

: A Wizard of Oz Prototyping Tool for Speech User Interfaces

Scott R. Klemmer, Anoop K. Sinha, Jack Chen, James A. Landay, Nadeem Aboobaker, Annie Wang

Group for User Interface Research
CS Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720-1776 USA
+1 510 643 3043

landay@cs.berkeley.edu • <http://guir.berkeley.edu>

ABSTRACT

Speech-based user interfaces are growing in popularity. Unfortunately, the technology expertise required to build speech UIs precludes many individuals from participating in the speech interface design process. Furthermore, the time and knowledge costs of building even simple speech systems make it difficult for designers to iteratively design speech UIs. SUEDE, the speech interface prototyping tool we describe in this paper, allows designers to rapidly create prompt/response speech interfaces. It offers an electronically supported Wizard of Oz (WOz) technique that captures test data, allowing designers to analyze the interface after testing. This informal tool enables speech user interface designers, even non-experts, to quickly create, test, and analyze speech user interface prototypes.

Keywords

Wizard of Oz, speech user interfaces, prototyping, design, low-fidelity, informal user interfaces, design tools

INTRODUCTION

Speech-based user interfaces are more appropriate than graphical user interfaces in many settings and thus will likely become a more common user interface paradigm in the near future [10]. However, speech-based user interfaces are difficult to design well. Speech UIs are hard to prototype due to high costs in terms of the time and technology expertise required to build them. This makes iterative design difficult and often results in poor user interfaces. We have developed an interactive prototyping tool called SUEDE to address this problem. SUEDE is a lightweight, easy-to-use design tool based on the concepts of example-based test scripts, prompt/response state transitions, Wizard of Oz studies, and analysis of user study data.

Why Speech-based UIs

Today's graphical user interfaces (GUIs) do not let users

communicate in ways that they naturally do with other human beings [35]. Additionally, a non-trivial percentage of the U.S. population is blind or has trouble seeing words in ordinary newsprint (5% of those over age 15 [29]). Many others have limited literacy skills (21% of those over age 16 [40]), typing skills, or use of their hands. The standard GUI does not work well for these users or for others in many situations: when users are moving around, using their hands or eyes for something else, or interacting with another person. To enjoy the benefits of ubiquitous computing [46], we need newer, better interface paradigms. Speech-based user interfaces are one paradigm that can successfully address many of the aforementioned problems.

Supporting Speech Interface Designers

Although many researchers and industry analysts believe that speech user interfaces will become commonplace, there are a number of factors that hinder their incorporation into everyday use.

A key limiting factor in speech interface design is the lack of basic knowledge about user "performance during computer-based spoken interaction" [10]. Many interaction designers who could contribute to this body of knowledge are excluded from speech design by the complexities of the core technologies, the formal representations used for specifying these technologies, and the lack of appropriate design tools to support iterative design.

The complex recognizer and synthesizer technologies inherent in a speech UI require a high level of technical competency to understand. These systems generally have a large number of "knobs" and parameters that must be meticulously adjusted in order to get optimum performance. Previous research studying the use of visual design tools showed that when given these knobs, designers spend many hours manipulating the parameters rather than exploring the design space [4, 15].

The most effective method for constructing high quality user interfaces is an iterative approach [16]. This requires a fast, repeated cycle of design, prototyping, and evaluation. Therefore, to be successful, a speech interface design tool must be easy to learn, require little programming expertise to use, and support the rapid creation, testing, and modification of interface designs. These requirements form

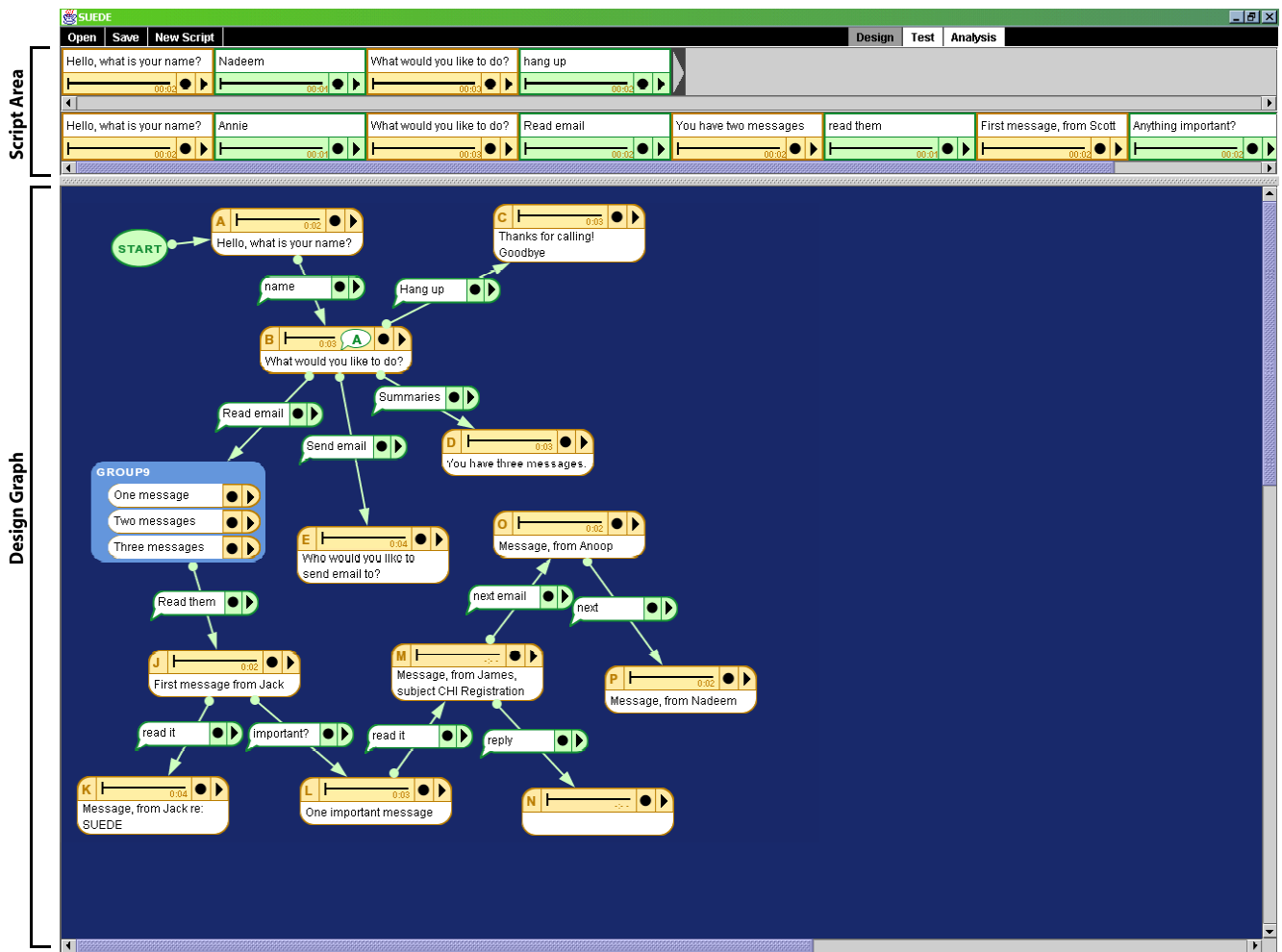


Figure 1. SUEDE's Design mode allows the easy creation of example scripts (*top*) and speech UI designs (*bottom*).

the basis of any user interface prototyping tool targeted towards interface designers [45].

The grammar and state machine representations used to design speech-based systems are formal and abstract. This is in awkward contrast with the informal, concrete representations, such as scenarios [6, 8], sketches [5, 22], and storyboards [23, 24, 45], that designers commonly employ to explore design ideas through examples. Designers typically start with usage scenarios and move towards abstractions over time.

This basic mismatch in approach is an important issue that must be resolved for any tool to be successful. We have previously argued that an "informal interface" approach, using unrecognized, natural input (e.g., sketching or speech) successfully supports designers in the early stages of design [18, 24]. Using these tools, designers are encouraged to explore the design space rather than detail one design idea too far. SUEDE embodies this approach.

The rest of this paper is organized as follows. We first give an overview of the Design / Test / Analysis methodology used by practicing speech UI designers and supported in SUEDE. After that section, we describe in detail how SUEDE supports the early stages of the speech user

interface design process. Next, we give an overview of the system implementation. We then review the related work in prototyping methodologies and tools for speech user interface design. We finish with future plans and conclusions.

DESIGN / TEST / ANALYSIS METHODOLOGY

SUEDE's design was based on interviews with six speech UI designers from both industrial research (SRI and Sun Microsystems Laboratories) and development organizations (Nuance Communications and Sun Microsystems).

SUEDE's interface is organized around three important phases that these designers perform in the early stages of design: Design, Test, and Analysis. In the *Design* phase, they often begin by creating conversation examples (see Figure 1, top). These examples evolve into, and provide the foundation for, the actual interface design (shown in progress in Figure 1, bottom). In the *Test* phase, they try out the design with target users. During *Analysis*, designers examine collected test data, deciding how it should influence the next design iteration.

The problems with real world speech user interfaces arise when users do not know what to say, speak outside the system's grammar, or say something that is misrecognized

by the system. The designers we spoke with wanted more help analyzing their test data to deal with these types of errors.

We developed the Design / Test / Analysis methodology as an explicit model for interface design tools as a result of remarks made by Edward Tufte at his *Presenting Data and Information* seminar (December 8, 1999 in San Francisco). Tufte argued that usability testing, as practiced today, does not work since it entails repeated cycles of Design and Test, resulting in a UI popularity contest. He argued instead that good design was a process of repeated application of Design and Analysis. We believe both approaches are insufficient; the approach that most successful design and usability specialists use is the repeated application of Design, Test, *and* Analysis. SUEDE provides an interface mode for each of these three tasks (see Figures 1, 2, and 3).

HOW SUEDE WORKS

We will explain the functionality of SUEDE using the example of someone who is designing a telephone-based speech interface for reading and sending email, as in the MailCall system [26]. A typical session in this system starts by prompting the user for their name and then asking them what they want to do: read email, send email, get summaries, or hang up.



Design Mode

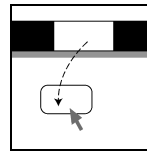
Speech designers often begin the design process by writing linear dialog examples in a word processor [33]. SUEDE allows linear dialog examples to be created horizontally in the top area, called the *script area*, of Design mode (see Figure 1, top).

Prompts are recorded by the designer for the phrases that the computer speaks. *Responses* are the phrases that participants make in response to prompts. System prompts alternate with user responses for accomplishing the different tasks. The orange *prompt cards* in the script represent the system's speech prompts and the green *response cards* represent example responses of the end-user. The designer can record her own voice for the speech on both types of cards, as well as type in a corresponding label for each of the cards. By playing the recordings from left to right, the designer can both see and hear the example interaction.

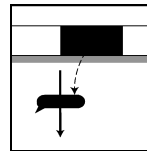
We can see from the second script at the top of Figure 1 that the designer has recorded the following alternating prompts and responses: "Hello, what is your name?", "Annie", "What would you like to do", "Read email", "You have two messages", "Read them", "First message from Jack" and "Anything important?".

After creating the examples, typically on index cards or small pieces of paper, a designer creates a flowchart representation of the dialog flow. In SUEDE, after constructing several example scripts and working with them to become familiar with the dialog flow, the designer starts to construct a *design graph*, representing the actual interface prototype (see Figure 1, bottom). The design

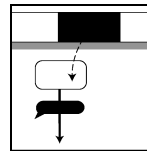
graph represents the dialog flow based on the user's responses to the system's prompts. Designs are created by dragging cards from the script onto the design area, creating new cards on the design area, and linking them into the dialog flow. The orange script cards become orange *prompt cards* () in the design area. The green script cards become green *response links* () in the design area. The important interaction techniques are illustrated in the below diagrams:



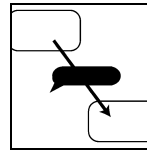
Dragging an orange script prompt card to the design area creates an orange prompt card.



Dragging a green script response card to the canvas area creates a green response link in the design graph.



Dragging a script response card onto an existing design graph prompt card creates an outbound response link from that prompt.



A left mouse drag gesture between two cards creates a link. Dragging from the background onto a prompt creates a global.

Globals, Groups, and Voice Balloons

SUEDE also supports special types of prompt cards called globals and groups, as well as a mechanism, called a voice balloon, for parameterizing a later prompt with a test participant's actual speech response.

A *global* is a speech command that can be spoken at any time during an end-user test session, such as "main menu" or "help." In Test mode, clicking on a global will transition the participant to the connected prompt card.

A *group* is a set of prompt cards that are possible prompts following a specific participant response. As an example:

Prompt: "Welcome to the weather system."

Response: "What is the weather like in San Francisco?"

Prompt Group:

"It is sunny"

"It is raining"

"It is foggy"

Another example is the "Message Options" group in Figure 1, containing the possible prompts resulting from the "read email" command.

The wizard has the option of choosing among any of these replies to the participant when testing. Each of these has the

same logical structure in the interface. Above, the designer used the message options group to enable the wizard to test different scenarios, yielding an interface with the appearance of being database backed (as the fully implemented system probably would be). Groups can also be used for tapering, a speech technique in which different prompts are given, based on the number of times the prompt has been previously played. Groups can also be used to give different error messages to different participants.

A *voice balloon* corresponds to “filling in the blanks” in a directed dialog system, where the user’s responses are used in the subsequent prompts. The participant’s unmodified recorded audio is automatically spliced into a later prompt. An example of this would be:

Prompt A: “What flight would you like?”

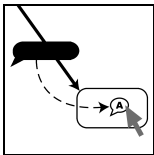
Response: “AIR2000”

Prompt B: “What day would you like to take the flight?”

Response: “Tuesday”

Prompt C: “The schedule for <AIR2000> on <Tuesday> is...”

Voice balloons are added to a prompt as follows:



A voice balloon is added to a card by dragging from a response link onto a prompt card.

A voice balloon represents the run-time response of a test participant. It can be used in a prompt at design time as a placeholder for a participant’s utterance. In the flight scheduling example above, the designer created a voice balloon corresponding to a user’s response to the prompt “What flight would you like?” This voice balloon was then dragged onto the later prompt “The schedule for,” resulting in “The schedule for <A>”. At run time, <A> would be filled in with the user’s own response to the first prompt.

A Scalable Design Visualization

The designers that we spoke with were building prompt and response interfaces of about 200 nodes. One designer currently using Visio for early stage design felt frustrated that her designs were spread across a dozen pieces of paper (her Visio designs had about fifteen nodes per page). She was interested in having the entire design fit on one display.

SUEDE supports scaling in two ways. First, scrollbars allow designers to pan around interfaces that are larger than one screen. More importantly, prompts and groups can be shown at three different scales. At their largest, prompts display their full text and all the prompt audio controls (prompt M in Figure 1). By default, prompts display one line of text and all the audio controls (prompt A in Figure 1). At their smallest, prompts display one line of text, and

only the record and play controls (the prompts inside “Message Options” in Figure 1).

Groups have three similar scales. Their large scale displays all their prompts at their full size. By default, groups display all prompts at their compact size (“Message Options” in Figure 1). When compacted, groups display only their title. Designers can switch between these representations by gesture. A left mouse gesture upward expands a prompt or group, and a left mouse gesture downward contracts a prompt or group.

Test Mode

SUEDE designs can be immediately executed by clicking on the Test button. The designer can try out her design ideas as soon as she has created them without the need for a speech back-end. That is, no speech recognition or speech synthesis is necessary to create prototypes in SUEDE.

Wizard of Oz methodologies have a long tradition in the design of speech systems [16]. In conventional WOZ studies, designers test scenarios manually by walking through the different steps in a dialog flow. The wizard simulates dialog transitions as a computer would, reading the system prompts to the participants and “processing” their responses.

In SUEDE, the designer switches to Test mode by clicking on the Test button in the upper right corner of the main

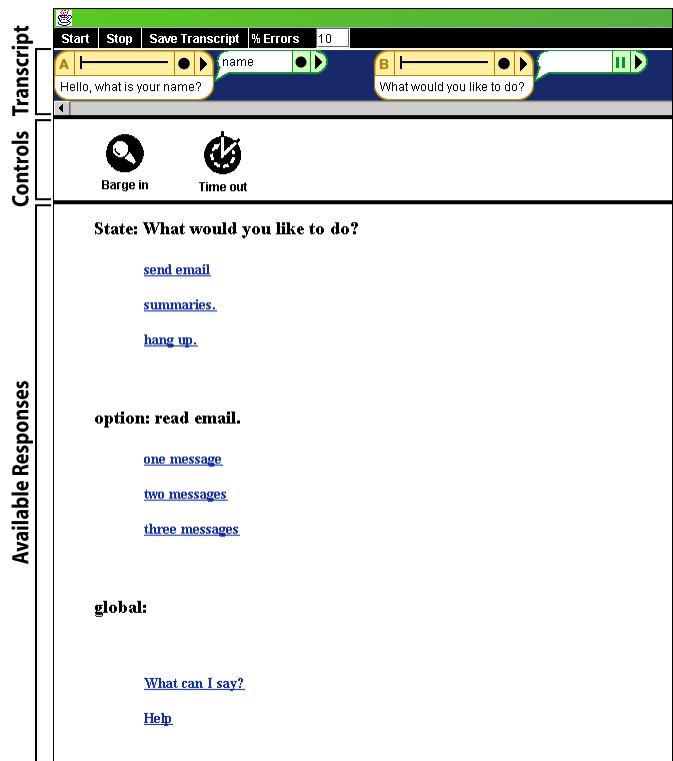


Figure 2. Test mode is presented in a web browser and allows the wizard to focus on the current state of the UI (*top*) and the available responses for that state (*bottom*).

screen (see Figure 1). When the designer switches to Test mode, SUEDE generates an appropriate HTML file for every prompt card in the design graph. The hyperlinks on each page represent the corresponding response links possible from that card. Clicking on a link moves from card to card.

The testing window is a browser-based interface (see Figure 2). The screen is broken up into four distinct sections (from top to bottom): session menu, a transcript of the current session, barge-in and timeout controls, and an HTML page of the valid user responses to the current prompt card. Global responses are available on all pages.

In Test mode, a wizard works in front of a computer screen. The participant performs the test away from the wizard, in a space with speakers to hear the system prompts and a microphone hooked up to the computer to record his responses. When the wizard starts a test session, SUEDE automatically plays the pre-recorded audio from the current prompt card. The wizard waits for the test participant to respond, and then clicks on the appropriate hyperlink based on the response. During the course of the test session, a transcript sequence is generated containing the original system audio output and a recording of the participant's spoken audio input.

The wizard's only job is to click on the appropriate controls and links in the test interface HTML area (see Figure 2, bottom). SUEDE incorporates functionality to automatically insert simulated speech recognition errors, which is further described in the "Error Modeling" section below. The wizard can monitor the progress of the session in the Transcript area at the top of the test interface, which shows the prompts that have been played so far, along with the matched responses.

Continuing our MailCall example, we see in Figure 2 that the test session has just played the "What would you like to do?" A participant might respond, "I'd like to write an email please." The wizard would interpret the user's response and click on the "Send email" link. Also note in Figure 2 the three distinct choices under "Read email," illustrating the test view of SUEDE's group structure. At test time, the wizard can choose any one of the grouped prompts to transition to next.

Since there is no speech recognition system underlying this Wizard of Oz test, in the early stages of design the wizard will use this opportunity to accept several alternative inputs. In our example, the wizard might accept "Send," "I want to send," and "Write email" as valid utterances for the "Send email" response link. These actual audio responses will be recorded and associated with the response link and can be later reviewed in Analysis mode to help determine the input grammar for the final design.

Error Modeling

In the session menu area, there is a parameter marked "% Errors". This value sets the simulated speech recognition error. As the wizard is running the system, SUEDE can

insert random misrecognition errors as a real speech recognizer might do, as described in [31]. If a random error happens, SUEDE overrides the wizard's choice, informs him of this fact, and randomly chooses one of the other possible links on that page. The wizard is not tasked with mimicking an error rate. A representative example of a random error follows:

Prompt: "On what day would you like to fly?"

Response: "Thursday"

Prompt: "The flights on Tuesday are ..."

A typical participant response in this scenario would be:

Response: "No I meant Thursday"

Handling this situation should be part of a robust speech interface design. Recording what participants say in response to errors helps the designer analyze and handle these errors in future design iterations. We plan to extend SUEDE to allow automatic backup to a previous prompt to assist in handling these types of errors.

Timeouts

Many speech interfaces treat the lack of response after a certain time window as a timeout error. A common strategy for a timeout in many interfaces is to repeat the last played prompt, in hope that the participant will be able to respond, in case they did not hear the prompt the first time. Clicking the Timeout button in Test mode executes this behavior.

A more sophisticated timeout handling response is to give cooperative incremental feedback to the participant. In SUEDE, incremental feedback can be modeled with prompt groups and response links.

Barge-In

Barge-in is a fairly sophisticated speech interface technique in which a participant responds to a prompt while the prompt is being played. It is especially important in conversational interfaces where prompts might be long and repetitive. SUEDE allows the wizard to simulate barge-in by stopping the current prompt and switching to recording audio when the Barge-in button is pressed. Because of its manual nature, this button-pressing process might not capture all of the response audio. Many real speech systems automatically recognize barge-in; in the future, we will incorporate this feature to ease the burden on the wizard and more completely capture the participant's response.

Transcript

The entire participant session is recorded in the set of prompt cards and response links in the transcript area of the test interface (see Figure 2, top). This transcript also appears together with those of other test participants in the script area of Analysis mode (see Figure 3, top).

Analysis Mode

Many designers take notes during test sessions. Often they must enlist the help of others to help keep track of statistics during their tests. SUEDE eases the burden of statistics collection by automatically recording audio, automatically

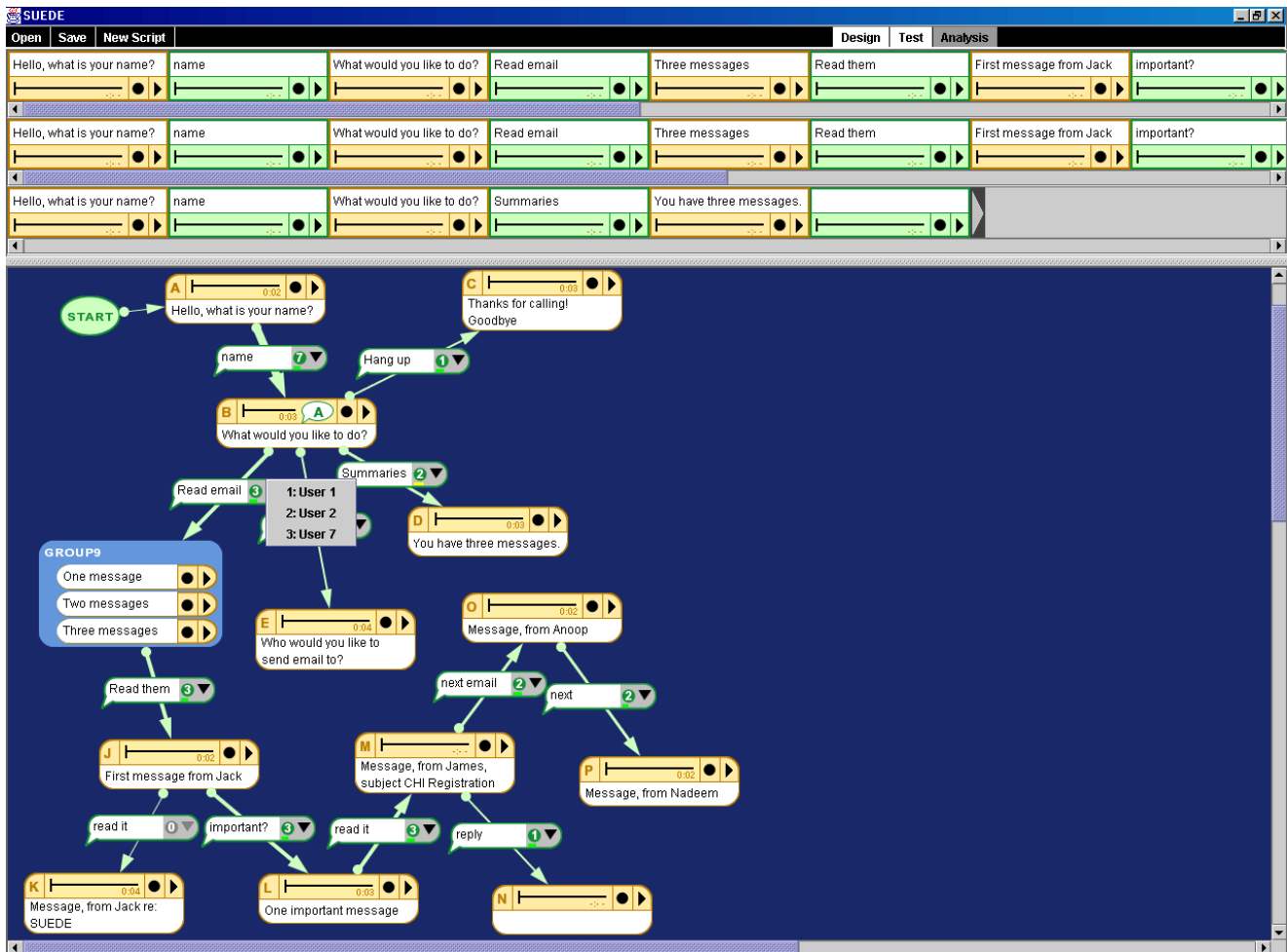


Figure 3. Analysis mode displays transcripts from user test sessions (*top*) as well as an annotated version of the design that summarizes the aggregate test results (*bottom*). The annotated version also provides the ability to hear the set of responses for a particular link.

creating transcripts of events, and providing several means of accessing this data.

Data collected in Test mode is displayed in Analysis mode (see Figure 3). The Analysis interface is similar to the Design interface, except the top of the screen contains *user transcripts* from the test sessions rather than just designer-made examples, and an annotated version of the design graph is displayed in the design area. The annotated design includes information on the number of participants who took a particular path, the actual responses they made, and how long they took to respond to each prompt. There is a pleasing duality between the designer examples and the actual session transcripts.

During testing, statistics are collected about the number of participants who traverse a given path in the design. Switching to Analysis mode displays that statistical information. Response links are scaled in width to show the number of times that link was traversed in the user sessions. Counters are also displayed on each response link to show the number of times that link was traversed. These two visualizations give the designer a feel for what parts of

the design were most used and thus need optimization, and what parts were not used and thus may need more work. Only by collecting and examining data from real users can a designer understand the good and bad features of a design and iteratively improve it.

Continuing our example, we see that three test participants followed the “Read email” link and that one participant followed the “Send email” link (see Figure 3, bottom).

The audio of each user session is also available in the script area so that the designer can review specific responses to a prompt. The Analysis mode also allows the designer to review all of the responses across participants for a specific prompt directly from the design graph node by clicking on the link counter (as illustrated with the “Read email” node in Figure 3). Examining the test transcripts and reviewing individual responses aids in the transition to a formal input grammar for the final interface design.

The Analysis visualization also shows the average time it took participants to respond to each prompt. This is represented by the length of the time bar in the lower right of the response link. For coarse feedback at a glance,

SUEDE presents the time bar in green if the average response time was short, yellow for a medium response time, and red for a long response time.

As mentioned previously, problems with speech interfaces arise when users do not know what to say, when they say invalid words, or when the system makes recognition errors. SUEDE addresses these issues through support in the tool itself. The displayed timing data lets the designer see where participants paused in a test dialog, possibly indicating that the participant did not know what to say. Playback of the transcript allows the designer to hear what participants thought they could say at any point. The designer can manually make text transcriptions of this data by editing the card labels in the transcript area and later use these textual transcriptions to help generate an input grammar. Finally, error simulation allows a designer to see how participants would cope with recognition errors in interface designs. Using this resulting test data, a designer can make appropriate design decisions about the flow of the interface and also the proper input grammar for her design.

Speech Interface Styles

Linguists have shown that human-computer conversation is quite different from human-human conversation [12]. Here, we offer a characterization of current speech interface application styles, based largely on a survey of existing speech systems:

1. Simple prompt and response interfaces: Automated call routing interfaces, such as the one used by United Airlines [42], and larger vocabulary command interfaces like M.I.T.'s VoiceNotes, a portable, voice-controlled system for storing, navigating, and retrieving recorded to-do lists [36], GPSS's interactive, speech-based, car guidance system [25], and Hyperspeech, a system for browsing a collection of hyperlinked audio segments [1].
2. Full sentence conversational interfaces: Speech Acts Calendar [49] and Office Monitor [50] developed at Sun, and Wildfire Personal Assistant, a voice-controlled system that manages contact lists, answers and places calls, and manages voice-mail messages [44].
3. Dictation-orientated applications: Dragon Systems' Dragon Dictate [14] and IBM ViaVoice [19].
4. Speech manipulating interfaces: Speech Skimmer, a system for scanning a collection of speech segments [2], and Storywriter, a speech-based document editing system [13].
5. Multimodal applications: Multimodal maps that use both pen and speech input [9, 28].

We have designed SUEDE to support the first two of these speech interface styles. In its current form, SUEDE is not suited for prototyping applications that have alternative modes of feedback (such as the text in a dictation or the

graphics in a multimodal application) or involve manipulating or editing audio.

IMPLEMENTATION

SUEDE is implemented in Sun's Java JDK1.3, using the JavaSound package for audio, and the Java AWT, Java Swing, and Java2D packages for graphics. We have built a custom set of vector graphics widgets to create the visual representations.

The SUEDE software architecture employs the Model-View-Controller paradigm [39]. SuedeModel manages the scripts, prompts (cards), and responses (links) that are displayed on the screen. Each of those individual items has a corresponding visual view and also a corresponding, reusable audio data structure that supports multiple audio streams.

SUEDE uses XML as its file format, and file I/O is implemented on top of the Java XML Parser (JAXP 1.0.1).

RELATED WORK

SUEDE is inspired by previous work in low-fidelity prototyping and Wizard of Oz studies, as well as by existing systems for prototyping and testing speech and multimodal user interfaces.

Low-fidelity Prototyping

Low-fidelity paper prototyping [34] is a popular design and evaluation technique, used to prototype systems quickly and easily. SUEDE aims to provide the same benefits to speech-based interfaces: rapid prototyping and testing with little expert knowledge necessary.

Ordinary low-fidelity prototyping is fallible because of the task complexity for the wizard, the "human computer," that simulates the prototype. Because SUEDE employs a computational element to perform some operations in concert with the wizard, user test stimuli and interface logic are more likely to be presented correctly between participants. One of the primary goals of our research was to make Test mode as simple as possible so that the wizard can react quickly and accurately to the test participant's responses.

SUEDE also offers a much more manageable visualization of an interface design than that offered by paper or domain independent flowchart tools such as Visio. In addition, SUEDE's designs are stored in a form that, in the future, may allow them to be semi-automatically converted to fully working systems, as was done for sketched GUIs in SILK [22].

Wizard of Oz Studies, Tools, and Toolkits

The Wizard of Oz technique has been used for years to simulate speech recognition systems when performing both low-fidelity tests of proposed design ideas and user studies on "finished" interface designs [12]. In a standard Wizard of Oz study [16, 20], a human simulates the speech system. Participants hear prompts read by the wizard face-to-face or remotely via a computer, intercom, or phone. To decide what prompt to read, the wizard follows a script or

flowchart based on the participants' responses. SUEDE's electronic support for WOz testing improves on traditional WOz by making it easy to carry out repeatable tests as well as keep track of what happens during the test sessions. A designer can easily capture and review evaluation statistics on specific parts of an interface.

Yankelovich made frequent use of "pre-design studies" in her work on Speech Acts [47, 48, 50]. These studies involve observing natural dialogues between people in the target setting, as well as performing WOz-like simulations of the speech system, as in the design of the Office Monitor [50]. These pre-design studies are an important component of speech interface design, and one of our goals was to make it easier for designers to carry them out. SUEDE's session recording and analysis features make the data generated from these studies easily accessible and even more valuable.

The NEIMO system [3, 11] is a platform for the study of multimodal systems, and SRI's Open Agent Architecture [7] is a toolkit for implementing multimodal applications. Both attempt to improve the difficult task of designing multimodal user interfaces by using WOz techniques. These systems require functioning software to be written before testing can begin. In contrast, SUEDE is oriented at early stage speech UI design and thus has no such software requirement. The freedom from requiring completed software makes creating interfaces in SUEDE more accessible to designers, who are typically non-programmers.

The SUEDE WOz methodology of performing no automated speech recognition offers the advantage that designers do not have to worry at this early stage about whether the participants have different accents [30] or genders [43]; it does not matter what language they are speaking at all. This makes the WOz process especially appealing for non-English UIs, where current recognizers generally perform worse than for English [32].

Speech-based UI Construction Tools

There are two existing speech UI construction tools which are similar to SUEDE in several respects: the CSLU Rapid Application Developer (RAD) [37, 38] and Unisys' Natural Language Speech Assistant (NLSA) [41].

Both CSLU RAD and NLSA combine speech recognition, speech synthesis, and the Wizard of Oz technique into an integrated tool for building speech applications. Like SUEDE, these tools use a visual, state machine representation of the speech UI [27]. CSLU RAD and NLSA are oriented towards specifying, testing, and implementing a more finished application interface. One could imagine a designer first prototyping and testing in SUEDE and then transferring a concrete design idea to CSLU or NLSA where she would add the details to create and test the final implementation.

Although NLSA and possibly CSLU, given some code additions, could be used in similar ways, SUEDE's

informal user interface makes it more appropriate for early phase design. Our earlier work on GUI design tools showed that letting designers ignore details, such as fonts, colors, and alignment, enabled them to focus on the interaction design [22]. With speech-based user interfaces, the need to adjust recognition parameters is even more tempting.

Although NLSA, like SUEDE, has a WOz mode that does not use recognition, this part of the tool offers no support for creating synthetic errors during WOz studies as in SUEDE and the work of Oviatt, et. al. [31]. In addition, neither NLSA nor CSLU RAD offers tools for analyzing the test data. For instance, SUEDE will record that the undefined "yeah" utterance was associated with the "yes" transition. This lets designers know what things users actually say.

A common limiting factor of all three tools is that because of the state-based metaphor, they are most appropriate for prompt/response interfaces.

FUTURE WORK

We have released SUEDE to the speech design community to further evaluate the scenarios most appropriate for its use. (See <http://guir.berkeley.edu/suede/>)

We have discussed SUEDE with several professional speech UI designers. One common interest has been a way to migrate SUEDE interfaces to the development environments of various speech recognition systems. We plan to extend SUEDE's Analysis tools to support grammar creation for a standard speech recognition system. Also, because SUEDE is open source, interested parties can add additional modeling in Test mode that might reflect the characteristics of their own systems.

SUEDE's supports the early stage of speech interface design. As an informal tool, SUEDE offers significant flexibility for designers. We will be adding more sophisticated support for speech design features such as tapering, error handling, and cooperative prompts, though it is possible to model these right now. Another logical extension is to allow SUEDE designs to function as reusable components, to be used in higher-level designs. We will further extend Test mode to collect additional user session data and wizard annotations. And Analysis mode will use this information to help the designer evolve his or her dialog design.

SUEDE's ability to save designs to disk in an XML format provides a primitive method of design versioning. In the future, we will develop a more sophisticated versioning strategy through which a designer can compare past designs with current designs.

CONCLUSIONS

The speech interface design problem is complicated; one cannot know in advance what users will say to a speech system. A high quality speech user interface can only be developed through iterative design and evaluation. SUEDE makes significant progress on support for the early stages of this process. Based on our interviews, SUEDE's Design,

Test, and Analysis paradigm maps quite well onto the speech designer's mental process.

Many designers use scripts as their initial concrete examples. SUEDE supports this work process. The script facilitates designer reflection about what it is they are building, and the dualism between script and transcript helps close the iterative design loop.

The high level of frustration associated with speech interfaces in their current incarnation may prevent them from ever becoming preferred by customers [17]. The problem here, we believe, is not one of medium, but one of design—design the speech interface well, and users will come to value the system.

ACKNOWLEDGEMENTS

We would like to thank Cindy Chen for her help with developing the software, and David Breslauer for testing SUEDE and creating a software download with an online manual. We greatly appreciate Jason Hong and James Lin's sage Java advice and comments on drafts of this paper. We also thank the designers we spoke with at Nuance, SRI, and Sun for the insight they offered into the speech design process. Finally, we thank the reviewers of this paper for their valuable feedback and insight.

REFERENCES

1. Arons, B., Hyperspeech: Navigating in Speech-only Hypermedia, in *Proceedings of ACM Hypertext '91*. p. 133-146, 1991.
2. Arons, B., SpeechSkimmer: A System for Interactively Skimming Recorded Speech. *ACM Transactions on Computer-Human Interaction*, 1997. **4**(1): p. 3-38.
3. Balbo, S., J. Coutaz, and D. Salber, Towards Automatic Evaluation of Multimodal User Interfaces, in *Proceedings of the International Workshop on Intelligent User Interfaces*. p. 201-208, 1993.
4. Black, A., Visible Planning on Paper and on Screen: The Impact of Working Medium on Decision-making by Novice Graphic Designers. *Behaviour & Information Technology*, 1990. **9**(4): p. 283-296.
5. Boyarski, D. and R. Buchanan, Computers and Communication Design: Exploring the Rhetoric of HCI. *Interactions*, 1994. **1**(2): p. 24-35.
6. Carroll, J.M., *Scenario-Based Design: Envisioning Work and Technology in System Development*: John Wiley & Sons. 408, 1995.
7. Cheyer, A., L. Julia, and J.C. Martin, A Unified Framework for Constructing Multimodal Experiments and Applications, in *Proceedings of CMC '98*: Tilburg, The Netherlands. p. 63-69, 1998.
8. Clarke, L. The Use of Scenarios by User Interface Designers. In *Proceedings of The HCI'91 Conference on People and Computers VI*. pp. 103-115, 1991.
9. Cohen, P.R., M. Johnston, D. McGee, S.L. Oviatt, J. Clow, and I. Smith. The efficiency of multimodal interaction: a case study. In *Proceedings of The International Conference on Spoken Language*, 1998.
10. Cohen, P.R. and S.L. Oviatt, The role of voice input for human-machine communication. *Proceedings of the National Academy of Sciences*, 1995. **92**(22): p. 9921-9927.
11. Coutaz, J., D. Salber, E. Carraux, and N. Portolan, NEIMO, a Multiworkstation Usability Lab for Observing and Analyzing Multimodal Interaction, in *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*. p. 402-403, 1996.
12. Dahlbäck, N., A. Jönsson, and L. Ahrenberg. Wizard of Oz Studies - Why and How. In *Proceedings of Intelligent User Interfaces '93*. pp. 193-200 1993.
13. Danis, C., L. Comerford, E. Janke, K. Davies, J. DeVries, and A. Bertrand, Storywriter: A Speech Oriented Editor, in *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*. p. 277-278, 1994.
14. Dragon, *Dragon Dictate*, 2000. Dragon Systems. <http://www.dragonsys.com/>
15. Goel, V., *Sketches of Thought*. Cambridge, MA: The MIT Press. 279, 1995.
16. Gould, J.D. and C. Lewis, Designing for Usability -- Key Principles and What Designers Think, in *Proceedings of ACM CHI'83 Conference on Human Factors in Computing Systems*. p. 50-53, 1983.
17. Gwyther, M., Voicemail Hell, *Management Today* pp. 76-77, 1999.
18. Hearst, M.A., M.D. Gross, J.A. Landay, and T.E. Stahovich, Sketching Intelligent Systems. *IEEE Intelligent Systems*, 1998. **13**(3): p. 10-19.
19. IBM, *ViaVoice*, 2000. IBM. <http://www.ibm.com/software/speech/>
20. Kelley, J.F., An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications. *ACM Transactions on Office Information Systems*, 1984. **2**(1): p. 26-41.
21. Landay, J.A., *Interactive Sketching for the Early Stages of User Interface Design*, Unpublished Ph.D., Carnegie Mellon University, Pittsburgh, PA, 1996. <http://www.cs.berkeley.edu/~landay/research/publications/Thesis.pdf>
22. Landay, J.A. and B.A. Myers. Interactive Sketching for the Early Stages of User Interface Design. In *Proceedings of Human Factors in Computing Systems: CHI '95*. Denver, CO. pp. 43-50, May 7-11 1995.
23. Landay, J.A. and B.A. Myers. Sketching Storyboards to Illustrate Interface Behavior. In *Proceedings of Human Factors in Computing Systems: CHI '96 Conference Companion*. Vancouver, Canada. pp. 193-194, April 13-18 1996.

24. Lin, J., M.W. Newman, J.I. Hong, and J.A. Landay, DENIM: Finding a tighter fit between tools and practice for web site design. *CHI Letters: Human Factors in Computing Systems, CHI '2000*, 2000. **2**(1): p. 510-517.
25. Lovelock, R., *GPSS's speech interface*, 1999. Sunninghill Systems. <http://www.gpss.co.uk>
26. Marx, M. and C. Schmandt. MailCall: Message Presentation and Navigation in a Nonvisual Environment. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*. pp. 165-172, 1996.
27. McTear, M.F. Modelling Spoken Dialogues with State Transition Diagrams: Experiences with the CSLU Toolkit. In *Proceedings of 5th International Conference on Spoken Language Processing (ICSLP '98)*. Sydney, Australia, Dec 1998.
28. Moran, L.B., A.J. Cheyer, L.E. Julia, D.L. Martin, and S. Park, Multimodal user interfaces in the open agent architecture. *Knowledge-Based Systems*, 1998. **10**(5): p. 295-304.
29. National Research Council, *More than screen deep: toward every-citizen interfaces to the nation's information infrastructure*. Washington, D.C.: National Academy Press. 433, 1997.
30. Oviatt, S. Mutual Disambiguation of Recognition Errors in a Multimodal Architecture. In *Proceedings of ACM CHI 99 Conference on Human Factors in Computing Systems*. pp. 576-583, 1999.
31. Oviatt, S., P. Cohen, M. Fong, and M. Frank. A rapid semi-automatic simulation technique for investigating interactive speech and handwriting. In *Proceedings of The International Conference on Spoken Language Processing*. Banff, Canada, October 1992.
32. Pallett, D.S., J.G. Fiscus, J.S. Garofolo, A. Martin, and M. Przybocki. 1998 Broadcast News Benchmark Test Results: English and Non-English Word Error Rate Performance Measures. In *Proceedings of 1999 DARPA Broadcast News Workshop*, 1998.
33. Pearl, C., Personal Communication, 2000.
34. Rettig, M., Prototyping for Tiny Fingers. *Communications of the ACM*, 1994. **37**(4): p. 21-27.
35. Sidner, C., Creating Interfaces Founded on Principles of Discourse Communication and Collaboration, in *More than screen deep: toward every-citizen interfaces to the nation's information infrastructure*, National Research. Council, Editor. National Academy Press: Washington, D.C. p. 315-321, 1997.
36. Stifelman, L.J., B. Arons, C. Schmandt, and E.A. Hulteen, VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker, in *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems*. p. 179-186, 1993.
37. Sutton, S. and R. Cole, The CSLU Toolkit: rapid prototyping of spoken language systems, in *Proceedings of UIST '97: the ACM Symposium on User Interface Software and Technology*. p. 85-6, 1997.
38. Sutton, S., D.G. Novick, *et al.*, Building 10,000 spoken dialogue systems, in *Proceedings ICSLP 96: International Conference on Spoken Language Processing*, H.T. Bunnell and W. Idsardi, Editors. p. 709-12 vol.2, 1996.
39. Tesler, L., The Smalltalk Environment, *Byte Magazine*, vol. 6(8): pp. 90-147, 1981.
40. U.S. Department of Education, *1992 National Adult Literacy Survey*. Washington, DC: U.S. Government Printing Office, 1992.
41. Unisys, *Natural Language Speech Assistant*, 1999. Unisys Corp. <http://www.unisys.com/marketplace/nlu/nlaproductinfo.html>
42. United, *United Airlines Customer Service*, 1999. United Airlines: 1-800-241-6522.
43. Vergin, R., A. Farhat, and D. O'Shaughnessy, Robust gender-dependent acoustic-phonetic modelling in continuous speech recognition based on a new automatic male/female classification, in *Proceedings ICSLP 96. Fourth International Conference on Spoken Language Processing*, H.T. Bunnell and W. Idsardi, Editors: Philadelphia, PA, USA. p. 1081-4, 1996.
44. Virtuosity, *Wildfire Personal Assistant*, 1999. Virtuosity. <http://www.wildfire.com/>
45. Wagner, A., Prototyping: A Day in the Life of an Interface Designer, in *The Art of Human-Computer Interface Design*, B. Laurel, Editor. Addison-Wesley: Reading, MA. p. 79-84, 1990.
46. Weiser, M., Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 1993. **36**(7): p. 74-84.
47. Yankelovich, N., Talking vs Taking: Speech Access to Remote Computers, in *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*. p. 275-276, 1994.
48. Yankelovich, N. and J. Lai, Designing Speech User Interfaces, in *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems (Summary)*. p. 131-132, 1998.
49. Yankelovich, N., G.-A. Levow, and M. Marx, Designing SpeechActs: Issues in Speech User Interfaces, in *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*. p. 369-376, 1995.
50. Yankelovich, N. and C.D. McLain, Office Monitor, in *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*. p. 173-174, 1996.