

Chapter 1

IVR USABILITY ENGINEERING USING GUIDELINES AND ANALYSES OF END-TO-END CALLS

Bernhard Suhm

*BBN Technologies, AVOKE Call Center Analytics group, bsuhm@bbn.com
10 Moulton Street, Cambridge, MA 02138*

Abstract: While speech offers unique advantages and opportunities as an interface modality, the known limitations of speech recognition technology and cognitive limitations of spoken interaction amplify the importance of usability in the development of speech applications. The competitive business environment, on the other hand, requires sound business justification for any investment in speech technology and proof of its usability and effectiveness. This chapter presents design principles and usability engineering methods that empower practitioners to optimize both usability and ROI of telephone speech applications, frequently also referred to as telephone Voice User Interface (VUI) or Interactive Voice Response (IVR) systems. The first section discusses limitations of speech user interfaces and their repercussions on design. From a survey of research and industry know-how a short list of guidelines for IVR design is derived. Examples illustrate how to apply these guidelines during the design phase of a telephone speech application. The second section presents a data-driven methodology for optimizing usability and effectiveness of IVRs. The methodology is grounded in the analysis of live, end-to-end calls - the ultimate field data for telephone speech applications. We will describe how to capture end-to-end call data from deployed systems and how to mine this data to measure usability and identify problems. Leveraging end-to-end call data empowers practitioners to build solid business cases, optimize ROI, and justify the cost of IVR usability engineering. Case studies from the consulting practice at BBN Technologies illustrate how these methods were applied in some of the largest US deployments of automated telephone applications.

Keywords: telephone speech application, speech user interface, design principles, best practices, usability engineering, end-to-end call, ROI

1. IVR DESIGN PRINCIPLES AND GUIDELINES

Usability engineering methods leverage design knowledge and analysis of usability data to optimize designs. Research in the field of human-computer interaction has created an array of usability engineering methods, ranging from the traditional usability tests in the lab to so-called discount usability engineering (Nielsen, 1993). Since about 1990, ethnographic methods, such as contextual design (Holtzblatt and Beyer, 1998), have been applied, especially to early stages of design. With the maturation of speech technologies, practitioners have adapted many of these methods to the design and optimization of speech user interfaces.

Since call center applications thus far represent the largest market for speech applications, IVR applications¹ have received considerable attention from practitioners and researchers close to the speech industry. A search of the literature reveals publications on basic design problems in IVRs, such as prompting in touch-tone IVRs and guidelines for menu design. Other research applies standard usability evaluation methods to research speech user interfaces (e.g., Bennacef, Devillers, Rosset and Lamel, 1996) and to commercial IVRs (Edwards, Quinn, Dalziel, and Jack, 1997; Delogu, Di Carlo, Rotundi and Satori, 1998). Usability design and re-engineering know-how for IVRs ranges from specific design issues of touch-tone IVRs (Halstead-Nussloch, 1989; Resnick and Virzi, 1995) to comprehensive collections of design guidelines for touch-tone and speech-enabled IVRs in recent books by some veterans of the industry (Balentine and Morgan, 1999; Cohen, Giangola and Balogh, 2004).

As knowledge on speech user interface and IVR design has been accumulating, consolidating this knowledge is becoming increasingly important. “Best practices” have become a hot topic lately in industry communication. To further the process of consolidating design knowledge, we have developed a framework for speech user interface design, which is described in the first section of this chapter. We decided to include a description of the framework in this chapter for two reasons. First, the framework can provide guidance in early stages of speech interface design by making designers cognizant of the various limitations of speech user interfaces. Second, we have employed the framework to compile a short list

¹ For simplicity, this chapter employs the term “IVR” to refer to both touch-tone and speech-enabled IVRs. Speech-enabled IVRs are also called telephone “voice user interfaces” (VUI). Technically, the latter refers to a class of human-computer interfaces, and may be more intuitive to readers with a human-computer interaction background, while the former refers to a specific instance of such an interface, and should be very familiar to most readers with a background in call centers.

of IVR design guidelines that practitioners can employ in various usability engineering methods, such as heuristic evaluation. In essence, these design principles and guidelines should empower designers to arrive at acceptable initial designs quickly – designing with a “broad brush” to use a painting metaphor.

While applying best practices can often improve IVR usability, the measurement and validation of those improvements can be quite difficult, especially in the context of a production IVR with many different tasks and a wide range of callers. Furthermore, when IVR design methods yield different plausible designs, it is often impossible to decide which design works best just by applying guidelines without empirical evaluation. In essence, these issues constitute the “fine print” of IVR design that guidelines and principles cannot address. To empower practitioners to optimize designs effectively, the second focus of this chapter is on a data-driven methodology for IVR usability engineering. The methodology presented in section 2 is unique because it leverages the ultimate field data for IVRs - end-to-end calls into a live call center, and because the methodology provides the basis for optimizing IVR cost-effectiveness. Unlike most other commonly employed usability methods, in addition to improving IVR usability this methodology empowers practitioners to build business cases for IVR usability engineering, and to maximize ROI for the call center.

1.1 A Taxonomy of Limitations of Speech User Interfaces

Good design is as much about creativity as about knowing the specific properties of the used material or media, which includes skillfully dealing with the limitations of the designed object. Knowing about the limitations and how to circumvent them is particularly important in the design of speech user interfaces. Such limitations represent the causes of errors and usability problems determine the key challenges that design most solve, and their understanding provides leverage for better designs. Therefore, as one foundation for our framework for speech user interface design, we developed a taxonomy of speech user interface limitations. Based on a comprehensive survey of relevant literature and expert knowledge, and extending similar taxonomies from other works (Novick, Hansen, Sutton and Marshall, 1999; Shneiderman, 2000), we identified six broad categories of limitations: speech recognition, spoken language, environment, human cognition, user, and hardware. Table 1-1 shows the limitation categories, their definition, and the specific limitations that we have identified in each category.

The following subsections discuss those three of the six limitations categories in more depth that capture the most unique properties of speech user interfaces: recognition, spoken language, and human cognition. The other three categories – environment, user, and hardware – are well-understood from other classes of interfaces, and a complete discussion of all categories is beyond the scope of this chapter. Each specific limitation (shown in the third column in Table 1-1) further expands into a list of specific design problems. Due to the growing body of speech interface knowledge, this taxonomy is necessarily a work in progress. The author believes that the structure of this taxonomy is solid and can accommodate new insights incrementally.

Table 1-1. Taxonomy of limitations of speech user interfaces

Limitation Category	Definition	Specific Limitations
Speech Recognition	Limitations of (current) speech recognition technology	Errors Finite vocabulary Grammar Acoustic model
Spoken Language	Limitations arising from characteristics of spoken language	Spontaneous Public Natural turn-taking protocol Anthropomorphism Limited expressive power
Environment	Disturbances from the user's environment	Noise Multiple people speaking Interruptions
Human Cognition	Properties of the human cognitive system	Sequential and slow Working memory capacity Low persistence Competition with verbal processing
User	Differences between and preferences among users	Task knowledge Expert / novice Speech competence
Hardware	Properties of the hardware used to implement a speech user interface	Channel Microphones Computing platform

1.1.1 Limitations of Speech Recognition

At the level of recognition technology, four categories of recognition limitations are distinguished: finite vocabulary, language model, acoustic model, and finally, recognition errors. The first three arise from the basic

components of speech recognizers. Expanding the corresponding rows from Table 1-1 (shown in grey), Table 1-2 lists specific design problems that arise from these limitations. Since the limitations of speech recognition are generally well-known, the focus in this subsection is not on the limitations themselves, but on how limitations are related to significant design problems in speech user interface design. The following paragraphs discuss a few specific design problems for each limitation of speech recognition technology.

- *Recognition Errors:* Since automatic speech recognition is not perfect (and probably never will be), any speech interface has to manage recognition errors. Minimizing the error rate is primarily an engineering problem. The many interactions between recognition parameters make it a non-trivial problem that requires speech recognition expertise. But even with optimal setup of the recognizer recognition errors remain. Graceful recovery from those errors is difficult for several reasons, including that it is often difficult to detect errors, that errors can be repeated or even cascade into error spirals (Karat, Halverson, Horn and Karat, 1999), and that hypertalk exacerbates recognition problems. Hypertalk refers to speech that is over-enunciated and spoken more slowly and loudly, in an attempt to overcome communication problems in human dialog. While hypertalk is effective in recovering communication problems during human-to-human conversation, hypertalk further degrades speech recognition performance, rather than helping the recognizer (Soltau and Waibel 2000). The last design problem mentioned in Table 1-2 refers to the confusion and frustration that counterintuitive recognition errors can cause in users. Speech recognizers do not work like human ears, therefore words that may “sound” alike to the recognizer sometimes sound very different to the human ear.
- *Finite Vocabulary and Grammar:* Most speech systems recognize only words from a pre-defined, finite vocabulary. All speech input is mapped to (a sequence of) words within that vocabulary. Words that are not covered by the finite vocabulary, called out-of-vocabulary words, inevitably lead to recognition errors. Related to the problem of which words to include in the vocabulary is the problem of grammar coverage. Since the grammar specifies which sequences of words are acceptable to the recognizer, the grammar must cover the various ways users may formulate their responses to a certain prompt. What makes the problem of vocabulary and grammar coverage difficult is the fact that increasing vocabulary size and grammar complexity must be balanced against making automatic recognition more difficult by adding confusability between words and word sequences. Generally, a small vocabulary and

tight grammar help to achieve high recognition accuracy. The next problem mentioned in Table 1-2 occurs in the context of statistical grammars. Statistical grammars are employed to recognize responses to open-ended prompts, such as in AT&T's famous "How may I help you?" research system (Gorin, Parker, Sachs and Wilpon, 1996). There is more on this topic in Chapter 3 later in this book. Any mismatch between actual user input and the data that are used to train the statistical language model degrades recognition performance. At a very basic level, a recognizer trained for dictation will perform rather poorly on recognizing telephone dialogs. But mismatches can be rather subtle, for example, changes in word choice due to regional accents, or shifts over time in how customers describe their reason for calling, in response to a prompt like "How may I help you?".

- *Acoustic Models*: Acoustic models of speech recognizers have only a limited capability to model variability inherent in speech. Therefore, any additional variability in the acoustic signal makes automatic recognition more difficult. Variability in the acoustic signal arises from many sources: voices of different users, regional or foreign accents, fast and slow speakers, co-articulation of words, confusable words, as well as hypertalk, discussed earlier in the context of recognition errors.

Table 1-2. Design problems arising from specific limitations of speech recognition

Limitation	Design Problems
Errors	Minimize errors How to detect errors Alleviate (avoid) repeated errors Cascading errors in correction Hypertalk exacerbates errors Errors are often not intuitive to users
Finite Vocabulary	Out-of-vocabulary words cause recognition errors Trade-off coverage with confusability and speed
Grammar	What people say is often difficult to predict Mismatch between training data and user input
Acoustic Models	Multiple speakers and accents Fast and slow speakers Co-articulation Confusable words Acoustic variability in "spontaneous" speech Poor quality of audio input Speech (endpoint) detection Distorted speech due to barge-in or background speech

Furthermore, any reduction in quality of the input signal will degrade performance, including noise, inaccurate speech endpoint detection (which refers to the problem of detecting where speech begins and ends in the audio signal), and any distortions in the signal, such as from speaking over other sounds (also known as “Lombard effect”). This explains why barge-in – commonly employed in many speech-enabled IVRs – is undesirable from a recognition point of view. While acceptable for certain dialog elements in repeat-use applications², barge-in is generally undesirable from a spoken language point of view, because it disrupts conversation and we therefore avoid it even in human conversation. This leads us to limitations inherent in spoken language.

1.1.2 Limitations of Spoken Language

Patterns and behaviors learned in human conversation “intuitively” carry over to spoken interaction with computers and other automated systems. Recent research shows that users even attribute human traits to interactive media, such as speech interfaces (Reeves and Nass, 1996), which is commonly referred to as anthropomorphism. Certain characteristics of spoken language must therefore be considered in speech interface design. We have identified the following characteristics of spoken language as being relevant to speech interface design: that it’s spontaneous, its public character, the rules of turn-taking in human conversation, anthropomorphism (describing the tendency of users to assign human qualities to non-human speech systems), and the limited expressive power of language. While spontaneity, turn-taking, and anthropomorphism can be leveraged beneficially in speech user interface design, these characteristics frequently lead to usability problems, unless mitigated through careful design. Table 1-3 shows specific design problems that arise from these five limitations of spoken language. The following paragraphs discuss in more detail how the characteristics of spoken language lead to design problems.

The “spontaneous” character of spoken language manifests itself in properties like redundancy and disfluencies. Although very typical of human conversation, these behaviors are not suitable to goal-oriented, efficient communication, as required when performing goal-oriented tasks like device

² Some studies (Cohen, Giangola, Balogh, 2004) have shown that after repeated use, and when encouraged to do so, callers will barge into prompts, and increasingly so with increased prompt length. Hence, leveraging barge-in in a controlled fashion when designing speech systems for frequent use may be appropriate. However, many IVR applications are used very infrequently by the majority of callers.

control, data entry, or dictation. Disfluencies make spontaneous speech more difficult to recognize than spoken responses from a directed dialog.

Despite the “spontaneous” character of human conversation, some users are surprised if automated systems emulate the spontaneous form of dialog in open-ended prompts, such as “How may I help you?” Typically, this design problem is solved by providing callers with examples when they do not respond to such a prompt.

Table 1-3. Limitations (characteristics) of spoken language

Spoken Language Characteristic	Specific Design Problem
“Spontaneous” character	Chatty behavior leads to inefficient communication, not suitable for command and control, data entry, dictation
	Some users are surprised by open-ended prompts, and confused about how to respond
Public	Speech can be heard by others: no privacy, and possible disturbance
Turn-taking	Users abide by turn-taking protocol of human conversation
Anthropomorphism	Degree of interface personification (“persona”)
	Complex nuances of human conversation are not conducive to machines
	Raised user expectations lead to backlash when disappointed
Limited expressive power	Need to resolve ambiguities to determine meaning
	Difficult to refer to objects in physical environment
	Difficult to describe locations or spatial manipulation precisely

Spoken language is public - speech can be heard by others. Speech interaction can thus lead to issues of privacy, for example, if sensitive information (for example, account and PIN numbers) needs to be communicated.

Linguistic research (Sacks and Schegloff, 1974) revealed that turn-taking in human conversation is governed by certain rules, in particular, that interrupting the speaker is generally considered impolite. Users will intuitively follow the same rules when communicating with a speech system, hence speech dialog design should generally abide by them. The common, yet problematic practice of encouraging callers to interrupt the system at any time, ignores this basic rule of turn-taking. Other reasons why relying on barge-in can lead to usability problems include lower recognition accuracy (as discussed in the previous subsection on limitations of recognition) and the tendency of designers to use it as an excuse for overly long prompts.

Anthropomorphism has become a controversial topic in speech user interface design, but a discussion that would do this topic justice is beyond the scope of this chapter. Our table above (Table 1-3) lists some generally accepted design problems relating to anthropomorphism.

Last in our list of limitations of spoken language is its limited expressive power. It is well-known that spoken language can be ambiguous. For speech interface design, this means that techniques to resolve ambiguity are required when extracting meaning from spoken words. Furthermore, designers need to be aware that prompts or spoken output can lead to user confusion and misunderstanding. Other symptoms of the limited expressive power of language include the difficulty of referring to objects in the physical environment, and describing locations or movements precisely.

1.1.3 Human Cognition

Summarizing well-known research in cognitive psychology, Table 1-4 lists some limitations of human cognition that apply to speech interaction.

Table 1-4. Cognitive limitations relevant to speech user interface design

Specific Limitation	Design Problem
Speech is sequential	Slowness of spoken output
Limitations of working memory	Limited “short-term memory” (STM) capacity of 7+-2 chunks Low persistence Primacy versus recency
Speech competes with verbal processing	Speaking competes with other verbal processing - if they occur simultaneously

The sequential nature of processing speech in the human brain makes spoken dialog a slow means of communication.

Limitations of the human “working memory” lead to several other design problems. First, “short-term” memory is limited to 7+-2 “chunks”. This limited short-term memory capacity severely limits how much information a speech interface can present to a user. Aggravated by the low persistence of speech in working memory, this is probably the most significant limitation of speech interfaces, compared to graphic user interfaces. Furthermore, the low persistence of speech has other important repercussions, most importantly that users cannot remember long prompts and instructions. Avoiding instructions and keeping prompts concise are design guidelines that arise from this limitation of spoken language.

Moving on to the next limitation from Table 1-4, primacy versus recency refers to the observation that we typically remember the beginning and the end of a longer list, but we have trouble remembering everything else.

Therefore, key information should be placed near the beginning or the end of a prompt.

Finally, there is evidence that speaking interferes with other verbal processing, such as remembering information and thinking (Shneiderman 2000). Further research is necessary to understand the interaction of speech with other cognition in more depth, and what other implications the limited capacity of working memory has on speech interaction.

1.2 Towards Best Practices for IVR Design

“Best practices” have recently become a hot topic in the speech and IVR industries. Over the past decade, a significant body of knowledge relevant to IVR design has been established, and the process of consolidating this knowledge has begun. This subsection presents a method for organizing this knowledge using a database, and for using the database to compile design guidelines for various classes of speech user interfaces. Most relevant to this chapter, we present a short list of ten guidelines for IVR design that we derived from our database of speech user interface design knowledge, and we demonstrate the applicability and usefulness of these guidelines using examples from our IVR design consulting experience.

1.2.1 A Database for Speech User Interface Design Knowledge

Table 1-5. Sample from the database of speech user interface design, showing some solutions to the limitation "recognition errors". Fields that are left blank can take on any value

Solutions	Modalities	Domain	Interaction Style	Solution Type	Source
Careful configuration of the recognizer				Recognizer configuration	Common know-how
Design prompts to elicit brief responses				Interaction Design	(Resnick and Virzi, 1995)
Adopt speaking style that minimizes error			Interactive	User training	(Newman, 2000)
Optimize work style for error correction		Dictation	Interactive	User training	(Karat, Horn et al., 2000)
Offer alternative modalities for error correction	GUI, buttons, keyboard		Interactive	Interaction Design	(Suhm, Meyers and Waibel 1999)

Knowledge relevant to speech user interface design is spread among vendors, consultants, speech recognition and usability researchers. To organize this knowledge, we developed a database for speech user interface design solutions and best practices. Realizing that (most) speech interface design problems are rooted in some limitation of speech, we organized this knowledge as “solutions” to specific design problems arising from limitations of speech, and used the taxonomy of speech limitations described in the previous subsection as the first index to the solution database. Furthermore, to be able to relate solutions to specific applications, we developed a set of solution attributes as orthogonal indices to the database. Table 1-5 shows sample content of the database for the first few solutions to the limitation of “recognition errors”. Due to space limitations, only the most important solution attributes are shown: the input/output modalities required by a design or solution, the domain, the interaction style (dialog-oriented, interactive, or non-interactive), the solution type (recognition algorithm, recognizer configuration, interaction design, and user training), and the source (reference that describes the solution). Based on our preliminary survey of the relevant literature, the database currently contains 140 specific design problems and solutions.

1.2.2 Compiling Guidelines for IVR Design

One application of the framework and solution database described above is the generation of lists of design guidelines for broad classes of speech user interfaces (SUIs). Guidelines can be compiled following a three-step process. First, the solution database is queried to obtain lists of “solutions” to specific design problems of the class of SUI. Second, expert knowledge is employed to reduce this (typically rather long) list to a short list of candidate guidelines. Third, the list of candidate guidelines can be developed into “best practices” by peer review and empirical evaluation.

We employed this method to generate a list of ten proposed guidelines for IVR design, shown in Table 1-6. A similar version of these guidelines has been presented previously (Suhm, 2003). Not to be understood as a complete representation of all applications of these guidelines, the table shows one specific design solution as an example for how to apply the guidelines, and how they relate to limitations of speech user interfaces discussed previously in subsection 1.1.

Clearly, this short list of ten design guidelines doesn’t cover all issues relevant to IVR design. Many other design issues are known and must be considered. We refer to the literature for a more comprehensive discussion than this chapter can offer.

Guidelines can be applied at different levels and in different stages of design. For the purposes of the discussion in this chapter, we are concerned about high-level design (i.e., the structure of the dialog, the call flow logic) and low-level (prompt) design. Guidelines 1, 2, 9, and 10 primarily apply to high-level design, and the other guidelines primarily to prompt design.

Table 1-6. Ten guidelines for IVR design

#	Guideline	Example specific design solution	Corresponding limitation(s)
1	Keep it simple	Keep lists or menus to 4-7 items	Limited capacity of working memory (human cognition)
2	Design for error	Employ yes/no queries to stabilize dialog during disambiguation and error correction	Errors (recognizers), ambiguity (spoken language)
3	Carefully control the amount of spoken output	Keep prompts short, especially opening instructions	Sequential nature & low persistence of speech (cognition)
4	Structure dialog the way users think	Word menu options such that they are clearly distinguished in the users' mind	Ambiguity (spoken language)
5	Minimize acoustic confusability	Resist temptation to include too many variants of some word in the vocabulary	Acoustic models (recognizer)
6	Abide by natural turn-taking protocol	Design prompts that encourage natural turn-taking (instead of relying on barge-in)	Turn-taking (spoken language)
7	Coach a little at a time	Use examples in error/timeout reprompt, especially after open-ended prompts	Limited STM capacity and low persistence (cognition)
8	Offer alternative input modalities	Offer touch-tone keypad as alternative to speech for any digit input, after errors, and for input that's sensitive to privacy	Repeated errors (recognizers), public (spoken language)
9	Carefully select the appropriate persona	Professional applications should employ professional personas	Anthropomorphism (spoken language)
10	Follow proven design process	Employ data-driven usability engineering	N/A

Guidelines such as these are useful for various standard usability engineering methods in various design stages, including high-level design, detailed design, design reviews, and heuristic evaluations.

1.2.3 Applying IVR Design Guidelines in Practice

To illustrate our guidelines below we provide examples from our usability assessments of call center IVRs. Select guidelines will be further substantiated with empirical data.³

Guideline #1: Keep it simple

Due to the limited capacity of working memory and the sequential and non-persistent nature of speech, a broad principle is to keep the dialog simple. For example, it is widely accepted to limit spoken menus or lists to 4-7 options.

Temptations to violate this guideline frequently arise when trying to squeeze out some additional call center automation by offering long lists of services. Below are two examples from different deployments at large telecommunication providers:

- a) From a wireless service provider: *You can say voicemail information, coverage area, credit for a dropped call, problem placing or receiving calls, handset problems, questions about 3G, or other options. To hear this list again, say "repeat". For more information, say "help".*
- b) From an online service provider: *Here's a list for you to choose from ... Dial-up, ISDN, T1, Frame Relay, WIFI, Business Security, Website Services, Business E-mail, Fios and DSL.*

Both prompts present too many options to the caller. The caller's short-term memory is filling up after the fourth option, leading to confusion and in some cases the wrong selection. But can design deal with business requirements that dictate these distinctions? One trick that helps with simplification is to focus on the few most frequently requested options. In most cases, the menu will be more effective by presenting fewer options, and having the other options handled by the default or a follow-up menu. Applying this trick, and using the knowledge that only call credit, coverage inquiries, and handset problems are requested of more than 10% of the callers, an improved version of prompt a) might read: *You can say "credit for a dropped call", "coverage area", "handset problems", or "other options".* Callers choosing "other options" either can be routed to the default agent (especially if other options represent only a small percentage of all inquiries) or to a follow-up menu. Knowledge about how often callers need specific options from a menu frequently can be inferred from the distribution of call

³ In this subsection, literal verbiage from prompts is highlighted in italics, along with the subheadings.

reasons. Section 2.3.1 later in this chapter describes how to obtain the call reason distribution from end-to-end call data.

This guideline applies much more broadly. For high-level design, the “Keep It simple” guideline challenges the designer to minimize the number of steps that a caller must go through to complete a task. One means to achieve this goal is to leverage backend information before prompting callers to provide it. Illustrating this trick, Figure 1-1 shows a design that leverages the backend account database to reduce confirmation of account numbers entered by callers. In the design shown in the figure, the account number capture and validation requires only one user interaction in the majority of calls – namely if the number was captured (entered or recognized) correctly and if the account is actually in the database. The additional confirmation step is required only if there is no match with the database, and re-entry of the information only if the caller indicates that the number has been captured incorrectly, for example, due to a recognition error. A different, commonly employed yet suboptimal design forces the caller to confirm what they entered before attempting to match the information with the account database. Such a design requires at least two interactions with all callers, one more than the design shown in Figure 1-1.

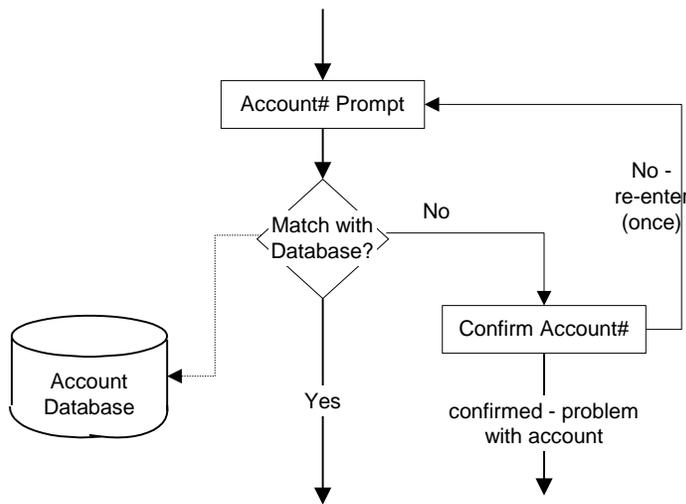


Figure 1-1. “Keeping it simple” applied to minimizing the number of steps to enter and validate an account number

Guideline #2: Design for error

Discussing error correction strategies is beyond the scope of this chapter. Principles and techniques for effective error recovery dialogs are discussed in the literature (e.g., Balentine and Morgan, 1999; Cohen, Giangola and Balogh, 2004).

Guideline #3: Carefully control the amount of spoken output

The amount of spoken output to the user must be carefully controlled, because callers quickly forget what they hear (cf. low persistence of speech) and because spoken output is slow. Many call center IVR applications violate this guideline by opening the dialog with lengthy instructions. Research shows that callers lose attention after 10-15 seconds of listening to a prompt. Barge-in does not justify lengthy instructions and prompts. While there are few situations where barge-in occurs, such as repeat callers and certain list selection designs, in most applications, the majority of callers will not interrupt the prompt - as discussed in section 1.1.2 earlier and Guideline #6 below. Therefore, good IVR design should always aim for concise prompts: not only in the instruction, but in any prompt, including information readouts.

Guideline #4: Word options the way users think

Wording menu options is notoriously difficult. Simplicity (see guideline #1 above) and clarity are crucial. Frequent mistakes include:

1. Options that are confusing to the caller, because they overlap in meaning, or because the distinctions are not meaningful to the caller.
2. Use of call center or technical jargon.
3. Not ordering options from specific to general, and from more frequent to less frequent.

Consider the following prompt in terms of overlap between options: *For your account balance or past or future payments, say "account information". If you want to make a payment, say "payment information". If you're calling about your bill or statement, say "billing information". Or you can ask for "other option".*

Isn't your head spinning just reading through this prompt, let alone hearing it on the phone? The problem is that the distinction between payment and billing information is unclear and confusing to callers. But many deployed IVRs attempt to make this artificial distinction! An improved prompt might read: *Please tell me what you'd like to do: check your balance, make a payment, or discuss a billing question. For other requests, say "other."*

Whether menu options are confusing to callers is sometimes difficult to determine. A fail-proof method for diagnosing such problems is a routing

analysis based on end-to-end calls. While a description of this analysis is beyond the scope of this chapter, the next section presents methods for capturing and analyzing end-to-end calls. Below is an example of a flawed main menu from an IVR application that we assessed:

To make payment arrangements, press 1.

To order new service or check an order already placed, press 2.

To add, change, move, or disconnect service, press 3.

For billing questions and all other requests, press 4.

To reach repair, press 5.

On first sight, this menu might appear quite clear. On closer inspection, and especially by analyzing – based on end-to-end calls – which options callers picked versus what they were really calling about, the confusion between options 2 and 3 becomes obvious. 25% of the callers selecting option 2 really just wanted to add some feature to their existing phone service, which is what option 3 was intended for. Vice versa, 10% of the callers selecting option 3 actually wanted to establish phone service. The lesson to be learned in the context of this guideline is that frequently, distinctions that are dictated by the business (such as, in this example, between establishing phone service and adding features to existing phone service) often are not intuitive to callers, and the menu design must take that into account.

Guideline #5: Minimize acoustic confusability of vocabulary

While obvious to anyone with some knowledge about speech recognition, this guideline is violated surprisingly often. Consider the prompt: *Say "one referral number" or "multiple referral numbers"*. The two phrases that are suggested to callers overlap by two words, and the only difference are the words “one” versus “multiple”. While speech recognizers will pick up this distinction under most circumstances, why not convert the prompt into a simple yes/no query instead, such as: *Do you want to specify multiple referral numbers?*, or – better yet – consider skipping the prompt altogether by making a default assumption, which can be corrected later in the dialog.

Guideline #6: Abide by natural turn-taking protocol

As discussed in section 1.1.2 on the limitations of spoken language, turn-taking in human conversation is determined by certain rules; due to the anthropomorphism that speech interfaces elicit, users generally abide by the same turn-taking protocol when interacting with dialog applications. Therefore, it is generally a good idea to abide by the same turn-taking rules that apply to human conversation. Vice versa, we shouldn't use barge-in capability as an excuse for designing overly long prompts. Most callers will

listen to the complete prompt, adhering to the rules of turn-taking courtesy learned in human conversation. Below is an example for a prompt that leads to turn-taking problems.

The prompt occurs in the context of confirming an account number: *Let me make sure I got that right: <read back account number> - is that correct? Please say 'yes' or 'no'.* What is the problem? The prompt violates natural turn-taking, because the question "... *is that correct?*" encourages the caller to take the turn and speak right after it. A recent analysis of a deployed application employing this and similar prompts showed that more than half of the callers barge with their response into "*Please say 'yes' or 'no'.*" The confirmation question "... *is that correct?*" provides such a strong turn-taking cue to callers that they blurt out the answer right away. At best the natural flow of the dialog is interrupted, and at worst the barge-in may be misrecognized, requiring the caller to engage in an error correction dialog.

Guideline #7: Coach a little at a time

Coaching a little at a time is a well-known error recovery technique in spoken dialog design. For example, it is a good practice to provide examples in the timeout reprompt. For example, following an open-ended prompt, such as: *Tell me, briefly, the reason for your call today* - coaching a little at a time might be realized by continuing with: *You can say something like: 'I need to make a payment', or 'my service is not working'* if the caller does not respond within 3-5 seconds.

Guidelines #8: Offer alternative input modalities

Research has shown (Oviatt, DeAngeli and Kuhn, 1997; Suhm, Meyers and Waibel, 1999) that offering alternative modalities for error correction dramatically increases correction success. Furthermore, alternative input modalities alleviate privacy concerns when sensitive information needs to be exchanged.

Offering both touch-tone and speech for digit input has become a widely accepted good practice. Still, some designers eager to promote speech suggest prompts like: *Please say your account number.* A modified prompt applying Guideline #8 might read: *Please enter or say your account number.* Analyses of end-to-end calls into live systems consistently show that 60% of all callers choose to enter the number using their touch-tone keypad when offered the alternative. Knowing that touch-tone input can be recognized more accurately than spoken input, why miss out on this opportunity to provide a choice *and* avoid the chance of recognition errors?

Guideline #9: Choose persona judiciously

Introduced by research from the late 90s (Reeves and Nass, 1996), the role of personas in voice user interfaces has been a topic of intense argument. Feedback from deployed applications and customer surveys indicate that for most applications, callers prefer simply a professional persona, i.e., a system that's focused on helping them to get the job done – nothing less, but nothing more (Boston Globe op-ed 10/16/2005). The following prompt, suggested for an application taking trouble reports, demonstrates inappropriate invocation of an empathic persona:

IVR: I'm Sam, your automated repair technician.

Caller: My phone's not working.

IVR: Ooouu, sorry to hear that, what kind of problem are you having?

Having the IVR pretend empathy is inappropriate because a repair service should act professionally, and because the persona would lose any credibility in false recognitions of "phone not working". Dealing with emotions, such as empathy, requires nuanced interactions that current speech systems generally cannot handle. Certain applications – for example, ones targeted at entertainment – might benefit from a careful use of emotions, but for most IVR applications we suggest to avoid eliciting emotions explicitly.

Guidelines #10: Follow proven design process

We decided to include this guideline in our short list to remind the reader of the importance of process. Most vendors of speech applications and human factors practitioners are in agreement about the recommended process for designing and deploying successful IVR applications, applying usability analysis methods and iterative design throughout the lifecycle of an application. However, too many projects still do not follow the proven design process, frequently for the same reasons that software development got into a major crisis decades ago, such as time and budget constraints. In this context, we would like to point out the importance of data-driven usability engineering. Usability engineering based on end-to-end calls, as described in the next section, is very effective and can provide huge benefits. While not among the standard repertoire of speech vendors and IVR usability specialists, we know of several practitioners who have learned these methods from previous articles (Suhm and Peterson, 2001), and who have successfully applied them in their own projects.

1.3 Best Practices for IVR Design?

Lists of guidelines such as presented in the previous subsection enable designers to apply discount usability engineering methods, such as heuristic (usability) evaluations, to voice user interfaces. As mentioned earlier, we do not claim that our list of IVR design guidelines constitutes “best practices”. Guidelines become “best practices” by some form of empirical evaluation that shows a consistently positive impact on some suitable metric. For IVR design, suitable metrics include objective usability metrics, such as transaction success, time to complete transactions, and total IVR benefit, discussed later.

Most of the guidelines that we presented above are widely accepted. Some may argue about their relative importance, or put forth additional proposed guidelines, but neither is the point of this discussion. Rather, we have presented a framework that can assist the process of consolidating all knowledge relevant to IVR design, including some that appears to contradict the above guidelines. Establishing a widely accepted list of “best practices” will require a coordinated and sustained effort by both industry and researchers.

Good design requires creativity and an understanding of the intricate dependencies among conflicting design parameters, which no set of guidelines can replace. While the skillful application of guidelines still requires significant experience, we believe that a better understanding of the limitations of speech and the knowledge of validated design guidelines will lead to better designs and more effective use of speech as an interface modality.

2. DATA-DRIVEN IVR USABILITY ENGINEERING BASED ON END-TO-END CALLS

Usability engineering is comprised of methods to identify usability problems, to develop specific design improvements, and to quantitatively compare alternative designs. This section presents a methodology for IVR usability engineering that distinguishes itself from standard data-driven usability methods, such as usability tests and expert evaluations, in the following two features: first, for using end-to-end calls - the ultimate field-data from call center IVRs – as the basis for the analyses, and second, for providing a means to build the business case for IVR usability engineering.

The first subsection cautions against relying on standard IVR reports as the main data source for understanding IVR performance. We then describe

methods for collecting data from thousands of live calls and processing that data efficiently into a database of complete event traces for calls. Subsection 2.3 presents methods to analyze IVR usability based on end-to-end event traces, including caller-path diagrams – a visual representation of the complete IVR interaction of thousands of calls. The following subsection introduces total IVR benefit as a metric that combines objective usability and cost-effectiveness of IVRs in a single measure. By accurately quantifying the benefit of an IVR, the cost-savings potential for usability engineering can be estimated, and alternative IVR designs can be compared objectively. Empowered to assign monetary value to design alternatives, practitioners thus can develop recommendations that maximize return on investment (ROI) for the call center. Examples from our IVR design consulting practice illustrate the methodology throughout this section. Some of this material has been presented previously (Suhm and Peterson 2001).

2.1 The Flaws of Standard IVR Reports

Why does this chapter emphasize the importance of end-to-end calls to IVR usability engineering? It is because the call center industry commonly relies on flawed data, so-called IVR reports, to evaluate IVR performance and to make IVR design decisions. IVR reports typically contain measures such as “IVR utilization”, average time spent in the IVR, and average agent handling time. IVR utilization (or “IVR take-rate”) is commonly defined as the difference between the percentage of callers entering the IVR and the percentage leaving the IVR to talk to a live agent. While often interpreted as the success rate for serving callers in an automated fashion, IVR take-rate is a poor measure of the effectiveness of an IVR, because callers hanging up in the IVR may not have received any useful information. In several large call centers we have seen that the majority of callers hanging up have actually received no useful information and therefore have not been served. For example, based on standard IVR reports, one call center believed that its IVR served more than 30% of the callers in the automated system. A detailed analyses based on end-to-end calls revealed that only 2% of all callers were actually served. Almost 20% hung up without receiving any useful information, and some 8% hung up while on hold for an agent.

One reason for the poor usability of many deployed IVRs is that decision makers in call centers often lack adequate information. Standard IVR performance reports often do not capture information on usability and lack sufficient detail to identify the specific problems, much less how to remedy them. As a solid foundation for IVR usability analyses, the next subsection describes how to capture end-to-end data from calls, which may be

complemented with select data from IVR reports, provided that they are interpreted adequately.

2.2 Capturing End-to-End Data from Calls

End-to-end calls the only complete record of user and system behavior in IVR applications. Therefore, data from end-to-end calls are invaluable for conducting comprehensive usability analyses of telephone speech applications. A call typically begins in a dialog with an automated IVR system, called the *IVR-caller dialog*, which is sometimes followed by a dialog with a live agent, called the *agent-caller dialog*. The sequence of events in the IVR-caller dialog can be captured from IVR logs, or alternatively can be reconstructed from end-to-end recordings of calls. Significant events in the agent-caller dialog can be annotated in end-to-end recordings, if such recordings are available, or alternatively by monitoring live calls. The following subsections describe these methods in more detail. We begin with how to obtain end-to-end recordings of calls.

Calls can be recorded end-to-end either on-site or off-site. Many IVRs have standard recording equipment integrated with the hardware platform. In all but small call centers, however, a call is typically handled in more than one piece of equipment, making on-site end-to-end recording difficult. In such situations, off-site recording may be the only way to record calls end-to-end.

Recordings of complete calls represent a large amount of data that is difficult to analyze in its raw form. To make the analysis of call data efficient, the recordings can be transformed into a trace of significant events for each call. The IVR-caller dialog is a sequence of system prompts followed by caller input, either touch-tone or speech. The preferred method for capturing the IVR event sequence is an event log that is generated by the IVR. However, to obtain such an event log, the IVR code has to be modified to write to an event log at appropriate states in the call. Generating such code is error-prone and labor intensive, especially because IVR systems are being changed all the time. For cases when IVR logs do not contain a complete event trace, we have developed a method that infers the complete IVR event sequence from the call recording alone.

Three main tools are employed to infer the event sequence for the IVR-caller dialog from a recording: a prompt detector, a DTMF detector, and a prompt inference tool. First, a commercially available DTMF detector is used to detect touch-tones. Next, a prompt detector recognizes important known prompts in all recordings. Finally, whenever the IVR is so complex as to make detection of all prompts impractical, a prompt inference tool

infers the complete prompt sequence based on complete knowledge of DTMF input and partial knowledge of prompts. An additional, crucial step is to determine how the call left the IVR: whether the call ended in the IVR with a hang up or was transferred to an agent. A transfer prompt, such as "Please wait for the next available representative," indicates that the call was transferred to an agent. The absence of a transfer prompt in the call indicates that the caller hung up. This method fails when the caller hangs up during the hold time, before reaching an agent. However, such cases can be corrected during the annotation of the agent-caller dialog, which we describe next.

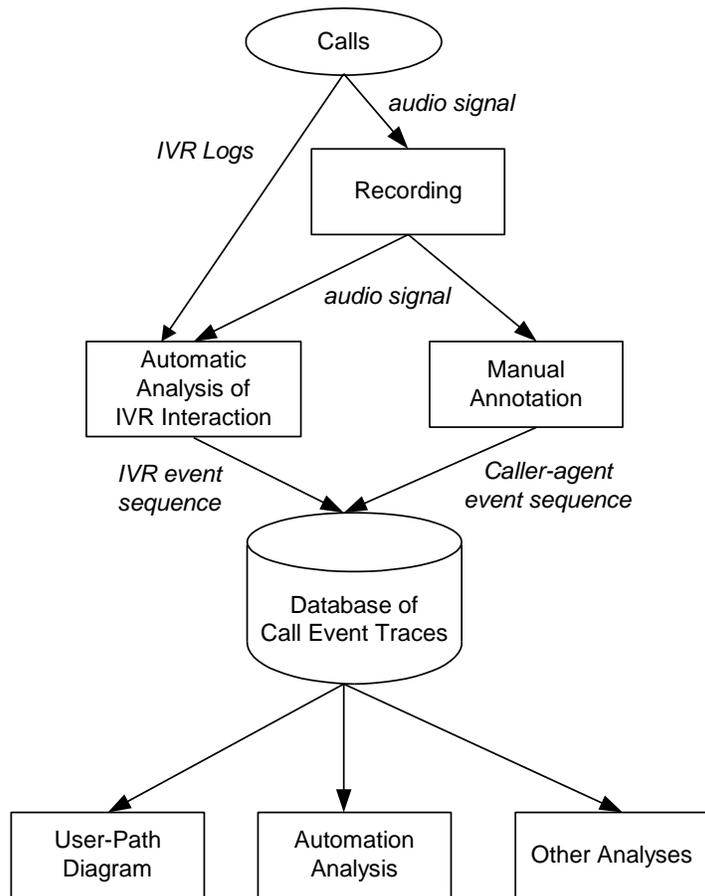


Figure 1-2. Process for capturing end-to-end data from calls and transforming it into a database for data-driven IVR usability engineering

Significant events in the agent-caller dialog are obtained from manual annotation using human transcribers. The annotator either monitors live calls, taking notes for later coding, or listens to the agent-caller portion of end-to-end call recordings. Significant events include start of the agent-caller dialog, the reason for the call and other topics discussed (e.g., question about a bill, inquiry into flight schedules), exchanges of information between caller and agent (e.g., account numbers, dollar amounts), and completion of transactions (e.g., making a payment arrangement or flight reservation). In addition, call annotation may characterize the call as a whole according to certain attributes, such as whether the call was resolved and agent courtesy.

Complete event traces of both the IVR-caller and agent-caller dialogs can be organized using a database, along with the end-to-end recording (if available). The ability to switch between the call recording and its representation as an event trace is very useful throughout the analysis process. The methods for obtaining the event trace apply to both touch-tone and speech-enabled IVRs. Figure 1-2 illustrates the complete process for capturing end-to-end call data and transforming it into an annotated call database.

2.3 Evaluating IVR Usability based on End-to-End Calls

The database of annotated calls is the basis for our methods to evaluate IVR usability and cost-effectiveness. This subsection introduces call-reason distributions and caller-path diagrams as effective tools to diagnose problems and to predict the impact of design changes. Examples further illustrate how to employ a caller-path diagram in conjunction with the distribution of call reasons to identify IVR usability problems. We begin by describing how to obtain and leverage knowledge of the call-reason distribution.

2.3.1 Call-reason Distribution

Obtaining the distribution of call reasons should be one of the first steps in any IVR usability analysis. Knowing what kinds of problems customers are calling about, and the absolute frequency of the various requests – relative to all calls entering an IVR – is the equivalent of understanding the user needs in more traditional usability work, such as developing a new software tool. While experienced practitioners can identify potential IVR usability problems simply by inspecting the dialog script (typically referred to as “call flow”), or by conducting usability tests, neither of these traditional methods provide the problem frequencies in live calls, thus making it

difficult to determine problem severity. Furthermore, the call-reason distribution can guide IVR design effectively: for developing the high-level design, for prioritizing which caller requests are important and should receive the most attention, and for ordering menus by request frequency. The distribution of call reasons is therefore a crucial step in IVR usability engineering. But estimating it correctly is not trivial.

Call centers sometimes attempt to infer call-reason distributions based on peg counts of IVR dialog states, i.e., based on how often callers access certain states in the IVR. However, this method is flawed because callers can bypass the IVR completely by transferring to a live agent, and callers who do cooperate frequently make a wrong selection in the IVR and get routed to the wrong specialist agent or miss an opportunity to resolve their inquiry in self-service. In our experience, only 35% to 75% of all callers get to the right area within a complex menu tree using directed (touch-tone or speech) menus.

Instead of relying on inaccurate IVR reports, we estimate the call-reason distribution by combining the distribution of calls that self-serve in the IVR with annotations of the call reason in randomly selected agent-caller dialogs. Table 1-7 shows such a call-reason distribution from one of our case studies.

Table 1-7. Example for a call-reason distribution

Call reason	% (All) Calls
Sales	24%
Establish new account	17%
Payment information and arrangements	11%
Account balance	17%
Billing questions	10%
Repair	7%
Other	14%

Later subsections will illustrate the uses of the call-reason distribution in more detail, including identifying usability problems, estimating upper bounds on IVR automation, and guiding IVR (re)design.

2.3.2 Diagnosing IVR usability using Caller-Path Diagrams

Caller-path diagrams are a diagnostic tool for identifying IVR usability problems and an analytic tool for estimating the impact of design changes. Caller-path diagrams visualize user behavior in the IVR by representing the sequence of events in the IVR, similar to state-transition diagrams. State-transition diagrams have been applied to many engineering problems, including user interface design (Parnas, 1969). Applied to visualizing user behavior in IVRs, state-transition diagrams visualize the paths and level of call resolution of many callers through the IVR, hence the name caller-path

diagram. To manage the complexity of such caller-path trees, individual IVR states are clustered into subdialogs, such as ID entry or menu selection. Such subdialogs may encompass multiple IVR-caller interactions from the captured IVR event sequence.

The nodes of the tree correspond to IVR states, arcs correspond to state transitions, and leaves correspond to end conditions of calls (call resolution). Each node and leaf is marked with the percentage of all calls that reached the node or leaf. In addition, arcs may be marked with the user input that causes the corresponding state transition, such as pressing a certain touch-tone in response to a prompt. We found it helpful to distinguish at least three end conditions. “Self-serve” refers to calls that are resolved in the IVR, i.e., the customer completes the call in the IVR, without talking to a live agent. “To agent” are calls that transfer to an agent. “Abandon” refers to caller hang ups without obtaining any useful information, either in the IVR or on hold before reaching a live agent. If the call center operates with distinct categories of agents, the “to agent” category is typically subdivided into various subcategories, each representing a distinct routing destination from an operational point of view.

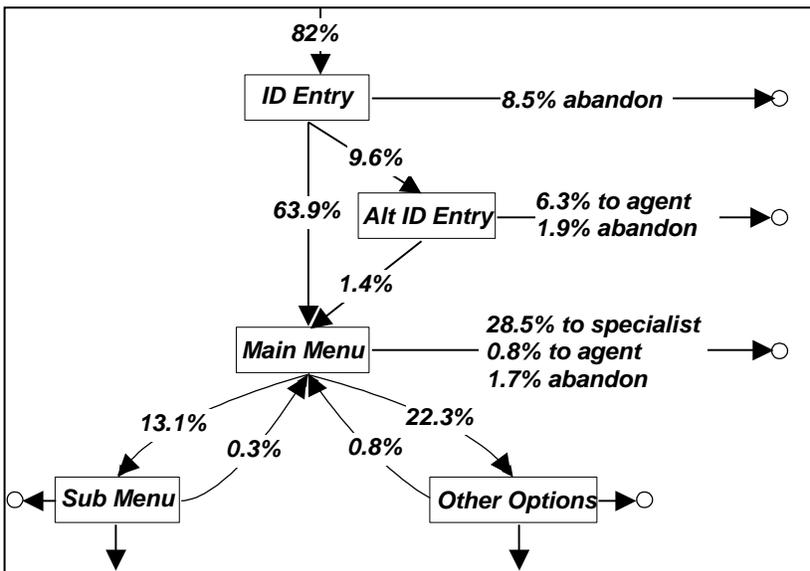


Figure 1-3. Caller-path diagram example

Figure 1-3 shows an excerpt from a caller-path diagram. Rectangular boxes represent IVR states, arrows represent call traffic, and circles indicate places where calls leave the IVR. In this example, 82% of all callers make it

past the opening menu to a state that prompts the callers to key-in their account number, called “ID Entry” in this figure. Of all callers, 8.5% abandon the call while attempting to provide their account number, shown as an arrow to the right. Note that the caller-path diagram shown in the figure represents initial, error, and timeout prompts as one rectangle. Hence, most of the 8.5% of all calls that abandon during “ID Entry” do so after encountering one or more error prompts. On the other hand, 63.9% of all callers enter their account number successfully and reach the main menu. At the main menu, 28.5% of callers select an option that routes them to a specialist agent, while 0.8% route themselves to a general (floor) agent, and 1.7% abandon the call.

Usability problems can be identified by inspecting caller-path diagrams. Usability problems are likely in those areas of the call flow that receive little or no caller traffic, or that have high rates of abandoned calls or transfers to an agent. In Figure 1-3, for example, the dialog state named “ALT ID Entry” receives 9.6% of all calls, but 1.9% are abandoned, 6.3% are transferred to a floor agent, and the account number is correctly entered in only 1.4%. Obviously, the “Alt ID Entry” dialog is ineffective.

Developing alternative designs that solve the usability problem requires an understanding of why callers fail in a particular part of the IVR dialog. Typical root causes for IVR usability problems have been discussed earlier in section 1.2.3, including:

1. *Task complexity*: for example, requiring callers to provide information that they don't have readily available, or attempting to offer self-service for rather complex problems. In this context, the sequential character of the voice channel, the limited short-term memory capacity of users, and the design principle of keeping it simple come into play.
2. *Poor prompt design*, including use of ambiguous or confusing terminology, excessive verbiage, long menus, short timeouts, and misleading turn-taking cues.
3. *Caller non-compliance* is elicited by poor IVR design, such as overstraining caller patience by requiring them to go through too many steps. As a rule of thumb, callers want to feel significant progress towards the goal of getting their job done within three steps.
4. *Speech recognition* problems, provided the IVR is speech-enabled.
5. *Backend* issues, such as excessive latencies to retrieve information from backend databases.
6. *Business rules* refer to decisions within a call flow that are determined by the business. Business rules may determine that calls get routed to an agent, for example, accounts that are past due. Such rules may force too

many callers to leave the IVR prematurely, if they are not carefully designed to balance business goals with self-service.

Deciding which of these causes applies to a specific IVR usability problem requires studying the call flow, experience, and may require additional analyses. In the example above, the “Alt ID Entry” dialog state prompted the caller for an obscure 12-digit account number, if the caller failed to provide their primary one. Most callers did not know their alternative account number. Hence, possible design solutions include:

- Prompting the caller for something that’s easier to remember as an alternative way to verify their identity, such as their social security number or telephone number.
- Including an instruction about where to find the “alternative ID” in the reprompt.

To decide between alternative design solutions, we need to consider feasibility and ratio of expected benefit versus the cost to implement the solution. For example, switching to a different kind of “alternative ID” might require significant modifications to both IVR and backend code. Our method for cost-justifying IVR usability engineering, presented later in section 2.4, enables practitioners to make design decisions objectively, such as in the example above, and to maximize ROI at the same time.

The following subsection presents how to identify IVR usability problems by jointly inspecting caller-path diagrams and call-reason distributions.

2.3.3 IVR usability analysis using call-reason distribution and caller-path diagram

The caller-path diagram becomes even more useful when analyzed in conjunction with the call-reason distribution. Comparing the true frequency of call reasons with what percentage of all calls reaches the corresponding IVR module reveals navigational problems. Navigation (also referred to as “call routing”) is an extremely important function of an IVR, and therefore an important subject of any IVR usability analyses.

The call center in this example serves many functions, including sales, billing questions, and repair. The caller-path diagram in Figure 1-4 shows the first two menu layers in detail, but abbreviates the provision of automated information as “Automated Billing Information” and “Automated Fulfillment”. Visual inspection of this caller-path diagram reveals several IVR usability problems, which are explained in more detail below. The numbers of the shaded ovals in the figure correspond to the paragraph number.

First, about 30% of all calls are either abandoned or are transferred “cold” to an agent right at the main menu, which is the first menu in this call flow. This traffic represents the callers who attempt to bail out of the IVR at the first opportunity. While we might empathize with such callers, they are likely to be transferred to the wrong agent who then transfer the caller to another agent. Transfers mean a second period of waiting on hold for the correct agent. Therefore, bailing out of the IVR before or at the main menu is bad from both the call center’s and the caller’s point of view.

Second, while 18% of callers choose “other billing questions,” only 3% actually get to the billing submenu using this alternative path, and 15% bail out to an agent – after spending more than 1 minute in the IVR without having received or provided any useful information.

Third, the billing IVR achieves very little automation, because only 5% of all callers find “Automated Billing Information”. Only 3% of callers actually obtain automated information in the billing IVR, although 10% of callers find the billing menu. By contrast, a standard IVR report for this call center would indicate a 19% IVR take-rate, which really just means that 19% of all callers hung up in the IVR. The IVR report would *not* reveal that less than one in six such callers (3% overall) actually obtained useful information!

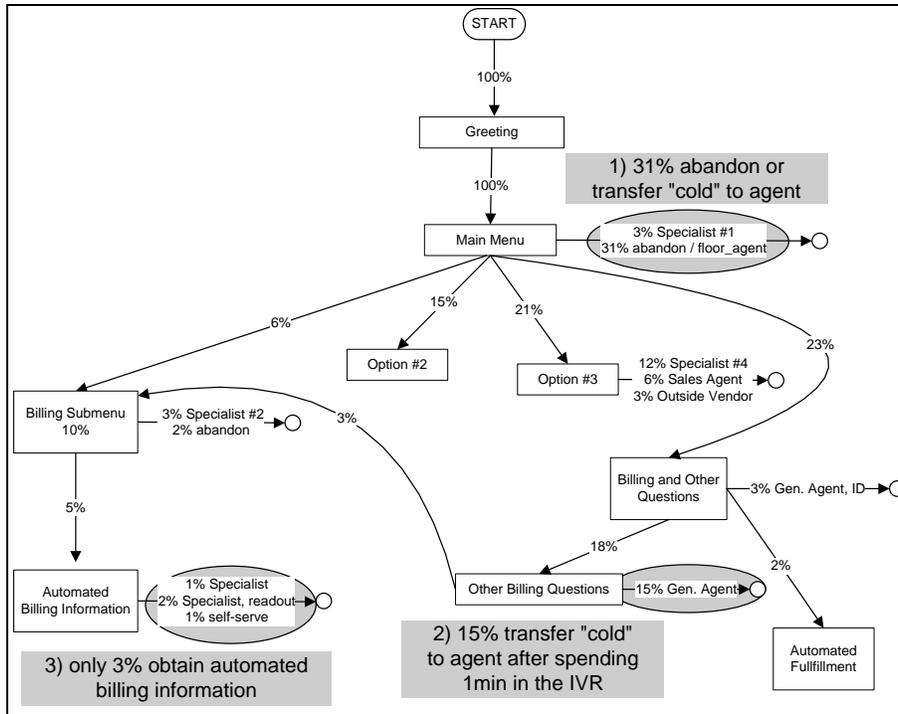


Figure 1-4. Identifying IVR usability problems by inspecting a caller-path diagram

This example also illustrates why inferring the call-reason distribution from IVR reports is flawed. IVR peg counts would indicate that 10% of callers reach the billing IVR, but this does not mean that 10% of all incoming calls are related to billing, because many callers may not find the billing IVR. In conjunction with knowledge of the correct call-reason distribution, derived from annotations of agent-caller dialogs and presented in Table 1-7 above, the following additional issues become obvious:

- 27% call about billing related questions, which include inquiries for the account balance. But only 10% of the callers find the billing IVR.
- 24% should be handled by a sales representative, but only 6% of the callers are actually transferred to a sales representative.

Developing design solutions for each of these usability problems is beyond the scope of this chapter. But more generally, based on our detailed analyses of IVRs across several industries, we have identified the following common IVR usability problems:

1. Excessive complexity - many IVR functions are underused because customers get confused early in the call.
2. Caller identification difficulties – dialogs that attempt to identify the caller frequently represent hurdles to the delivery automated customer service. Even with effective use of Automatic Number Identification (ANI), the success rate may be low because customers call from phones other than the one registered with their account.
3. Confusing menus - menu wording is often based on call center or technical jargon and may not reflect how the customers think about the problem at hand. The customers then make the wrong selections, and miss the opportunity to self-serve in the IVR.

Other analyses allow practitioners to identify confusing menus and quantify routing benefit, but they are beyond the scope of this chapter. This subsection demonstrated how to leverage end-to-end calls to diagnose IVR usability problems. Knowing the specific problem frequently suggests a remedy. In the above example, the self-service functionality is underused because we are losing 1/3 of all callers already at the main menu. Inspection of the main menu wording may make a solution obvious; for example, the agent option is advertised in the initial version of the menu, or too many options are presented in too many words, thus confusing callers. The following subsection moves on to leveraging these analyses in evaluating IVR cost-effectiveness and building the business case for IVR usability engineering.

2.4 Evaluating IVR Cost-effectiveness

Evaluating call center IVRs quantitatively is difficult. Evaluation criteria from the caller's point of view (usability) and from the call center's point of view (cost-effectiveness) appear difficult to reconcile. Standard evaluation methods are either inadequate or address usability and cost-effectiveness in isolation. As mentioned earlier, standard IVR reports can be misleading and do not capture reliable usability metrics. Methods to evaluate subjective usability exist, but they do not quantify the cost for the call center. Common laboratory usability evaluations, using task-based measures in controlled experiments on a few tasks, are impractical for complex call center IVRs, which can offer many different functions (tasks). We therefore introduce total IVR benefit as a single measure that combines IVR usability and cost-effectiveness. Further subsections describe how to measure total IVR benefit, how to employ this metric to estimate the improvement potential, and how to justify the cost of IVR usability engineering and other IVR enhancements.

2.4.1 Defining Total IVR Benefit

In defining a metric that captures both IVR usability and cost-effectiveness several issues must be considered. On the one hand, callers want to accomplish their goals quickly and easily over the phone. Therefore, objective usability can be quantified by the standard measures of task completion rates and times. On the other hand, agent time dominates the cost in most call centers. The ratio between cost of agents and all other costs, such as telecommunications time, IVR hardware and software, and facilities charges, is at least 4:1 (Balentine, 2006). Therefore, agent time saved is a good measure of the cost-effectiveness of an IVR. We define the *total IVR benefit* as the agent time that is saved by the IVR per call, averaged across all calls, compared to handling the complete call by live agents.

Table 1-8. Taxonomy of tasks that can be automated in an IVR, along with conservative benefit assumptions

Automatable task	Benefit (agent seconds)
Capture caller ID or account number	15
Correct routing (to specialized agents or self-service)	40
Obtain useful information	40
Complete transaction	60

An IVR “saves” agent time whenever it performs tasks successfully that otherwise would have to be performed by an agent. Tasks that typically can be performed within an IVR include identifying the caller, providing information to the caller, performing transactions, and routing the caller to specialized agents. In some cases, completing these tasks successfully may resolve the call so that the caller hangs up without any assistance from an agent. Such calls are commonly referred to as self-serve or full automation. It is important to note, however, that even if a call is not fully automated, the IVR can still provide significant savings through partial automation. Table 1-8 shows typical agent time savings for various “automatable” tasks. These savings can be derived from benchmark assumptions or measured in annotated agent-caller dialogs.

While the emphasis in this context is on cost, we note that IVR automation understood in this manner corresponds to task completion. In this sense, what we refer to as IVR automation is a more differentiated version of task completion, which is a standard measure of objective usability. Total IVR benefit can thus be interpreted as a metric that combines cost-effectiveness with (objective) usability.

2.4.2 Measuring Total IVR Benefit

Total IVR benefit could be measured directly by timing the length of agent-caller dialogs. But as agent time has a large amount of variation, the length of thousands of agent-caller dialogs would have to be measured, which requires manual annotation of calls and thus is costly. Furthermore, it is impossible to obtain unbiased data from commercial call centers, because many factors may have a significant impact on caller behavior and agent handling time. We therefore have developed a method to estimate total IVR benefit based on call event sequence data, which we call *IVR automation analysis*.

Table 1-8 defines tasks that can be automated in the IVR. But how does one determine whether IVR automation was achieved during a call, i.e., the caller successfully completed one of these tasks? Typically, the completion of a task can be associated with reaching a certain state in the IVR-caller dialog. Thus, the set of completed tasks can be inferred directly from the event sequence data for a call, using a simple lookup table that indicates which IVR states correspond to the completion of certain tasks.

We make one important exception to the assumption that IVR states indicate successful task completion. Specifically, we do not assume that routing decisions made in the IVR are necessarily correct. Rather, we look at subsequent agent-caller interactions to determine, based on the annotated reason for a call, whether the call was correctly routed or misrouted to an agent. Calls that misroute to specialists usually need to be transferred somewhere else and, therefore, incur a cost equal to the time it takes the specialist to reroute the call, which can be thought of as a negative routing benefit.

Given the definition of tasks that can be completed within an IVR, we characterize each call according to distinct combinations of automated tasks, which we refer to as *call profiles*. Given a set of calls with their sequence of IVR-caller and agent-caller events, we annotate each call with its set of completed tasks, and use the pattern of completed tasks to accumulate counts for each call profile. The call traffic handled by an IVR is thus partitioned into a set of call profiles, each representing a distinct pattern of automation.

Leaning on the well-known concept of task completion rates, *IVR automation rates* are defined as the percentage of automation achieved over all calls, for each automatable task. This percentage can be calculated simply by adding the percentages of all call profiles that include the specific automatable task. Some call profiles correspond to the completion of more than one task. In that case, their percentage is part of the automation rate for all those tasks.

Table 2-9 shows an example IVR automation analysis, which distinguishes two agent types, “specialist” and “floor.” The left column lists the call profiles. The next two columns (labeled “Traffic”) show the breakdown of the total data set - consisting of 5530 calls - into the various profiles. For example, 5.6% of the calls were fully automated, and 7.9% of the calls were abandoned without the caller getting anything done. Further to the right in the table, the three “Automation” columns show the automation categories for each profile. This analysis is based on three automation categories: capture of the caller's account number, routing, and delivery of (useful) information. In each “Automation Category” column we enter the associated agent time savings from Table 1-8. For example, the profile “Transfer to floor with information” achieved capture of the account number and automated delivery of information. The bottom row in Table 1-9 for the three “Automation” columns shows the total automation rates for each category: 29% capture of account number, 29% routing, and 9% information delivery.

Table 1-9. IVR Automation Analysis, with two agent categories (“specialist,” “floor”)

Call Profile	Call Traffic		IVR Automation Categories			Benefit [agent seconds]	
	Calls	%	Acc ount	Rou- ting	Useful info	Per call	Avg
Fully-automated	307	5.6%	15	40	40	95	5.3
To specialist w/ information	99	1.8%	15	40	40	95	1.7
To floor with information	101	1.8%	15		40	55	1.0
To specialist with account	641	11.6%	15	40		55	6.4
To specialist “cold”	545	9.9%		40		40	3.9
To floor with account	471	8.5%	15			15	1.3
To floor “cold” (no account)	2927	52.9%				0	0
Abandoned	439	7.9%				0	0
Total	5530	100%	29%	29%	9%		19.6

For each call profile, the average saved agent time over all calls handled by the IVR (shown as the last column in Table 1-9) is the product of the total agent time saved for one call with the corresponding percentage of traffic. For example, the call profile “transfers to specialist with account” corresponds to 55 seconds savings of agent time per call, because the call was transferred to the right place (routing automation), and the caller was identified (account number automation). Since 11.6% of all calls fit this profile, the contribution of this profile to the total savings of agent time is estimated as 11.6% times 55 seconds, which equals 6.4 agent seconds. The *total IVR benefit*, then, is the sum of the net IVR benefits for all call profiles.

For the example in Table 1-9, our analysis estimates a total IVR benefit of 19.6 agent seconds saved, shown in the bottom right corner cell. In other words, we estimate that this IVR shortens, on the average, the agent handling time for every call by 19.6 seconds.

Besides quantifying IVR effectiveness, the automation analysis can also be used to identify usability problems, because low automation rates in one or more automation categories, relative to their upper bounds, point to usability problems. A method for estimating upper bounds on automation is described in the next subsection.

How does such an IVR automation analysis compare to the standard way of quantifying IVR automation in the call center and speech industries? The industry typically distinguishes two categories (in our terminology, “profiles) of calls: calls that are fully automated and never reach an agent, versus calls that reach an agent. For fully automated calls, the IVR is credited with saving the average agent handling time of a call, typically multiple minutes. For all other calls, on the other hand, no benefit is credited to the IVR-caller interaction. This approach is motivated by the viewpoint that the main objective of an IVR is to keep calls from reaching the call center agents. This viewpoint, however, is one-sided and unnecessarily limits what one can achieve with an IVR. Suggesting that the goals of caller and call center are mutually exclusive is one-sided. Our approach, on the other hand, sees the IVR as a means to offload routine tasks from agents, enabling call centers to shave a few seconds of agent handling time from most calls by getting at least the most basic tasks done in the IVR. The callers get their job done without waiting on hold for an agent - if they have a routine request – or because the agent can focus on providing better service, leveraging the information that the caller has provided in the IVR - such as capturing the caller’s account number or even the reason for the call. We believe that a more differentiated view of IVR automation is key to analyzing IVR usability comprehensively and maximizing benefit to both call center and caller.

The IVR automation analysis adapts the standard usability measures of task completion time and rates to the problem of evaluating usability and effectiveness of call center IVRs. But how does this help us evaluate the cost-effectiveness of an IVR and build the business case for making changes to it? First, total IVR benefit is easily translated to cost savings. For example, 1 agent second across all calls in a year in a call center handling 10M calls per year, at an hourly agent cost of \$0.70, corresponds to $10M * \$0.70 / 60 = \$117K$. Knowing that, the above IVR automation analysis means that the current IVR is saving $19.6 * \$117K = 3.27M$ per year in agent time. Second, using the call-reason distribution, one can estimate

upper bounds for each automation category. By comparing the upper bounds with the existing automation (as quantified in the automation analysis), we can obtain fairly accurate estimates of the improvement potential, i.e., how much additional agent time could be saved by redesigning the IVR. The next subsection illustrates this process in a continuation of our case study above.

2.4.3 Estimating Improvement Potential

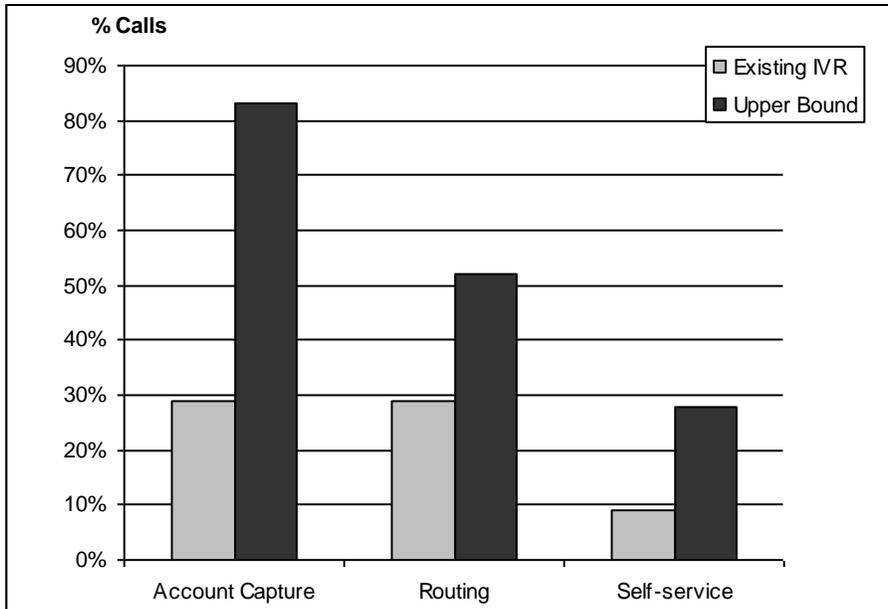


Figure 1-5. Estimating the potential to improve (partial) IVR automation

Knowing how much an IVR could be improved is the basis for building the business case for IVR usability engineering. Based on the analyses of end-to-end call data, described above, the improvement potential can be estimated based on upper bounds on automation for each automation category (using the call-reason distribution), as the difference between upper bounds and existing automation (as determined in the IVR automation analysis). We obtain realistic improvement estimates by further discounting the improvement potential, knowing that even the best design will not always be successful. By distinguishing several categories of automation, and by crediting the IVR for partially automated calls that saved some (but

not all) of the agent handling time, this method evaluates IVR cost-effectiveness more accurately than the common method of building IVR business cases based on the distinction of fully self-served versus agent-handled calls.

Which are the upper bounds for capture of account number, self-service information, and routing for the IVR whose call-reason distribution was shown in Table 1-7? First, the upper bound on capturing the account number is $100\% - 17\% = 83\%$, because the 17% of callers that want to establish a new account cannot be expected to provide an account number. Second, regarding self-service, it is offered by the current IVR for obtaining the account balance, paying the bill, and making payment arrangements. Without adding further functionality, the upper bound on self-service is therefore $11\% + 17\% = 28\%$. To estimate the upper bound on routing, let's assume that new accounts and repair requests are handled by specialist agents, while all other requests are handled by the floor (agents). Under these assumptions, the upper bound on (beneficial) routing is $17\% + 7\% + 28\%$. Note that we also credit the IVR for routing callers to self-service functions. Using the levels of existing automation from the automation analysis in Table 1-9, Figure 1-5 illustrates the improvement potential for this IVR. As can be seen, there is significant potential for improvement in all automation categories; the account number could be increased by 50% absolute, self-service could be increased 20%, and beneficial routing by 24%.

As the next step towards building the business case, improvement potentials need to be translated into cost-savings opportunities. Automation rates are easily translated to average savings in agent time by multiplying them with the appropriate benefits assumption (cf. Table 1-8). In the above example, at best the IVR could save an additional 25 agent seconds per call. Realistically, applying a rule of thumb, IVR usability engineering might achieve no more than half of the potential; in the example, half of 25 agent seconds would correspond to almost \$1.5M over the 10M calls that are handled per year.

2.4.4 Building the Business Case for IVR Redesign

Due to the cost pressures in the call center environment, the redesign of IVR applications must be justified with a business case. The IVR automation analysis and benefit calculation presented above can provide the necessary business justification for IVR redesign, because the cost savings of the redesigned IVR can be estimated. Based on an automation analysis of the existing IVR and knowledge of usability problems, we can derive bounds for

improvements in the various automation categories. From these bounds, we can project upper limits on annual cost savings, which are then used to justify reengineering effort.

To illustrate how to build the business case for a specific design recommendation, let's go back to the example discussed in section 2.3.2 and Figure 1-3. Our analysis of the caller-path diagram revealed that out of 10% of all callers who were prompted for the 12-digit alternative account number, only 1.4% could provide it. One design recommendation was to prompt for the social security number as an alternative ID. To estimate the impact of this change, we need to estimate how many more callers might successfully provide a social security number, instead of an obscure 12-digit number. A conservative assumption would be to expect a 2/3 success rate for entering the social security number. Using our benefits calculation method, and just considering the benefit of increasing account screen pops to agents (corresponding to 15 seconds of savings each), the potential cost savings of this change is estimated at: $2/3 * (10\% - 1.4\%) * 15 \text{ seconds} * \$117 \text{ K/second} = \$100\text{K}$. Hence, we expect to save the equivalent of \$100K per year in agent time. If the cost of implementing the change was less than \$100K, we would have a payback of less than one year for this design improvement.

Our reengineering methodology, which is based on evaluating designs with real callers, eventually produces very tight benefit projections. The example below illustrates how the business case was built for moving from touch-tone menus to speech-enabled routing with an open-ended prompt. In this case study, which is taken from a deployment at a large telecommunication services provider in the US, the provider did not want to commit to the large capital expenses associated with a large-scale deployment of speech. The project proceeded in several phases. We first assessed the existing, touch-tone IVR system to obtain a baseline. Based on an analysis of the improvement potential, the business case for migrating from touch-tone menus to call routing using an open-ended prompt was positive. Employing an open-ended prompt to determine the reason for the call is also referred to as "natural language call routing" or "call steering". Chapter 3 will provide further detail on such systems. Examples include AT&T's "How may I help you?" system (Gorin, Parker, Sachs and Wilpon, 1996), deployments of BBN Call Director at Verizon, and various deployments of Nuance "Say Anything" technology.

The analysis of end-to-end calls also uncovered some easy-to-implement improvements of the touch-tone menu system. To prove that the natural language call routing system would outperform even an improved touch-tone menu system, we collected and analyzed end-to-end call data from prototypes of the modified touch-tone IVR and the natural language call

router. The automation analyses showed that the natural language call router provided benefit even beyond the improved touch-tone design. Overall, the number of successful routes in the IVR increased by a factor of three over the original touch-tone system. After accounting for the part of the gain that could be attributed to reordering the sequence of menus versus capture of the customer ID, the speech-enabled call router increased IVR benefit by an additional nine agent seconds, thus effectively doubling the total IVR benefit compared to the baseline. Figure 1-6 below shows the comparison of these three designs based on total IVR benefit, as well as the breakdown of automation rates in the various categories.

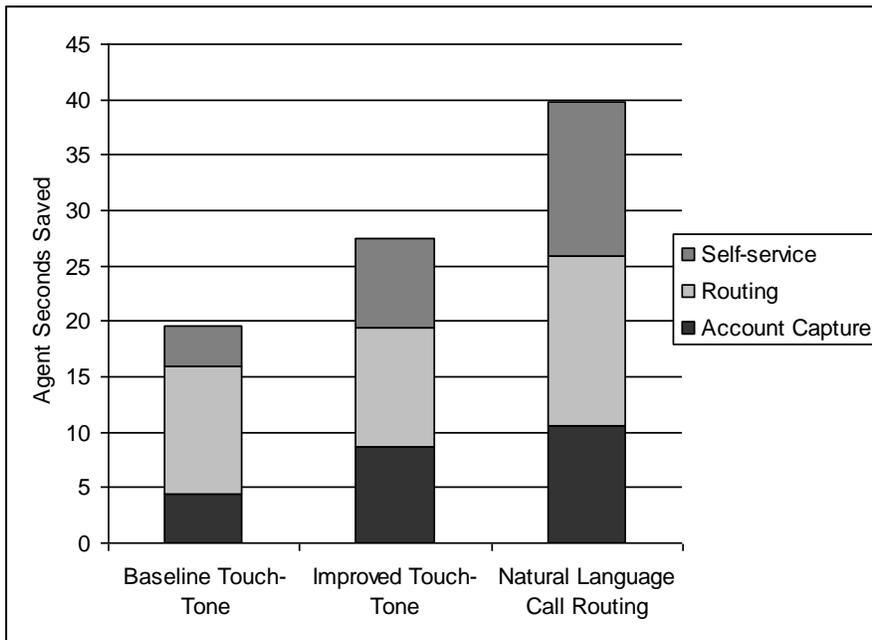


Figure 1-6. Iterative improvement from a baseline touch-tone IVR via an improved touch-tone system to a speech-enabled IVR employing natural language call routing

3. SUMMARY AND CONCLUSIONS

The widespread adoption of speech-enabled IVRs by the call center industry over the past decade has resulted in a significant body of know-how in IVR usability evaluation and engineering methods. Despite significant

advances, usability of many deployed IVRs is still poor. Decision-makers have not fully embraced the value of usability and are forced to cut costs without being able to assess the significant impact of usability on customer satisfaction and the financial bottom line. To help remedy this situation, this chapter has presented:

1. an overview of IVR design principles and usability challenges;
2. a short list of IVR design guidelines; and
3. a methodology for evaluating both cost-effectiveness and usability of IVRs based on end-to-end calls.

Speaking in a painter's metaphor, this chapter provides designers with a broad brush for decent initial designs, and effective tools for the detailing work to optimize a design iteratively. The design principles and short list of design guidelines presented in the first section can help practitioners to develop decent initial designs, which then can be further improved using data-driven usability engineering based on end-to-end calls. For improving an existing IVR, analyses of end-to-end calls represents a powerful tool. Ideally, data from end-to-end calls should influence all phases of an IVR project, including requirements analysis, development of the business case, high-level design, detailed design, and functional and usability testing. An analysis of end-to-end calls therefore should be included in any major IVR (re)design project, and it can be complemented by other usability methods.

Calls must be recorded in their entirety to capture the complete user experience. Hundreds, or even thousands, of calls are necessary to obtain statistical significance in the analyses. We described several methods to measure and analyze IVR usability, including call-reason distributions, caller-path diagrams, IVR automation analyses, and comparative IVR analyses. These methods enable practitioners to solve the tough problems in IVR redesign. Call-reason distributions and (provided a legacy IVR application exists) caller-path diagrams provide valuable guidance in the requirements analysis and high-level design phases. By identifying IVR usability problems and comparing alternative designs, very specific recommendations on how to improve an existing design can be developed. By quantifying the improvement opportunity and measuring potential cost savings, a solid business case can be built, and the cost of call flow re-engineering can be justified, helping call center managers to prioritize their limited resources.

Our methodology of quantifying IVR automation and benefit is superior to standard IVR reports. In particular, we have shown that the standard measure of "IVR take-rate" can mislead call center managers to believe that their IVR is quite effective, while IVR usability actually may be very poor. We have presented total IVR benefit as an accurate, quantifiable measure

that combines objective usability and cost-effectiveness. We recommend adoption of total IVR benefit as the standard benchmark for IVR performance. Our methodology for data-driven IVR usability engineering based on end-to-end calls is more powerful than standard usability tests, because it enables practitioners to optimize both IVR usability and ROI.

Our methodology currently does not formally evaluate user satisfaction or any other subjective usability measure. While the impact of user satisfaction on customer attrition can be large, most managers of call centers focus on operational savings and ignore user satisfaction, because it is difficult to quantify. We believe that standard methods developed in the human factors community are sufficient to evaluate user satisfaction with call center IVRs. Some of these methods, such as expert walk-throughs and surveys in the evaluation phase, and usability tests or focus groups in the redesign phase, are complementary to our data-driven assessment. With each method having its own strengths and weaknesses, a combination of complementary methods can be powerful, bringing in the user perspective in various ways throughout the entire evaluation and design process.

ACKNOWLEDGEMENTS

The assessment methodology presented in this chapter was developed over years of research and consulting for several large call centers, with contributions from all members of the Call Director team at BBN Technologies, in particular, Pat Peterson. The author also gratefully acknowledges the contribution of Suresh Bhavnani in developing the structure of the design framework. Sincere thanks also to the editors for their comments and careful proofreading of this chapter.

REFERENCES

- Balentine, B., and Morgan, D. P. (1999). *How to build a speech recognition application*. San Ramon, CA: Enterprise Integration Group.
- Balentine, B. (2006). *It's Better to Be a Good Machine*. San Ramon, CA: Enterprise Integration Group.
- Bennacef, S., Devillers, L., Rosset, S., and Lamel, L. (1996). Dialog in the RAILTEL telephone-based system. In *International Conference on Spoken Language Systems (ICSLP)*, Philadelphia, PA:IEEE. Vol. I, pp. 550-553.
- Cohen, M. H., Giangola, J. P., and Balogh, J. (2004). *Voice User Interface Design*. Addison-Wesley.

- Delogu, C., Di Carlo, A., Rotundi, P., and Satori, D. (1998). A comparison between DTMF and ASR IVR services through objective and subjective evaluation. In *Interactive Voice Technology for Telecommunications Applications (IVTTA)*, Italy: IEEE. pp. 145-150.
- Edwards, K., Quinn, K., Dalziel, P. B., and Jack, M. A. (1997). Evaluating commercial speech recognition and DTMF technology for automated telephone banking services. In *IEEE Colloquium on Advances in Interactive Voice Technologies for Telecommunication Services*, pp. 1-6.
- Edwards, K., Quinn, K. et al. (1997). *Evaluating Commercial Speech Recognition and DTMF Technology for Automated Telephone Banking Services*. IEEE Colloquium on Advances in Interactive Voice Technologies for Telecommunication Services.
- Gorin, A., Parker, B., Sachs, R., and Wilpon, J. (1996). How may I help you? In *Interactive Voice Technology for Telecommunications Applications (IVTTA)*. IEEE. pp. 57-60.
- Halstead-Nussloch, R. (1989). The design of phone-based interfaces for consumers. In *International Conference for Human Factors in Computing Systems (CHI)*, ACM. Vol. I, pp. 347-352.
- Holtzblatt, K. and Beyer, H. (1998). *Contextual Design*, Morgan Kaufmann.
- Karat, C.-M., Halverson, C., Horn, D., and Karat, John. (1999). *Patterns of Entry and Correction in Large Vocabulary Continuous Speech Recognition Systems*. International Conference for Computer-Human Interaction (CHI), Pittsburgh (PA), ACM, Vol. 1, pp. 568-76.
- Karat, J., D. Horn, D., Halverson, C., and Karat, C.-M. (2000). *Overcoming Unusability: Developing efficient strategies in speech recognition systems*. International Conference for Human Factors in Computing Systems (CHI), Amsterdam (NL), ACM, Vol. 2.
- Newman, D. (2000). *Talk to Your Computer: Speech Recognition Made Easy*. Berkeley, CA: Waveside Publishing.
- Nielsen, J. (1993). *Usability engineering*. Morristown, NJ: AP Professional.
- Novick, D. G., Hansen, B., Sutton, S., and Marshall, C.R. (1999). *Limiting Factors of Automated Telephone Dialogues*. in D. Gardner-Bonneau and H. Blanchard (eds.): *Human Factors and Voice Interactive Systems*. Boston/Dordrecht/London Kluwer Academic: pp. 163-186.
- Oviatt, S., DeAngeli, A., and Kuhn, K. (1997). *Integration and Synchronization of Input modes during multimodal Human-Computer Interaction*. International Conference on Human Factors in Computing Systems (CHI), Atlanta (GA), ACM, Vol. 1, pp. 415-22.
- Parnas, D. L. (1969). *On the use of transition diagrams in the design of a user interface of interactive computer systems*. In proceedings of ACM Conference, pp. 379-385.
- Reeves, B. and Nass, C. (1996). *The Media Equation*. Cambridge (UK): Cambridge University Press.
- Resnick, P., and Virzi, R. A. (1995). *Relief from the audio interface blues: expanding the spectrum of menu, list, and form styles*. Transactions on Computer-Human Interaction (TOCHI), Vol. 2, No. 2, pp. 145-176.
- Roberts, T. L., and Engelbeck, G. (1989). The effects of device technology on the usability of advanced telephone functions. In *International Conference on Human Factors in Computing Systems (CHI)*, ACM. Vol. I, pp. 331-338.
- Sacks, H., and Schegloff, E. A. (1974). *A simplest systematics for the organization of turn-taking in conversation*. Language, Vol. 50, pp. 698-735.
- Shneiderman, B. (2000). *The Limits of Speech Recognition*. Communications of the ACM Vol. 43, No. 9.

- Soltau, H. and Waibel, A. (2000). *Acoustic Models for Hyperarticulated Speech*. International Conference on Speech and Language Processing (ICASSP), Beijing (China).
- Suhm, B. (2003). *Towards Best Practices for Speech User Interface Design*. European Conference on Speech Communication and Technology (Eurospeech), Geneva (Switzerland), Vol. IV, pp. 2217-20.
- Suhm, B., Meyers, B., and Waibel, A. (1999). *Empirical and Model-based Evaluation of Multimodal Error Correction*. International Conference on Computer-Human Interaction (CHI), Pittsburgh, PA, ACM.
- Suhm, B., and Peterson, P. (2001). Evaluating commercial touch-tone and speech-enabled telephone voice user interfaces using a single measure. In *International Conference on Human Factors in Computing Systems (CHI)*, Seattle, WA, ACM. Vol. II, pp. 129-130.