# Universal Access to Mobile Computing Devices through Speech Input

Bill Manaris
Computer Science Department
University of Southwestern Louisiana
manaris@usl.edu

Valanne MacGyvers
Psychology Department
University of Southwestern Louisiana
macgyvers@usl.edu

Michail Lagoudakis
Department of Computer Science
Duke University
mgl@cs.duke.edu

## Abstract

This paper presents results on a user interface model for providing universal access to mobile computing devices. The model uses a continuous speech understanding engine to provide access to a virtual keyboard and mouse through speech input. This research has been targeted towards users with permanent motor disabilities. However, these results also apply to able-bodied users with temporary, task-induced motor disabilities, such as users performing alphanumeric data entry through a cellular phone keypad. The proposed solution might complement (or even replace) miniaturized keyboards and other physical keyboard alternatives, such as stylus-type "soft" keyboards. Since it only requires a microphone (and perhaps a speaker for feedback) which are already included in many mobile devices, it may allow such devices to shrink considerably in size, as alphanumeric input is no longer bound to a physical area. The paper describes the underlying architecture employed by the system. It presents empirical results addressing the effectiveness of this interface over alternative input methods for alphanumeric data entry. Finally, it discusses implications and future directions.

## Introduction

An effective user interface has to address several issues. Initially, it needs to help the user develop an accurate conceptual model of the application domain; if the user already has such a model, due to earlier experience with the application domain, the interface needs to comply with and perhaps build on this model. Then, it has to provide an effective mapping between the user's conceptual model and the underlying application. Finally, it should make good use of existing metaphors, whenever possible, to improve learning and retention as well as overall user satisfaction.

User interface developers for mobile computing devices face an additional problem: Size and weight become major constraints that may affect the marketability and viability of a platform. Consequently, the developer has to find resourceful solutions for incorporating the necessary input/output devices. Since most mobile devices attempt to provide a user interface that allows at least entry of alphanumeric characters, much of the available device area is consumed. Conceptually, this area corresponds to a two-dimensional matrix whose resolution depends on the application. This matrix maps the user's motor control to alphanumeric characters. Since this mapping is not always one-to-one, software may be used to provide for disambiguation (e.g., Tegic's T9 software). (Comerford 1998). Examples include miniaturized keyboards in cellular telephones and hand-held PCs, as well as stylus-based "soft" keyboards (see Figure 1). Some devices use time as a third dimension to further reduce the required matrix area, as in the PalmPilot handwriting recognition device. The physical area used for alphanumeric input is sometimes used for visual feedback, as in the case of touch-sensitive screens (e.g., PalmPilot, and HP 300).

The keyboard (physical or "soft") is not the most effective input device for every task. There exist many tasks that can be better performed through alternate modalities, such as point-and-click and speech. Nevertheless, since the keyboard is a *de facto* "universal" interface for general computing devices (i.e., the *typewriter* computer metaphor), it is a requirement on any mobile device that supports access to general computing. For instance, using a cellular phone with a keyboard interface a user can send e-mail or enter data into a spreadsheet stored on a remote PC.

This keyboard interface requirement renders mobile devices at least as large as the size of the device they

**Figure 1**. Various mobile computing devices (left to right: NOKIA 9110, Hewlett Packard 300 Series, 3Com PalmPilot).

employ for alphanumeric data entry. Given Moore's law[1], very soon this requirement will become a major obstacle in the continued reduction of mobile device size. In other words, as long as alphanumeric input requires physical device area, potential size (and weight) reduction of mobile devices is rigidly constrained.

## A Virtual Keyboard

Goldstein et al. (1998) propose an innovative solution for removing the constraint imposed by the keyboard interface. Their solution involves a virtual keyboard accessible through electromyography (EMG) sensors. These sensors perceive the muscular contractions of the user's fingers as (s)he is "typing" on any flat surface. Keyboard characters are assigned to each finger using the traditional placement of fingers on a QUERTY keyboard. Information as to which finger was pressed is sent using wireless signal transmission to the mobile device, e.g., cellular phone. Since input is one-dimensional (e.g., finger "1" was pressed, then finger "6", followed by finger "2"), the system uses a language model to disambiguate among alternative keys accessible by each finger.

One advantage of this solution is that any flat surface will do. However, it presupposes that such a surface is readily available and that there is enough "elbow room" to utilize it – obviously, this would not be the case in a crowded bus, for example, or while walking in the street. It also presupposes that the user is an expert "blind" typist on a QUERTY keyboard. Finally, it does not handle cursor movements, concurrent key presses, as well as point-and-click operations. However, its major contribution is that it completely disassociates alphanumeric input from the physical device area. Thus it eliminates keyboard-based constraints on physical size requirements of mobile devices.

## *SUITEKeys*[2] User Interface

The solution proposed herein is to provide a speech user interface that models a virtual keyboard and mouse. This interface, named *SUITEKeys*, provides an one-to-one mapping between user utterances and keyboard/mouse-level operations, such as pressing/releasing a key and moving the cursor a certain distance/direction. Thus, it provides access to the complete functionality of any computing device (mobile or not) similarly to a physical keyboard/mouse interface.

*SUITEKeys* is based on a speech understanding architecture that encapsulates a hybrid language model consisting of statistical and symbolic components. It accepts regular and military alphabet pronunciation. It supports additional features such as selection from a list of frequently entered words, and the ability to switch into a non-active (sleep) state (so that another, task-specific speech application may be used). The list of frequently entered words is based on techniques for statistical word prediction that have been shown to speed up word data entry (Copestake 1996; Tegic Communications, Inc.). Finally, it supports modeling of task-specific activities, such as dialing a number, managing e-mail communication, and maintaining verbal macros for commonly used operations.

Similarly to the EMG-sensor solution discussed above, it disassociates keyboard input from the physical area of the mobile device, and thus eliminates constraints on size reduction. Additionally, in the case of devices that already have a microphone and speaker (e.g., cellular phones, personal digital assistants, palm-held PCs), it requires no additional physical apparatus. One drawback of this solution is that it may not be appropriate in public places due to privacy issues. However, this is also true for traditional use of cellular telephones, which are very prolific in spite of this drawback due to the other advantages they offer, such mobility and reduced size.

*SUITEKeys* originated from research targeted to users with motor disabilities (Manaris and Harkreader 1998). However, it soon became clear that it also applies to users with temporary, task-induced motor disabilities. This includes users involved in tasks that "monopolize" hand motor control (such as driving a car or servicing a jet engine). It also includes users performing alphanumeric data entry on mobile devices. As the dimensions of traditional input devices shrink, the motor skills of the user become less effective, almost inadequate. For instance, studies show that users may experience severe reduction in data entry speed (wpm) when switching from a regular QUERTY keyboard to a mobile device keyboard alternative. One study reports a 4:1 reduction on a stylus-based "soft" keyboard (PalmPilot) (Goldstein et al. 1998). Another study reports a 3:1 reduction on a telephone keypad (MacKenzie et al. 1998).

## System Architecture

The architecture of the *SUITEKeys* user interface is based on SUITE, a framework for developing speech understanding interfaces to interactive computer systems (Manaris and Harkreader 1997). As shown on Figure 2, the system architecture integrates speech recognition and natural language processing modules. When used with

[1] Moore's law states that the *power/area ratio* of integrated circuits doubles every 18 months. It is suspected that, as integrated circuits begin growing in the third dimension, this law will no longer hold. This is because, strictly based on geometric principles, the *power/volume ratio* should grow much faster than the power/area ratio of traditional 2D designs.

[2] *SUITEKeys* is an application of research on **S**peech **U**nderstanding **I**nterface **T**ools and **E**nvironments.

mobile devices that are equipped with microphone and speaker, such as cellular phones and personal digital assistants, it requires no additional hardware. The complete system is encapsulated within the mobile device and runs on top of the device's operating system like any other application.

The architecture consists of knowledge-base and processing components. These are organized in a pipelined fashion for converting user utterances to the corresponding instructions for the mobile device operating system. These components are described in the following sections.

## Dialog Management

The dialog manager encapsulates a multi-layered dialog architecture (Agarwal 1997; Jönsson 1997). This architecture is modeled as a dialog grammar containing dialog states and actions. The top grammar layer consists of domain independent states, whereas the lower layers consist of domain specific states. Each rule represents an AND/OR graph. AND nodes represent dialog states that need to be satisfied sequentially, whereas OR states represent alternative dialog paths. Additionally, each state may be associated with a specific language model (e.g., lexicon, grammar) which may help focus speech understanding, thus increasing accuracy. Actions associated with dialog states may be performed before a state is visited (pre-actions), or after a state is visited (post-actions). Pre-actions may be used as preconditions, i.e., if the action fails, the associate dialog state is not explored. Post-actions may be used to maintain knowledge about the interaction as well as guide other processing components.

A focus structure is provided to help interpret referents (e.g., "this", "it") as well as other context-dependent phenomena. This focus structure holds a dialog state's focal parameters and can be accessed by the state's actions. The focus structure of a new dialog state is initialized using the focus structure of the immediately preceding state. This accounts for most context-dependent utterances (Jönsson 1997).

This model supports mixed-initiative dialogs, in that either the system or the user can initiate a (sub)dialog at any point during interaction (Agarwal 1997). This is accomplished by checking the user input against the upper-layer of the dialog grammar, and/or against a collection of task-specific, user-interrupt dialog states. Mixed-initiative dialogs are supported by a set of actions for pushing/popping a dialog state, starting a new dialog, or ending the current dialog state. The model includes a dialog stack whose size may be limited to accommodate memory constraints on mobile devices. Additionally, appropriate verbal prompts may be constructed to help the user develop a conceptual model of the current dialog state, e.g. "You have new e-mail. Would you like to read it?"

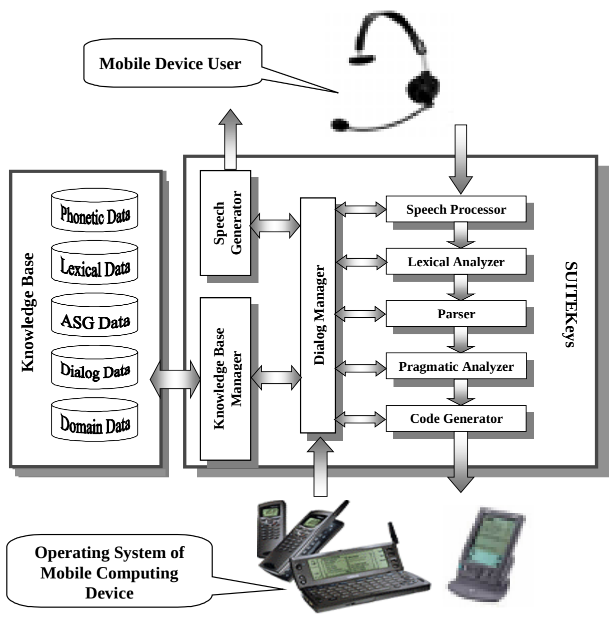

**Figure 2.** SUITEKeys Architecture

(Yankelovich 1998). Finally, this module handles exceptions/errors that are raised by other components. In mobile devices with limited (or non-existent) displays, it may use the speech generator to provide feedback to the user, if necessary.

## Speech Processing

Originally speech processing was performed through a neural-network based phoneme probability estimator (Manaris and Harkreader 1997). This accepted a continuous stream of input feature vectors and produced a phoneme probability matrix that was given to a Viterbi process. This process generated the N-best hypotheses as to what sequences of phonemes had been spoken. Due to the dramatic evolution in continuous speech processing engines within the last two years, our architecture now implements the Microsoft Speech API (SAPI). Thus it can support any SAPI-compliant engine. Currently, we use the NaturallySpeaking SDK provided by Dragon Systems (Dragon NaturallySpeaking SDK).

## Lexical Analysis

Lexical processing is performed through a dictionary of lexemes (words supported in the application domain). Each dictionary entry defines a lexeme in terms of its syntactic/semantic category as well as additional linguistic or extra-linguistic information. For example,

```
"go" : ([GO-VERB]  {other info})
"to" : ([TO-PREP] {other info})
"sleep" : ([SLEEP-VERB] {other info})
"b" : ([B-LETTER] (homophones b p) …)
"bravo" : ([B-LETTER] {other info})
"d" : ([D-LETTER] (homophones d t) …)
```

This process allows for modeling of *homophones* (lexemes that sound alike). For instance, in the *SUITEKeys* domain, there exist several lexemes that can be easily confused by a speech engine (such as "b" and "p", and "d" and "t"). This is not necessarily a shortcoming of the speech engine, as these lexemes are easily confusable by humans as well – hence the invention of the military alphabet.

Since speech engines are normally not customizable to resolve ambiguities at the level of individual letters, we provide a statistical disambiguation algorithm based on *overlapping trigrams*. This is similar in function to the algorithms used by the T9 software (Tegic) and Goldstein et al. (1998). Specifically, trigrams are extracted from *textual units* (a sequence of tokens delimited by whitespace characters) by moving a "window" of three characters along a textual unit (Adams and Meltzer 1993). For each possible trigram, the dictionary includes the probabilities of its occurrence within different positions in a textual unit. For instance,

```
"abc"  :([TRIGRAM]  (probabilities  (1  .02)
        (3 .05) (7 .0023) ...)
```

For each textual unit in the input, the algorithm cycles through all trigrams (starting with the first) keeping track of position within the textual unit. It replaces each character c with its set of homophones $c_h$. For example, ("pat") should be replaced by (("b" "p") ("a") ("d" "t")). From this set of potential trigrams, e.g., ("bad" "bat" "pad" "pat"), it picks the most probable trigram given the trigram's position in the textual unit and uses it to construct the disambiguated textual unit.

## Other Components

The *SUITEKeys* architecture incorporates the following additional components. Due to space limitations, only an overview is given This is because these components are either traditional in nature and/or are described elsewhere (Manaris and Dominick 1993; Manaris and Harkreader 1997):

**Parser:** This is a left-to-right, top-down, non-deterministic parser. It allows for multiple parse trees, thus providing for ambiguity handling at the semantic level. It incorporates semantic actions for constructing semantic interpretations of user input.

**Pragmatic Analyzer**: This module further refines semantic interpretations produced by the parser using predicate-calculus logic constraints. These constraints are dynamic, in that they may be updated as the result of an action originating from other components such as the parser, error handler, and dialog manager.

**Code Generator**: This module converts semantic interpretations to low-level code understood by the operating system of the mobile device.

**Knowledge-Base Manager**: This module coordinates interaction of other components with the *SUITEKeys* knowledge base.

**Speech Generator**: This module provides primitives for text-to-speech conversion. It is supported by the SAPI-compliant speech engine.

## *SUITEKeys* Prototype

The *SUITEKeys* prototype is being developed as an interface to the Microsoft's Windows®95 and NT 4.0 operating systems. It is currently implemented in Visual BASIC, Visual C/C++, and LISP using the NaturallySpeaking SDK. Each component is either an ActiveX control or a dynamic link library (DLL). This allows objects written in one ActiveX-aware language to communicate with objects written in a different ActiveX-aware language. Languages that support ActiveX objects include *Microsoft Visual Basic*, Microsoft and Borland implementations of C++, *Microsoft Java 6.0*, and *Borland Delphi*. Once the prototype is refined, the next step will be to port it to Windows CE® which is supported by a multitude of mobile devices.

## Usability Evaluation

This section reports on two experiments designed to assess the following hypothesis: *Speech input as provided by* SUITEKeys *is an effective input modality for motor-challenged users.*

### Pilot Study

The first experiment served as a pilot study to help refine the *SUITEKeys* functional requirements, as well as assess the methodology for the second, full-scale evaluation study. In this study, subjects where asked to type in a short paragraph on a Windows95 platform using two alternative input modalities: (a) their preferred input modality, and (b) a Wizard-of-Oz prototype[3] of *SUITEKeys*. The use of a Wizard-of-Oz prototype allowed as to

- constrain subjects to the application domain,
- provide freedom of expression, and
- evaluate and refine application requirements, such as vocabulary, grammar, interaction patterns, design of prompts/feedback, as well as the overall concept effectiveness (Yankelovich 1998).

[3] A Wizard-of-Oz experiment involves a human acting as a hidden interface between the user and the computer; the human is simulating the functionality of the system under study.

This study was carried out in the fall of 1997 and involved three motor-disabled users as subjects. To collect data for analysis, we captured the computer display and room audio on video, logged keyboard and mouse events, and examined the text files created by the subjects. The results, although not statistically significant, supported the hypothesis – overall, subjects performed better using speech, in terms of task completion rate, typing rate, and error rate. For a detailed description of this study and its results, see (Manaris and Harkreader 1998).

## Main Study

The follow-up, full-scale study was carried out in the summer of 1998. It involved 43 psychology students who participated as part of their course requirements. In this study we compared one of the standard input modalities, the *mouthstick*, with the *SUITEKeys* Wizard-of-Oz prototype. Subjects where asked to type in and save a one-paragraph document in each condition. Two linguistically equivalent paragraphs were used as sources (one in each condition). Although the subjects where not permanently disabled, a motor challenge was introduced through the use of a *handstick.* Specifically, subjects where asked to hold an unsharpened pencil with both hands, having hands held under the chin, thus simulating a *mouthstick* or single-digit typing.

We assume that the *handstick* effectively simulates a range of alternative input modalities that have the following characteristics: (a) decrease physical input area (e.g., miniaturized keyboards), (b) increase visual scan time (e.g., stylus-type "soft" keyboards), and (c) add a third dimension through time (e.g., handwriting recognition devices). Such input modalities share the following symptoms: (a) decreased data entry rate, (b) decreased task completion rate, and (c) increased error rate. We are currently working on a post-experiment study to evaluate this assumption.

Subjects were randomly assigned to the speech-first or handstick-first condition. Following both tasks, subjects completed a brief questionnaire assessing their impressions of the two interface procedures. A query assessed whether the participant had suspected the Wizard-of-Oz nature of the experiment. Only 7 of the 43 reported a suspicion, and only 3 of those were based on rational evidence. The questionnaire also included an item where subjects self-reported their level of expertise with computer environments like MS Windows® or Macintosh®. This was used to generate three groups of users, namely Novice, Intermediate, and Expert.

## Data Analysis

The results of the analyses of variance suggest that the *SUITEKeys* prototype would be easily learned by users, especially when compared to the labor-intensive alternative. Each of the four summative measures showed a significant main effect of condition favoring the speech condition. These effects are most pronounced for novices. Table 1 presents the group means and statistics. As shown on Figure 3 and Table 1, users performed best in the speech condition (i.e., took less time, typed faster, were more complete, and made fewer errors). Note that while the means in CompletionRate are very similar, the variance is also very small, so those differences are significant. A significant main effect of User Level was found for *TotalTime* and *TypingRate*. This indicates that novice users took longer to complete the task and were slower typists. Examination of means shows that this was truer in the handstick condition than in the speech condition. There were no significant interactions between Condition and User Level.

It should be noted that one significant order effect was found. Participants made more errors in the speech condition when it was preceded by the handstick condition (F (1,41)=5.31, p=.03), which we interpreted as a fatigue

| **Variable** | **User Level (n)** | **Means Handstick Condition** | **Means Speech Condition** |
|---|---|---|---|
| *Total Time* | Novice (11) | 359.1 | 207.6 |
| | Intermediate (12) | 329.0 | 219.6 |
| | Expert (20) | 295.5 | 187.3 |
| | All (43) | 321.1 | 201.5 |
| *Completion Rate* | Novice (11) | .984 | .996 |
| | Intermediate (12) | .994 | .999 |
| | Expert (20) | .992 | .998 |
| | All (43) | .991 | .998 |
| *Typing Rate* | Novice (11) | .682 | 1.290 |
| | Intermediate (12) | .749 | 1.203 |
| | Expert (20) | .821 | 1.440 |
| | All (43) | .766 | 1.336 |
| *Error Rate* | Novice (11) | .083 | .034 |
| | Intermediate (12) | .104 | .021 |
| | Expert (20) | .086 | .040 |
| | All (43) | .090 | .033 |

| **Variable** | **Main Effects ANOVAs** | **P(F)** |
|---|---|---|
| *Total Time* | Condition F (1,40) = 110.42 | .0001 |
| | User Level F (2,40) = 5.89 | .0057 |
| *Completion Rate* | Condition F (1,40) = 6.90 | .0122 |
| | User Level F (2,40) = 2.15 | .13, ns |
| *Typing Rate* | Condition F (1,40) = 127.82 | .0001 |
| | User Level F (2,40) = 4.85 | .0131 |
| *Error Rate* | Condition F (1,40) = 23.8 | .0001 |
| | User Level F (2,40) = 0.05 | .949, ns |

**Table 1.** Selected Results of Main Study

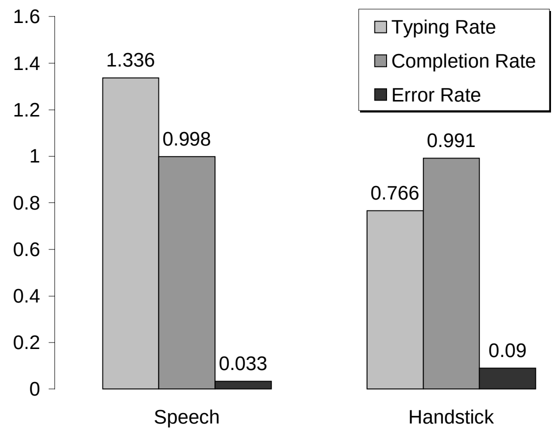

**Figure 3**. Rate Means by Condition

effect. This conclusion is supported by the questionnaire data.

The questionnaire presented eight statements about the two systems. To each statement, participants indicated their level of agreement using a six-point scale, where 1 indicated strong disagreement and 6 strong agreement. The first item stated, "The voice controlled interface was easier for me to use than the hand stick." The mean level of agreement was 5.35, supporting the fatigue interpretation.

Participants also indicated that the speech condition worked well (M=5.58) and that they would want to purchase a program like this (M=4.38), especially if they were physically disabled (M=5.67). Further, the participants felt they would prefer a speech-activated system to a handstick- or mouthstick-activated one if they were disabled (M=5.51). Finally, users generally disagreed with statements suggesting that the speech-activated system was not user-friendly (M=1.23), too confusing to work with (M=1.22), and harder to learn than the handstick (M=1.42).

## Conclusion

This paper presented on-going work on a speech user interface for providing universal access to mobile computing devices. This model uses a continuous speech understanding engine to provide access to a virtual keyboard and mouse through speech input. Although this work was originally targeted towards computer users with permanent motor disabilities, it also benefits able-bodied users with temporary, task-induced motor disabilities, such as users performing alphanumeric data entry through a cellular phone keypad.

The conducted study indicates that speech interaction with a virtual keyboard and mouse, as implemented *SUITEKeys*, is a very effective input modality in terms of user data entry, task completion, and error rates. Moreover, it suggests that this modality is far better than alternative modalities used in mobile devices that require physical manipulation of a device component for alphanumeric data entry. Such modalities are characterized by decreased physical input area, increased visual scan time, and/or increased character specification time (e.g., handwriting recognition). A speech user interface similar to *SUITEKeys* would be relatively easy to learn and to use, particularly for the motor disabled and/or computer illiterate user. Anecdotal evidence from the novice subjects of the study suggests that this system is far less intimidating than other interfaces. Of course, these results hold for users without significant speech impediments and, currently, only in low-noise environments. It is expected that improvements in microphone technology will minimize the low-environment-noise constraint.

Although speech is not the best modality for all human-computer interaction tasks, when delivered at the level of keyboard and mouse it allows for universal access to computing devices – similar to the one enjoyed through a standard QUERTY keyboard and mouse. Thus, the proposed solution might complement or even replace miniaturized keyboards in many application domains, as well as other physical keyboard alternatives, such as stylus-type soft keyboards. Since it does not require much physical device area for alphanumeric data entry (only microphone and perhaps speaker, for feedback), the physical device may shrink as much as advances in microelectronics may allow. Considering Moore's law, this result is of significant importance. It's only a matter of time (perhaps in the order of a few years) before new delivery platforms for computing applications may be successfully exploited, such as eyeglass frames, watches, and perhaps even body implants (e.g., tooth crowns). Although the latter raises significant ethical issues, it will also provide for innovative solutions to a variety of problems faced by disabled as well as able-bodied people.

## Acknowledgements

This research is partially supported by the (state reference) Board of Regents grant BoRSF-(1997-00)-RD-A-31. The authors acknowledge the contribution of Rao Adavikolanu, Alan Harkreader, Jonathan Laughery, Eric Li, Christopher Schmidt, Rohit Sonak in the implementation phase of this work. The authors would like to thank Najwa Dibbs, Page Salley, Nona Wilson, and the subjects for their contribution to the usability study. Finally, Larry Duplantis, and James Wright provided feedback on initial user requirements.

## References

1. Adams, E. S.; and Meltzer, A. C. 1993. Trigrams as Index Elements in Full Text Retrieval. In *Proceedings of the ACM 21st Annual Computer Science Conference*, 433-439. New York, NY: ACM Press.

2. Agarwal, R. 1997. Towards a PURE Spoken Dialogue System for Information Access. In *Proceedings of the ACL/EACL Workshop on Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications*, 90-97. New Brunswick, NJ: ACL.
3. Comerford, R. 1998. Pocket Computers Ignite OS Battle. *IEEE Spectrum* 35(5): 43–48.
4. Copestake, A. 1996. Applying Natural Language Processing Techniques to Speech Prostheses. In *Working Notes of the 1996 AAAI Fall Symposium on Developing Assistive Technology for People with Disabilities*. Menlo Park, CA: AAAI Press.
5. Dragon NaturallySpeaking SDK. http://www.dragonsystems.com
6. Goldstein, M.; Book, R.; Alsio, G.; and Tessa, S. 1998. Ubiquitous Input for Wearable Computing: QUERTY Keyboard without a Board. In *Proceedings of the First Workshop on Human Computer Interaction with Mobile Devices*, Glasgow, Scotland. http://www.dcs.gla.ac.uk/~johnson/papers/mobile/HCIMD1.html
7. Jönsson, A. 1997. A Model for Habitable and Efficient Dialogue Management for Natural Language Processing. *Natural Language Engineering* 3(2/3): 103–122.
8. MacKenzie, I. S.; Zhang, S. X.; and Soukoreff, R. W. 1998. Text entry using soft keyboards. Forthcoming. http://www.uoguelph.ca/~imackenz/SoftKeyboard.html
9. Manaris, B.; and Dominick, W. 1993. NALIGE: A User Interface Management System for the Development of Natural Language Interfaces, *International Journal of Man-Machine Studies*, 38(6): 891–921.
10. Manaris, B.; and Harkreader, A. 1997. SUITE: Speech Understanding Interface Tools and Environments. In *Proceedings of FLAIRS '97*, 247–252. St. Petersburg, FL: Florida AI Research Society.
11. Manaris, B.; and Harkreader, A. 1998. SUITEKeys: A Speech Understanding Interface for the Motor-Control Challenged. In *Proceedings of The Third International ACM Conference on Assistive Technologies (ASSETS '98)*, 108–115. New York, NY: ACM Press.
12. Yankelovich, N. 1998. Using Natural Dialogs as the Basis for Speech Interface Design, In *Automated Spoken Dialog Systems*, MIT Press. Forthcoming.
13. Tegic Communications, Inc. http://www.tegic.com