

Speech Interface Guidelines*

Alexander I. Rudnicky

School of Computer Science, Carnegie Mellon University

5000 Forbes Avenue, Pittsburgh, PA 15213-3890 USA

Telephone: +1 412 268 2622

email: air@cs.cmu.edu

May 22, 1996

This document provides an overview of speech interface design principles as applied to the range of applications that have been developed at Carnegie Mellon. For the most part these are workstation-based applications based on spoken language understanding technology. Nevertheless the guidelines should be applicable to a wider range of applications.

Speech interfaces have two properties not normally found in more mature interface technologies:

- They are errorful
- Their state is often opaque to the user.

These properties dictate that speech interfaces incorporate a number of specific features. Below are a set of guidelines that should be followed in the design of speech interfaces.

1 State Transparency

A common scene, repeated many times, finds a user who attempts to use a speech system, but nothing happens. It is not apparent to the user (or to others) what might be wrong.

A well-designed speech interface should clearly and continuously indicate its state to the user. Do the following:

*The current version of this document may be found at: <http://www.speech.cs.cmu.edu/rspeech-1/air/papers/SpInGuidelines/SpInGuidelines.html>

1.1 Provide feedback to the user in the form of a graphic state display.

1. Recognition-based applications may have up to four states (READY, LISTENING, RECOGNIZING, BLOCKED) but a minimum of two (LISTENING, BLOCKED) are necessary.
2. A spatial display is better than an multi-state indicator in a single location.
3. Color coding is better than monochrome
4. Redundancy is best (e.g., using both location and color).

State indicators are by necessity placed outside or at the periphery of the active task region, yet users need to monitor these. The above suggestions increase the visual distinctiveness of the feedback display and make it easier for the user to notice (peripherally) a change in status.

1.2 Provide feedback about basic system parameters.

1. A VU meter allows continuous monitoring of whether the system is live; there is some response to any audible input.
2. A continuous sound-level feedback helps shape proper vocal behavior (speaking not too soft nor too loud).
3. A background noise monitor helps the user determine when the environment is suitable for recognition, and suggests a possible reason why the system may be malfunctioning.

1.3 If graphic feedback is impractical provide auditory feedback.

In some applications, no real-estate is available for speech-specific displays. In some situations, such as telephone-based recognition, no display is possible.

1. Auditory feedback should be short and not take away from task time.
 - Minimally, indicate LISTENING. For example, a tone may be used to indicate the start of a listening period.
2. If multiple states are necessary, make tones maximally different. Several dimensions of variation are available:
 - Tone count: one, two, three beeps.
 - Pitch differences; timbre differences.
 - Combinations of the above.

Additional dimensions are available, such as onset/offset (attack/decay) characteristics, amplitude variation and timbre. These may be used provided that easily differentiated tones can be produced. While a principle such as maximal separation in the chosen space is useful for design, distinctiveness should always be empirically verified.

Auditory feedback is often confusing for the novice or casual user, since there is no ready framework for interpreting auditory feedback. Thus it is best to restrict the variety of auditory feedbacks and to make their meaning easy to discern from their context of occurrence. It is also useful to borrow signals whose meaning is already established in other environments, say telephony.

1.4 Make users aware of long BLOCKED intervals, if such can be predicted.

1. Use graphic feedback, such as a progress bar.
2. Use verbal feedback, e.g., “Please wait while I compute...”
3. Use auditory feedback.
 - A “busy” tone.
 - A variable-interval tone stream (which, e.g., speeds up to provide the equivalent of a progress bar).

1.5 At all times, the user should know what can be spoken.

Speech systems know only a restricted language and consequently will not be able to interpret utterances that fall outside the currently active language. On the other hand, users bring into the situation assumptions based on the characteristics of human-human communication (in particular that they are dealing with an intelligent entity with flexible understanding skills).

Guide the user into acceptable language in the following ways:

1. Provide guidance on expected inputs.

Prompts can be phrased in such a way as to indicate the system’s expectation, e.g., “Do you want to proceed? Please say Yes or No”.
2. Display legal inputs on the screen.

If a small number of alternatives are legal, display these in a menu. Alternately, provide legal sentence frames, either on-screen or in a browsable list.
3. Build complete domain languages.

Many tasks are sufficiently restricted in domain that with sufficient observation of user behavior, a reasonably complete domain language can be built. Of course, this only works if the necessary resources are available.

The real problem here is of course the need to learn the interface. A user familiar with the task and experienced with the interface will have much less difficulty staying within the bounds of the language.

1.6 Status displays must be rapidly interpretable, without detracting from the task-related activity.

Very small feedback displays that must be explicitly attended to are worse than ones that can be noted peripherally or that come over a separate mode (such as sound).

1.7 Do not overwhelm the user with feedback.

While feedback is a good thing, too much feedback becomes confusing, as it now requires the user to scan and interpret the display for information relevant to their performance.

1. Avoid display of information primarily of interest to a developer (such as decoding scores, utterance durations, etc.)
2. Any displayed information must clearly contribute to increased interface usability.

2 Input control

2.1 A robust continuous listening system is better than one that requires explicit user action.

1. The user's ability to coordinate button presses with onsets and offsets of utterances is generally poor, due to unfamiliarity with this skill.
2. Interface-specific activity detracts from task-specific activity.

Thus a continuous listening system is always preferable. Sometimes it is essential, as for a telephone-based recognition system.

2.2 A push-and-hold protocol is better than a two-click protocol (start, stop).

In push-and-hold, the two states are mechanically yoked and the user is better aware of what state the system is in (the button needs to be held down).

2.3 A one-click system is an acceptable intermediate.

Only one action needs to be performed, thus the cognitive load on the user is less (since they do not need to remember to initiate an utterance-terminal action).

3 Error recovery

Speech systems will be prone to error in the foreseeable future. There must always be some way to recover from recognition error.

3.1 Offer an Undo capability.

In reality this is a property of the application for which speech is an interface. Some states are reversible, others may not be or may require complex book-keeping.

1. Undo may not be necessary for stateless applications.

The cost of a single transaction governs the usability of a speech interface in this case. A high cost for a transaction (whether primary or undo) decreases usability.

2. Undo is sometimes referred to as Disconfirmation of an input.

3.2 If Undo is not practical, use a confirmation protocol.

The user explicitly confirms an input with an additional utterance (“OK”), after examining a candidate input buffered by the interface.

1. Ideally, confirmation makes use of a sub-state of the decoder that guarantees very high accuracy for the confirm/disconfirm utterance.
2. A less-desirable but more reliable alternative is to use a positive action in some other mode (e.g., a button press) to note confirmation.
3. Provide an interrupt capability.
 - This is always true for systems that decode in longer than real-time.
 - Applications that typically have a long response latency (should) provide interruptability. The speech interface should not interfere with this capability.

4 Error Detection

Speech input may be errorful, though under some circumstances it should be possible to determine that an error has occurred (for example by comparison with a parallel phonetic decoding or noting a discrepancy as a consequence of semantic analysis). Simpler recognition system may be able to detect errors by comparing the recognition score to a threshold value and rejecting all decodings that fall below threshold.

4.1 Generate a CONFIDENCE SCORE for each decoding.

This allows the interface to automatically initiate an error recovery dialog with the user, saving the user the effort of noting the error and initiating a recovery.

1. The discriminability of correct and incorrect decodings is a factor in deciding whether or not to include automatic error recovery. Consider the cost in terms of the application as well. Consider the overall cost of automating error detection.

5 Error Correction

For tasks that require the creation of correct input strings by speech (such as dictation), the system must provide support for error correction. The support need not be entirely embedded in the speech interface, but can make use of facilities provided by the underlying application (for example, a word processor).

5.1 Provide an alternative input mode that overrides speech input.

The user should always be able to regain control of the system, either to reset its state or to enter data through an alternate mode.

5.2 Provide positive methods for identifying the erroneous portions of an input string.

This applies to applications that involve the creation of text or the formulation of very long queries. That is, in cases where the chances of an extended input being recognized perfectly are only moderate.

1. If possible, keep the original waveform.
Users may not be certain of what they said originally and may be led astray by the erroneous decoding.

5.3 For high-accuracy systems, provide a facility for choosing among alternative decodings.

1. Provide close-scoring alternatives for the user to choose from.
Most systems can provide alternative decodings (though they typically will act immediately on the best-scoring one).
2. Provide a controlled selection mode to ensure rapid convergence to the correct decoding.
 - Use either an alternate modality (such as a menu) or a restricted sub-language to achieve this goal.

5.4 In speech-only systems, provide a dialog-based recovery procedure.

Error recovery can be made efficient if the system helps the user identify and correct the error(s).

1. Not all applications require that an exact recognition be recovered; a semantically equivalent interpretation is sufficient.
2. The system must know its state and have some sense of what user goals may be, basing recovery dialog on this information.

6 Logging Performance

Systems used for development should have some provision for logging performance data. Such information can be used for evaluating performance and for diagnostics.

6.1 Record basic recognition parameters for each utterance.

Include parameters such as utterance decoding, duration, recognition time, any error messages.

If possible record relevant state of the application, the date and time, as well as any other information that can help the developer interpret the results.

Ideally record enough information so that the error can be replicated off-line.

6.2 Allow the user to note unusual behavior.

Note that the most likely user in this case is one who is familiar with the technology and is an early tester of the system. Naive users may not have the skills or inclination to record useful information.

1. It should be possible to save the waveform for a just decoded utterance.
2. Users should be prompted to type in a short note stating why this utterance should be of interest to the developer.
3. Save relevant application state, in particular pointers to the acoustic and language models used in decoding.

7 Integration with an application.

7.1 The very simplest applications can use a speech interface that does not take into account the state of the application.

Such interfaces can be generic and can be used with any application that can be scripted or addressed through an emulation based on an existing interface.

1. The application needs to be stateless (or non-modal).
2. The cost of any single transaction must be low.

7.2 More complex applications need to track state, so that actions are interpreted in the proper context.

1. Different user inputs may have different consequences depending on state.
2. Speech input may allow more efficient expression of user goals, which are best implemented through direct access to the functional level of the application as opposed to the emulation of an existing interface.

Tracking state is essential in cases where the user has alternate means of changing application state (for example, input through a non-speech mode).