening times. *Human Factors* 33(4): 471491.

Reynolds, M., C. Isaacs-Duvall, B. Sheward, and M. Rotter. 2000. Examination of the effects of listening practice on synthesized speech comprehension. *AAC: Augmentative & Alternative Communication* 16(4): 250259.

Reynolds, M., Z. Bond, and D. Fucci. 1996. Synthetic speech intelligibility: Comparison of native and non-native speakers of English. *AAC: Augmentative & Alternative Communication* 12(1): 3236.

Rosch, E. 1976. Classification of real-world objects: Origins and representations in cognition. In S. Erlich and E. Tulvings, eds., *La Mémoire sémantique*. Paris: Bulletin de Psychologie.

Rosenthal, M. 1974. The magic boxes: Pre-school children's attitudes toward black and standard English. *Florida F. L. Reporter* 210: 55-93.

Schwab, E., H. Nusbaum, and D. Pisoni. 1985. Some effects of training on the perception of synthetic speech. *Human Factors* 27(4): 395408.

Smither, J. 1993. Short term memory demands in processing synthetic speech by old and young adults. *Behaviour & Information Technology* 12(6): 330335.

Stern, S., J. Mullennix, C. Dyson, and S. Wilson. 1999. The persuasiveness of synthetic speech versus human speech. *Human Factors* 41(4): 588595.

Stifelman, L.J., B. Arons, C. Schmandt, and E. Hulteen. 1993. VoiceNotes: A speech interface for a hand-held voice

notetaker. Proceedings of INTERCHI 1993, ACM, New York.

Sutton, B., J. King, K. Hux, and D.R. Beukelman. 1995. Younger and older adults rate performance when listening to synthetic speech. *AAC: Augmentative & Alternative Communication* 11(3): 147153.

Venkatagiri, S. 1994. Effect of sentence length and exposure on the intelligibility of synthesized speech. *AAC: Augmentative & Alternative Communication* 10(2): 96104.

Vromans, B., R.J. van Vark, B. Rueber, and A. Kellner. Extending the SUSI System with negative knowledge. Proceedings of Eurospeech 1999, Budapest.

Waterworth, J.A. 1983. Effect of intonation form and pause durations of automatic telephone number announcements on subjective preference and memory performance. *Applied Ergonomics* 14(1): 3942.

Whalen, D., C. Hoequist, and S. Sheffert. 1995. The effects of breath sounds on the perception of synthetic speech. *Journal of the Acoustical Society of America* 97(5, pt. 1): 31473153.

Yankelovich, N. (in press). Using natural dialogs as the basis for speech interface design. In S. Luperfoy, ed., *Automated spoken dialog systems*. Cambridge, MA: MIT Press.