INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 5, 159–169, 2002

Embarking on Spoken-Language NL Interface Design

ANOOP K. SINHA, SCOTT R. KLEMMER AND JAMES A. LANDAY
Group for User Interface Research, CS Division, EECS Department, University of California at Berkeley, Berkeley, CA 94720-1776, USA
aks@cs.berkeley.edu
http://guir.berkeley.edu

Abstract. Natural language (NL) user interfaces are growing in popularity. Unfortunately, the complexity of NL interaction makes these interfaces difficult to design. For NL interfaces to become successful, universal tools are needed to help support the NL design process. What work practice should these tools explicitly support? Interviews with NL designers and our own experiments have identified a specific work practice that designers should consider as they begin to incorporate NL into user interface designs. The work practice study highlights the value of using Wizard of Oz prototyping in NL design. We describe a tool that we have built, called SUEDE, to explicitly support the first stage of NL design for spoken-language user interfaces. Our tools and tools like it will help make NL in human-computer interaction (HCI) more commonplace.

Keywords: speech user interfaces, design tools, informal user interfaces

Introduction

Natural language (NL) techniques have much to contribute HCI. The HCI community strives to continuously find ways to make user interfaces more "natural," "invisible," and "ubiquitous" (Weiser, 1993). The tool used for human-human communication, natural language, has all three characteristics and many more.

For NL interfaces to become successful, universal tools are needed to help support the NL design process (Sutton et al., 1998). At present, spoken-language NL user interfaces are hard to design 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 spoken user interfaces that are far from natural.

Interviews with NL designers and our own experiments have identified a specific work practice that designers should consider as they begin to incorporate NL into user interface designs. The work practice study highlights the already identified value of using Wizard of Oz prototyping in NL design (Dahlbäck et al., 1993).

We have created a Wizard-of-Oz prototyping tool, called SUEDE, to explicitly support this NL design work practice (Klemmer et al., 2000). SUEDE and tools like it will help make NL in HCI more commonplace.

Why NL User Interfaces?

Today's graphical user interfaces (GUIs) do not let users communicate in ways that they naturally do with other human beings (Sidner, 1997). Furthermore, standard GUIs do not work well for users with poor vision or limited use of their hands (5% of those over age 15 (National Research Council, 1997)). When users are moving around, using their hands or eyes for something else, or interacting with another person, they need better interface paradigms. Spoken-language NL user interfaces are one paradigm that can successfully address many of the aforementioned problems.

Supporting NL Designers

A key, limiting factor in spoken-language NL design is the lack of basic knowledge about user "performance during computer-based spoken interaction" (Cohen and Oviatt, 1995). To remedy this problem, an NL designer needs to be able to do quick experiments with design tools that support prototyping spoken interfaces. NL designers also need guidelines to understand how users want to interact with the computer, because that has been determined to be different than how they interact with other humans (Dahlbäck et al., 1993).

Many interaction designers who could contribute to NL design are excluded by the complexities of the core technologies and the formal representations used for specifying these technologies. The complex recognizer, synthesizer, and NL technologies inherent in a spoken-language NL interface require a high level of technical competency to understand. Specifically, the grammar and state-machine representations used to design speech-based systems are formal and abstract. This is in awkward contrast with informal, concrete representations, such as scenarios (Clarke, 1991; Carroll, 1995), sketches (Boyarski and Buchanan, 1994; Landay and Myers, 1995), and storyboards (Wagner, 1990; Landay and Myers, 1996; Lin et al., 2000), which designers commonly employ to explore design ideas through examples.

This basic mismatch in approach between user interface design and NL implementation is an important issue that must be resolved for NL in HCI to become more successful. An *informal interface* approach, using unrecognized, natural input (e.g., sketching or speech) successfully supports designers in the early stages of design (Landay and Myers, 2001). This approach enables a designer to plan an NL interface before any code has been written. Using informal tools, designers can be encouraged to explore the design space leading to better NL interface designs.

What do Users Want to Say?

For a given application, "What do users want to say?" is the first question that a spoken-language NL designer needs to ask. To gain insight into this question, we explored the nature of the dialog between a human and a computer in a simulated, voice-controlled e-mail application. In our experiment, we placed eight participants, one at a time, in front of a computer screen with a graphical e-mail application. We asked each participant to perform various tasks in the application using voice, and we remotely controlled the application using remote control software. We explicitly let the participants know that the recognition was being performed by a remotely located human, and told them they could say whatever was most natural for them, assuming full language recognition.

All eight participants in our tests spoke commands in what we could call "computer-speak," a simple step-by-step description, born out of the mental model of how they perform the task traditionally on the computer with keyboard and mouse. For sending an e-mail message, these were a typical participant's spoken steps:

1. "*Create new message*"
2. "*Address it to John*"
3. "*Make subject hello*"
4. "*Make message how are you doing*"

This step-by-step interaction was in spite of the fact that the human backend would have parsed and understood the more natural phrase, "send a message to john saying hello, how are you doing?"

From our follow-up interviews with the participants afterwards, we learned that the participants viewed the computer as passive in the dialog, without inference ability. The users figured out that the way to make the computer accomplish the tasks is to instruct directly it, step-by-step.

From that experiment, we formalized a set of recommendations for picking speech commands, as follows:

1. *Use descriptions of direct manipulation tasks as the guide for the commands*
2. *Pick commands at the levels of actions*
3. *Perform iterative tests to further finalize the choices of NL commands*

This underlying philosophy has helped us design speech commands for a wide variety of speech-activated applications. We also found this method of choosing initial NL commands in our interviews with NL interface designers.

NL Design Work Practice

To profile existing NL design work practice, we interviewed six spoken-language NL designers from both industrial research (SRI and Sun Microsystems Laboratories) and development organizations (Nuance Communications and Sun Microsystems). We asked them details about their work practice, including what specific artifacts and tools they use in their design process. Typical applications on which they have worked include stock lookup, airline flight information, and

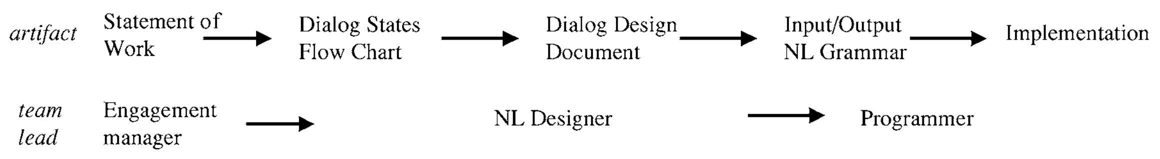


Figure 1. Timeline for a spoken-language NL interface implementation.

universal e-mail. As a case example, we will briefly outline the typical work process for one of the spoken-language NL interface designers that we interviewed.

The NL designer's main role is dialog specification: What will the dialog look like in the final application? He/She is in charge of all aspects of the dialog design, from brainstorming to pre-testing to iteration. Ultimately he/she creates a dialog design specification, which is then translated into the NL input/output grammar and used by the programmer in implementation (Fig. 1).

Statement of Work

New projects begin with a *statement of work*, a list of the requests by the client company. The statement of work outlines the desired business requirements, the amount of time it should take, and agreements on cost. The statement of work is referred to throughout the design process, but is quickly left as the attention shifts towards dialog design.

Example Dialogs

With the statement of work delivered, the designer begins to create *example dialogs* at the outset of the dialog design process. Example dialogs are linear sequences that represent the interaction of a typical user in the final application. As might be expected, these dialog examples include only partial functionality of the final system. Nevertheless, the examples give structure for various types of tasks. For instance, the name of the company in a stock look-up application is a parameter in the final application.

Dialog States Flowchart

These dialog examples are soon structured into *dialog states flowcharts*, outlines of the application functionality and the flow among different prompts and responses (Fig. 2). The dialog states flowcharts are created either using paper 3×5 cards or, more often, in a flowcharting program such as Microsoft Visio. The flowcharts are

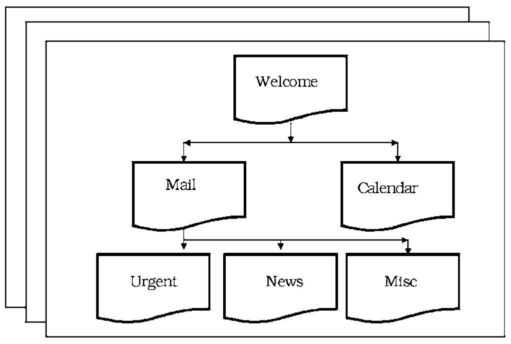


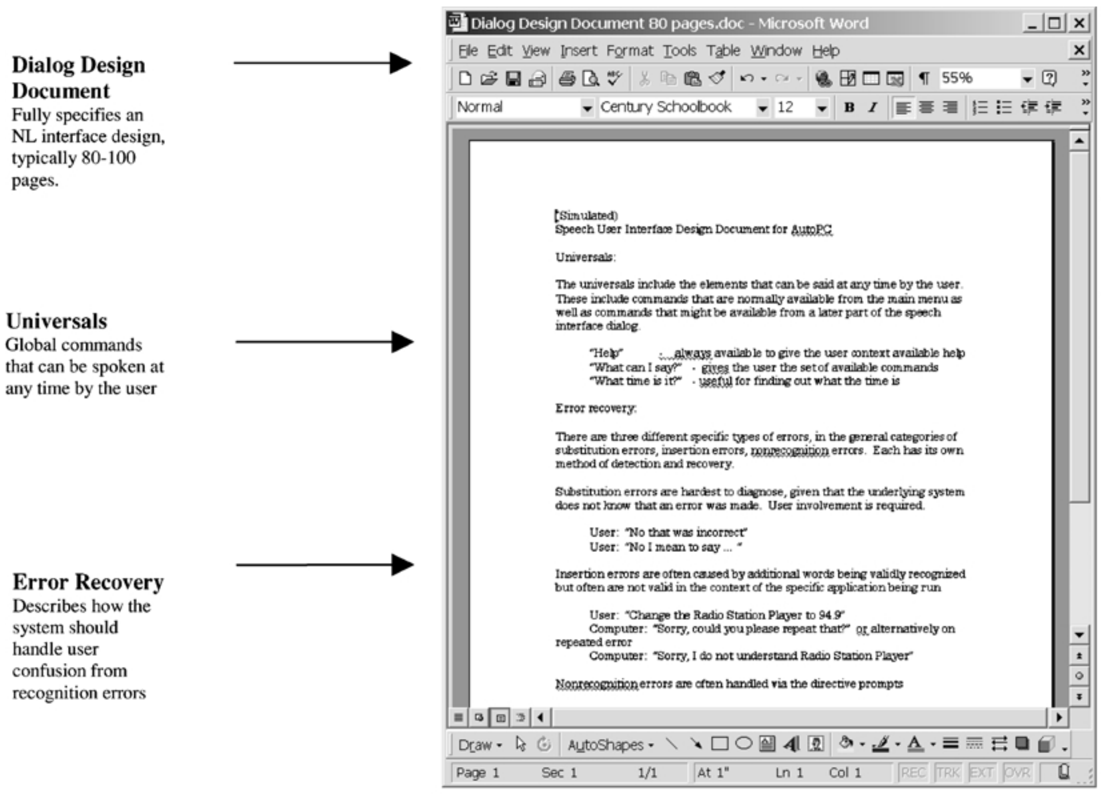
Figure 2. A dialog states diagram outlines the interface functionality.

hierarchical, with common functionality separated into subcomponents, sometimes from existing libraries, on different pages. A single page typically has 10 to 12 states on it. A typical application is outlined in about 100 states or 10 pages.

The interaction specified by the flowcharts is difficult to comprehend fully without testing. Testing typically involves laborious programming to run through sample scenarios. Even at this early stage, the dialog designer needs to involve the help of a programmer to implement the test, unless he/she has sufficient programming ability with the target speech recognizers.

One designer we interviewed likes to use Wizard-of-Oz testing instead. The Wizard-of-Oz process involves simulating computer interaction by speaking prompts aloud. User sessions sometimes are over the telephone, which is a typical NL interface interaction mode for her designs, and are captured on analog tape recorders. Changing the design by reordering the 3×5 cards is straightforward. However, reviewing the analog tapes for critical incidents is very time intensive. Another challenge in Wizard-of-Oz testing is simulating errors and error handling. Methods for simulating errors are *ad hoc*.

Due to the manual labor involved, Wizard-of-Oz testing is neither popular nor easy. There is no specific computer tool support available for Wizard-of-Oz tests. All our interviewees agreed that an easy-to-use Wizard-of-Oz tool is something that they want to have.


Dialog Design Document
Fully specifies an NL interface design, typically 80-100 pages.
Universals
Global commands that can be spoken at any time by the user
Error Recovery
Describes how the system should handle user confusion from recognition errors
Dialog Design Document 80 pages.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Normal
Century Schoolbook
12
55%
Draw
AutoShapes
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

(Simulated)
Speech User Interface Design Document for AutoPC

Universals:

The universals include the elements that can be said at any time by the user. These include commands that are normally available from the main menu as well as commands that might be available from a later part of the speech interface dialog.

"Help" - always available to give the user context available help
"What can I say?" - gives the user the set of available commands
"What time is it?" - useful for finding out what the time is

Error recovery:

There are three different specific types of errors, in the general categories of substitution errors, insertion errors, nonrecognition errors. Each has its own method of detection and recovery.

Substitution errors are hardest to diagnose, given that the underlying system does not know that an error was made. User involvement is required.

User: "No that was incorrect"
User: "No I mean to say ... "

Insertion errors are often caused by additional words being validly recognized but often are not valid in the context of the specific application being run

User: "Change the Radio Station Player to 94.9"
Computer: "Sorry, could you please repeat that?" or alternatively on repeated error
Computer: "Sorry, I do not understand Radio Station Player"

Nonrecognition errors are often handled via the directive prompts

Figure 3. A dialog design document for a spoken-language NL interface specifies all details of the interface, including the dialog states, error handling, and globals. It is provided from the NL designer to the programmer performing the implementation once a design has been finalized.

Dialog Design Document

Testing and analysis typically lead to refinements in the design and expansion of the input and output grammars. Ultimately these are detailed in the *dialog design document* (Fig. 3). The dialog design document is the text specification, written from an initial outline, which will be provided to the programmers for implementing the final design. It includes detailed descriptions of all of the states in the dialog states flowchart. It also includes specifications about universals or globals, such as "help" and "main menu." These globals are commands that can be spoken at any time by the user. Specific details about transitions from state to state, including details of all of the error handling cases, points where the users might be in a confused state due to a recognition error, are included in the document.

This document serves as the hand-off document from the dialog designers to the programmers of the final system. The programmers begin work on the final application once this document is created. Changes to the design are difficult and expensive to make once the dialog design document is produced and implementation begins.

Implementation

Changes to the input and output grammars are sometimes necessary as further data are collected from tests of the final application. These changes are much less painful if the overall dialog states structure remains the same. The designers that we interviewed emphasized the importance of creating a good design before the programmer begins implementation.

An Informal Prototyping Tool for Spoken-Language NL User Interfaces

Based on our insights about the nature of spoken-language NL dialog and our study of NL designer work practice, we designed and built a tool, called SUEDE, to support the NL interface design process (Klemmer et al., 2000). SUEDE is designed as an informal prototyping tool, one that is used to map out spoken-language NL interaction quickly and test that interaction.

SUEDE uses a Wizard-of-Oz approach for testing designs, which has a long tradition in the design of spoken-language NL systems (Gould and Lewis, 1983). Problems with NL interaction arise when users do not know what to say, when they say invalid words, or when the system makes recognition errors. SUEDE addresses testing these issues through computer support.

Supported Speech Interface Styles

SUEDE supports simple prompt and response interfaces and conversational interfaces as speech interface styles. These are the style of interface that is most common in telephone interfaces for call centers, airline look-up applications, and speech-enabled e-mail systems, which are common application styles according to the designers that we interviewed. SUEDE does not have any explicitly designed support for mixed-initiative style NL applications, though the designer's designation of global speech commands, described later in this section, does provide a way for a user to jump to a different part of the dialog flow or for the Wizard to move that user to a different part of the flow.

Based on the visual layout, SUEDE supports an application of about 100 prompts at a time, which is the size of a typical prototype among the designers with whom we talked. For a larger, more sophisticated application, SUEDE could be used to design one part at a time, but multiple different parts are not automatically linked. SUEDE is not geared towards dictation-oriented interfaces or speech manipulation applications.

Design/Test/Analysis Methodology

Tufte (1999) argued that usability testing, as practiced today, does not work because it entails repeated cycles of Design and Test, resulting in a user-interface popularity contest. He argued instead that good design was a process of repeated application of Design and Analysis. In our work practice study, we found all three steps in use, Design, Test, and Analysis. Therefore, we used the Design/Test/Analysis methodology as an explicit model for SUEDE.

In the SUEDE *Design* phase, we allow NL interface designers to create linear conversation example scripts (see Fig. 4, top). These examples provide the foundation for the dialog flow specification (shown in progress in Fig. 4, bottom). In the *Test* phase, the designer can perform Wizard-of-Oz testing with participants (see Fig. 5). During *Analysis*, designers examine collected test data, deciding on the speech input and output grammar (see Fig. 6).

Design Mode

At the start of the design process, SUEDE allows linear dialog examples to be created horizontally in the top area, called the *script area*, of Design mode (see Fig. 4, top). 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 records speech on both types of cards in her own voice, and can type a label on each card. By playing the recordings from left to right, the designer can both hear and see the example interaction.

We can see from the second script at the top of Fig. 4 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 Scott," and "*Anything important*?".

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 Fig. 4, bottom). This is analogous to the dialog-states flowcharts in our study. Designs are created either by dragging cards from the script onto the design area or creating new cards on the design area. Cards in the design are linked together into the dialog flow.

Globals, Groups, and Voice Balloons. SUEDE also supports commonly used spoken-language NL interface design elements: globals, groups, and voice balloons.

A *global* is a speech command that can be spoken at any time, such as "main menu" or "help." This is represented by a response link starting off of the background and linked to the global prompt.

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

Prompt: "What would you like to do?"
Response: "Read messages"
Prompt Group:
"You have one message"
"You have two messages"
"You have three messages"

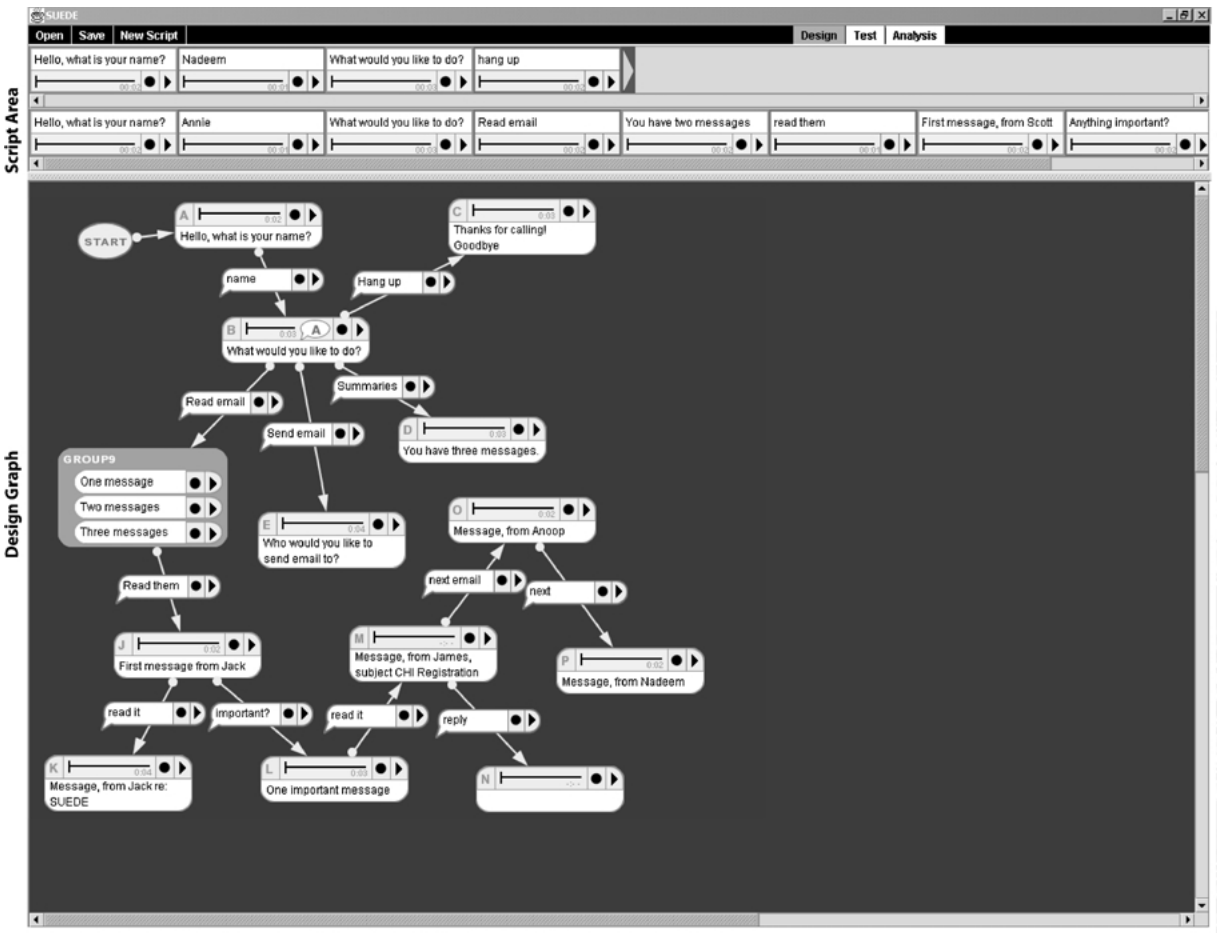

SUEDE
Open
Save
New Script
Design
Test
Analysis
Script Area
Hello, what is your name?
Nadeem
What would you like to do?
hang up
Hello, what is your name?
Annie
What would you like to do?
Read email
You have two messages
read them
First message, from Scott
Anything important?
Design Graph
START
A Hello, what is your name?
name
C Thanks for calling! Goodbye
Hang up
B What would you like to do?
Read email
Summaries
Send email
D You have three messages.
GROUP9
One message
Two messages
Three messages
E Who would you like to send email to?
O Message, from Anoop
next email
next
Read them
J First message from Jack
M Message, from James, subject CHI Registration
P Message, from Nadeem
read it
important?
read it
reply
K Message, from Jack re: SUEDE
L One important message
N

Figure 4. SUEDE's Design mode enables the easy creation of example scripts (top) and speech UI designs (bottom).

When testing, the wizard has the option of choosing among any of these replies to the participant. Groups can also be used for tapering (Yankelovich et al., 1995) or directive prompts (Kamm, 1995), in which different prompts are used based on the number of times the prompt has been played previously. Groups can also be used to give different error messages in different scenarios.

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 spliced automatically into a later prompt. An example of this would be:

Prompt: "What is your name?"
Response: "John Doe"
Prompt: "What would you like to do <*John Doe*>?"

These special features of design mode support the basic details of what designers do during early stage design, up to creating the dialog states diagram.

Test Mode

Informal, unrecognized SUEDE NL interface designs can be executed immediately by clicking on the "Test" button. No speech recognition or speech synthesis is used to test prototypes in SUEDE.

The testing window is a browser-based interface for the wizard performing the testing (see Fig. 5). The screen is divided into four distinct sections (from top to bottom): the session menu, a transcript of the current session, barge-in and time-out controls, and an HTML page of the valid user responses to the current prompt card. Global responses are available on all pages.

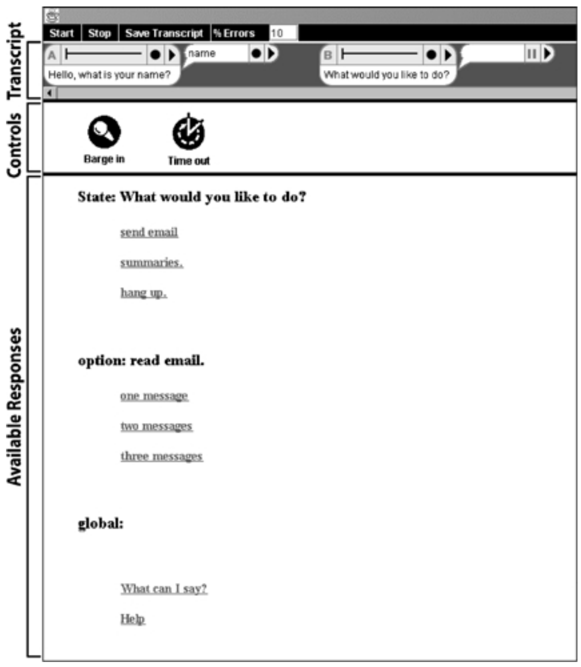


Figure 5. Test mode is presented in a browser and allows the Wizard to focus on the current state of the UI (top) and the available responses for that state (bottom).

In Test mode, a wizard works in front of a computer screen. The participant performs the test in a space where there are speakers to hear the system prompts and a microphone hooked up to the computer to record his/her responses.

SUEDE plays automatically the pre-recorded audio from the current prompt card and records audio from the responses. 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 automatically containing the original system audio output and a recording of the participant's spoken audio input.

Continuing our example, we see in Fig. 5 that the test session has just played "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. Since there is no speech recognition system underlying this Wizard-of-Oz test, the wizard will use this opportunity to accept several alternative inputs. These actual audio responses can later be 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 rate. As the wizard is running the system, SUEDE can insert random misrecognition errors as a real speech recognizer might do, as described in Oviatt et al. (1992). 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. Handling errors should be part of any robust NL interface design.

Time-Outs. A common strategy for a lack of response after a certain time window, a time-out, is to repeat the last played prompt in case the participant did not hear the prompt the first time. Clicking the "Time-out" button in Test mode executes this behavior.

Barge-In. Barge-in, in which a participant responds to a prompt while the prompt is being played, is especially important in conversational NL 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.

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 Fig. 5, top). This transcript also appears together with those of other test participants in the script area of Analysis mode (see Fig. 6, top). There is a pleasing duality between the designer examples and the actual session transcripts.

Analysis Mode

SUEDE eases the burden of statistics collection by recording audio automatically, creating transcripts of events automatically, and providing several means of accessing this data. Data collected in Test mode is displayed in Analysis mode (see Fig. 6). 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.

In Analysis mode, 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

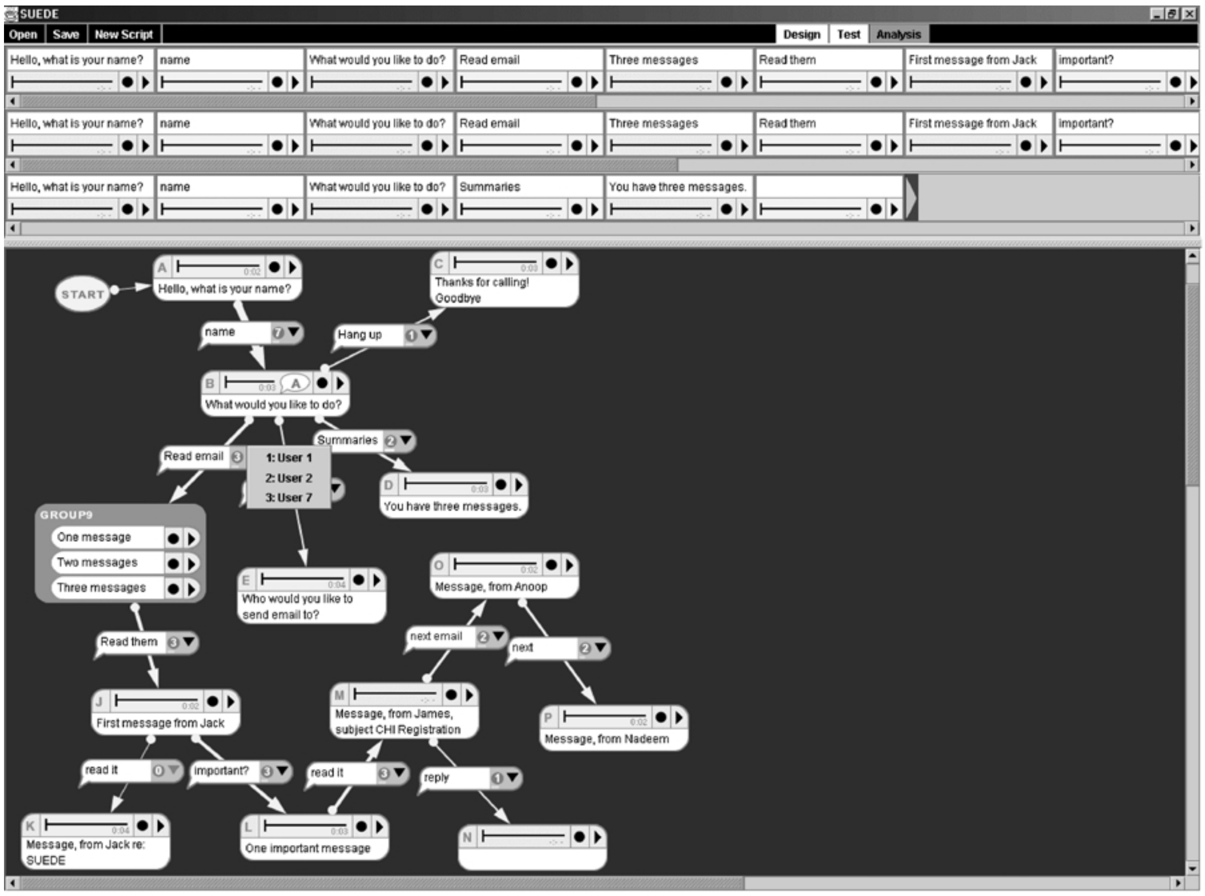


Figure 6. Analysis mode displays transcripts from user test sessions (top) as well as an annotated version of the design that summarizes the aggregate test results (bottom). The annotated version also provides the ability to hear the set of responses for a particular link.

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, one participant followed the "Send email" link, and two followed the "Summaries" link (see Fig. 6, bottom).

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 Fig. 6). Examining the test transcripts and reviewing individual responses facilitates creating a formal input grammar for the final interface design.

User Testing

Though we have not run any formal usability tests, we have provided SUEDE to the NL design community on the public Web since mid-2000 and have had over 500 downloads (See *http://guir.berkeley.edu/suede/*). We have seen some use from major mobile telcommunications organizations, such as Nokia, Ericsson, and Qualcomm, and speech interface organizations such as Nuance and TellMe. Each of these organizations commented to us that they have realized the importance of reaching a good design before any programming and that SUEDE's focus on building and testing an NL interface quickly helps them achieve this goal. In their early stage testing, experienced designers from Nuance and TellMe have explicitly wanted to use Wizard-of-Oz testing, rather than real speech recognition. They believe that the Wizard-of-Oz testing helps them focus on the NL design issues, rather than the recognition technology issues, and helps them reach a good design before worrying about the limitations of the speech recognizer.

Bridging to Final Implementation

For the project phase when a design is almost complete, many SUEDE users have asked for bridges to final tools, such as specific speech recognizers or development environments. We wrote a utility that can take a SUEDE interface and generate a speech user interface grammar, but given our focus on the early stage of design, we have not implemented other bridges.

VoiceXML 1.0 (W3C 2000) was introduced a few months after SUEDE was released. One small company that uses SUEDE, Mobido, modified SUEDE to generate a VoiceXML program as output. They are using their modified SUEDE for early testing with Wizard-of-Oz and then are using the VoiceXML generation when they are ready to move their application to their speech recognition production environment.

SUEDE fits best as the first tool in the overall NL interface design process for these users. It works well when used for early experimentation and iterative design.

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 (Rettig, 1994) 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.

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 (Dahlbäck et al., 1993). In a standard Wizard-of-Oz study (Gould and Lewis, 1983; Kelley, 1984), a human simulates the speech system. SUEDE's electronic support for Wizard-of-Oz testing improves on traditional Wizard-of-Oz by making it easy to carry out repeatable tests as well as keep track of what happens during the test sessions.

Yankelovich made frequent use of "pre-design studies" in her work on SpeechActs (Yankelovich et al., 1995). One of our goals was to make pre-design studies easier for designers to carry out. SUEDE's session recording and analysis features make the data generated from these studies easily accessible and even more valuable.

The NEIMO system (Balbo et al., 1993; Coutaz et al., 1996) is a platform for the study of multimodal systems, and SRI's Open Agent Architecture (Cheyer et al., 1998) is a toolkit for implementing multimodal applications. Both allow use of Wizard-of-Oz techniques. However, these systems require functioning software to be written before testing can begin. In contrast, SUEDE is oriented to early stage speech UI design and thus has no such software requirement.

The SUEDE Wizard-of-Oz 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 (Oviatt, 1999) or genders (Vergin et al., 1996); in fact, it does not matter what language they are speaking at all.

Speech-Based UI Construction Tools

There are two existing speech UI construction tools to which SUEDE is similar in several respects: the CSLU Rapid Application Developer (RAD) (Sutton et al., 1998) and the Unisys' Natural Language Speech Assistant (NLSA) (Unisys, 1999).

Both CSLU RAD and NSLA combine speech recognition, speech synthesis, and an available Wizard-of-Oz technique into an integrated tool for building speech applications. It would be possible to use these other tools for the same processes as SUEDE supports, but it would not be as inviting or efficient for a non-programmer designer. Specifically, the Design-Test-Analysis methodology is explicit in SUEDE. Like SUEDE, these tools use a visual, state machine representation of the speech UI (McTear, 1998). Arguably, SUEDE's visual, state machine representation is simpler than CSLU RAD and NSLA, and most closely resembles the NL designers' dialog states representation. This simplicity in visual form and explicit work practice is implemented in SUEDE, because it is meant to be the first tool that the NL designer sits down to use.

Unlike SUEDE, these tools additionally support using a speech recognizer behind the dialog flow. Both CSLU RAD and NLSA provide access to underlying speech recognition grammars and parameters for recognition. CSLU RAD, in particular, has been extremely popular for this capability to build a working speech recognition application in a graphical environment. CSLU RAD and NSLA are oriented towards specifying, testing, and implementing a more finished application interface—what we would call rapid prototyping. SUEDE is targeting specifically a different niche, informal prototyping—quick and easy testing of potential designs. One could imagine a designer first informally prototyping and testing in SUEDE and then transferring a concrete design idea to CSLU or NSLA, where he/she would add the details to test with a recognition system. Rather than building a working application, SUEDE's only focus is on enabling the designer to create a great design.

SUEDE Future Work

As an informal tool, SUEDE offers significant flexibility for designers. Though it is possible to model commonly used NL design features right now, we will be adding more automatic support for speech design features such as tapering, error handling, and cooperative prompts. Another logical extension is to allow SUEDE designs to function as reusable components, to be used in higher-level designs.

SUEDE could be extended for speech-centric applications that have alternative modes of feedback (such as graphics feedback in a multimodal application). Our group is taking ideas from SUEDE, including the Design, Test, and Analysis methodology, and incorporating them in a multimodal prototyping tool (see http://guir.berkeley.edu/crossweaver/).

Conclusions

The spoken-language NL interface design problem is complicated. Explicit tool support, such as in SUEDE, is valuable for improving the spoken language NL design process.

Based on our interviews with NL designers, SUEDE's Design, Test, and Analysis paradigm maps quite well onto the designers' mental process. SUEDE makes significant progress in supporting the early stages of this process. Specifically, designers use scripts as their initial concrete examples, they create dialog flowcharts, and they test their designs with Wizard-of-Oz techniques. SUEDE supports this work process and helps spoken-language NL designers analyze collected test data more efficiently than does reviewing analog tapes.

The high level of frustration associated with spoken-language NL interfaces is not one of medium, but one of design. Design the spoken-language NL interface well, and NL in HCI will proliferate.

References

Balbo, S., Coutaz, J. et al. (1993). Towards automatic evaluation of multimodal user interfaces. *Proceedings of the International Workshop on Intelligent User Interfaces*, pp. 201–208.

Boyarski, D. and Buchanan, R. (1994). Computers and communication design: Exploring the rhetoric of HCI. *Interactions*, *1*:24–35.

Carroll, J.M. (1995). *Scenario-Based Design: Envisioning Work and Technology in System Development*. New York: John Wiley & Sons.

Cheyer, A., Julia, L. et al. (1998). A unified framework for constructing multimodal experiments and applications. *Proceedings of CMC '98*. Tilburg, The Netherlands, pp. 63–69.

Clarke, L. (1991). The use of scenarios by user interface designers. *Proceedings of the HCI '91 Conference on People and Computers*, pp. 103–115.

Cohen, P.R. and Oviatt, S.L. (1995). The role of voice input for human-machine communication. *Proceedings of the National Academy of Sciences*, *92*(22):9921–9927.

Coutaz, J., Salber, D. et al. (1996). NEIMO, a multiworkstation usability lab for observing and analyzing multimodal interaction. *Proceedings of ACM CHI '96 Conference on Human Factors in Computing Systems*, pp. 402–403.

Dahlbäck, N., Jönsson, A. et al. (1993). Wizard-of-Oz studies—Why and how. *Proceedings of the International Workshop on Intelligent User Interfaces*, pp. 193–200.

Gould, J.D. and Lewis, C. (1983). Designing for usability—Key principles and what designers think. *Proceedings of ACM CHI'83 Conference on Human Factors in Computing Systems*, pp. 50–53.

Kamm, C. (1995). User interfaces for voice applications. In D. Roe and J. Wilpon (Eds.), *Voice Communication between Humans and Machines*. National Academy Press, pp. 422–442.

Kelley, J.F. (1984). An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Office Information Systems*, *2*(1):26–41.

Klemmer, S.R., Sinha, A.K. et al. (2000). SUEDE: A Wizard of Oz prototyping tool for speech user interfaces. *CHI Letters, The 13th Annual ACM Symposium on User Interfaces Software and Technology: UIST 2000*. San Diego, CA, *2*(2):1–10.

Landay, J.A. and Myers, B.A. (1995). Interactive sketching for the early stages of user interface design. *Proceedings of the ACM CHI '95 Conference Human Factors in Computing Systems: CHI'95*. Denver, CO, pp. 43–50.

Landay, J.A. and Myers, B.A. (1996). Sketching storyboards to illustrate interface behavior. *Conference Companion of the ACM CHI '96 Conference on Human Factors in Computing Systems*. Vancouver, Canada, pp. 193–194.

Landay, J.A. and Myers, B.A. (2001). Sketching interfaces: Toward more human interface design. *IEEE Computer*, *34*(3):56–64.

Lin, J., Newman, M.W. et al. (2000). DENIM: Finding a tighter fit between tools and practice for web site design. *CHI Letters: Human Factors in Computing Systems, CHI '2000*, *2*(1):510–517.

McTear, M.F. (1998). Modeling spoken dialogues with state transition diagrams: Experiences with the CSLU toolkit. *Proceedings of the Fifth International Conference on Spoken Language Processing ICSLP '98*. Sydney, Australia.

National Research Council (1997). *More than Screen Deep: Toward Every-Citizen Interfaces to the Nation's Information Infrastructure*. Washington, D.C.: National Academy Press.

Oviatt, S. (1999). Mutual disambiguation of recognition errors in a multimodal architecture. *Proceedings of the ACM CHI '99 Conference on Human Factors in Computing Systems*, *1*:576–583.

Oviatt, S., Cohen, P. et al. (1992). A rapid semi-automatic simulation technique for investigating interactive speech and handwriting. *Proceedings of the International Conference on Spoken Language Processing ICSLP '92*. Banff, Canada, *2*:1351–1354.

Rettig, M. (1994). Prototyping for tiny fingers. *Communications of the ACM*, *37*(4):21–27.

Sidner, C. (1997). 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*. Washington, D.C.: National Academy Press, pp. 315–321.

Sutton, S., Cole, R. et al. (1998). Universal speech tools: The CSLU Toolkit. *Proceedings of International Conference on Spoken Language Processing ICSLP '98*, 7:3221–3224.

Tufte, E. (1999). Presenting data and information seminar. San Francisco, CA.

Unisys (1999). Natural Language Speech Assistant, Unisys Corp.

Vergin, R., Farhat, A. et al. (1996). Robust gender-dependent acoustic-phonetic modelling in continuous speech recognition based on a new automatic male/female classification. *Proceedings of the International Conference on Spoken Language Processing ICSLP '96*. Philadelphia, PA, 2:1081–1084.

W3C (2000). Voice eXtensible Markup Language VoiceXML 1.0.

Wagner, A. (1990). Prototyping: A day in the life of an interface designer. In B. Laurel (Ed.) *The Art of Human-Computer Interface Design*. Reading, MA: Addison-Wesley, pp. 79–84.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, *36*(7):74–84.

Yankelovich, N., Levow, G.-A. et al. (1995). Designing speechActs: Issues in speech user interfaces. *Proceedings of ACM CHI '95 Conference on Human Factors in Computing Systems*, *1*:369–376.