Prototyping an Intelligent Agent through Wizard of Oz

David Maulsby * Saul Greenberg * Richard Mander

* Department of Computer Science
University of Calgary
Calgary Alberta T2N 1N4 Canada

maulsby or saul@cpsc.ucalgary.ca
+01 403-220-6087

 Human Interface Group, Apple Computer Inc.
20525 Mariani Ave.
Cupertino CA 90514 USA

mander@appleLink.apple.com
+01 408-974-8136

ABSTRACT

Turvy is a simulated prototype of an instructible agent. The user teaches it by demonstrating actions and pointing at or talking about relevant data. We formalized our assumptions about what could be implemented, then used the Wizard of Oz to flesh out a design and observe users' reactions as they taught several editing tasks. We found: a) all users invent a similar set of commands to teach the agent; b) users learn the agent's language by copying its speech; c) users teach simple tasks with ease and complex ones with reasonable effort; and d) agents cannot expect users to point to or identify critical features without prompting.

In conducting this rather complex simulation, we learned some lessons about using the Wizard of Oz to prototype intelligent agents: a) design of the simulation benefits greatly from prior implementation experience; b) the agent's behavior and dialog capabilities must be based on formal models; c) studies of verbal discourse lead directly to an implementable system; d) the designer benefits greatly by becoming the Wizard; and e) qualitative data is more valuable for answering global concerns, while quantitative data validates accounts and answers fine-grained questions.

KEYWORDS: Intelligent agent, instructible system, programming by demonstration, Wizard of Oz, prototyping

INTRODUCTION

We used the Wizard of Oz method to test a new design for an instructible agent. In this paper we describe how end users learned to teach a simulated agent called Turvy, in particular the set of instructions and commands they adopted. These findings will be valuable to implementors of programming by demonstration systems. We also explore the lessons we learned using the method, which experimenters can apply in their own studies of intelligent interfaces. These lessons differ from other Wizard of Oz experiences in being oriented towards prototyping an implementable system, rather than a proof of concept.

The paper begins with a brief discussion of intelligent agents and the Wizard of Oz method. It then describes the Turvy experiment and results, and ends with a retrospective on our use of Wizard of Oz.

Intelligent agents

When given a goal, [an intelligent agent] could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. –*Alan Kay (1984)*

Intelligent interface agents have been touted as a significant new direction in user interface design. Videos from Apple and Hewlett-Packard show futuristic interfaces in which agents play a dominant role, serving as computerized office clerks, database guides and writing advisors. Reality is a bit behind, but prototype intelligent agents have been implemented by researchers. For example, Eager (Cypher, 1991) detects and automates a user's repetitive actions in HyperCard; it matches examples by parsing text strings and by testing numerical relationships. Metamouse (Maulsby, 1989) learns drawing tasks from demonstrations; it applies rules to find significant graphical constraints.

Yet today's agents are "intelligent" in the narrowest sense of the word. They understand only specialized or highly structured task domains, and lack flexibility in conversing with users. Kay (1984) suggests that agents should be illusions that mirror the user's intelligence while restricting the user's agenda. Unfortunately, because most work on agents stems from the field of Artificial Intelligence, users' needs are second to algorithm development. While the systems prove that particular approaches can be codified in algorithms, they rarely pay more than lip service to the usability tradition of Human Computer Interaction. As a result, they tend to fail as interfaces.

Agents must be designed around our understanding of what people require and expect of them. However, the traditional approach of system building is an expensive and unlikely way to gain this understanding. The underlying discourse models and algorithms for agents are usually so complex and entrenched with assumptions that changes—even minor ones—may require radical redesign. Moreover, because agents act as intermediaries between people and their applications, the designer must craft and debug the agent/application interface as well. A viable alternative to system building is Wizard of Oz.

Wizard of Oz

Wizard of Oz is a rapid-prototyping method for systems costly to build or requiring new technology (Wilson and Rosenberg, 1988; Landauer, 1987). A human "Wizard" simulates the system's intelligence and interacts with the user through a real or mock computer interface.

Most Wizard of Oz experiments establish the viability of some futuristic (but currently unimplementable) approach to interface design. An example is the use of complex user input like speech. Gould et. al., (1982), who pioneered the method, simulated an imperfect listening typewriter to find out whether it would satisfy people used to giving dictation. Similarly, Hauptmann (1989) tested users' preferences for manipulating graphic images through gesture and speech, by simulating the recognition devices.

Other Wizard of Oz experiments concentrated more on human behavior than futuristic systems. Hill and Miller (1988), for example, investigated the complexities of intelligent on-line help by observing interaction between users of a statistical package and a human playing the role of help system. Likewise, Dahlbäck, Jönsson and Ahrenberg (1993) studied differences between human-human and human-computer discourse through a variety of feigned natural language interfaces. In another experiment, Leiser (1989) showed that people can be led to use a language understood by the computer through *convergence*, a phenomenon of human dialog in which participants unconsciously adopt one other's speech pattern. When users typed a natural language database query, the Wizard, using only certain terms and syntactic forms, would verify it by paraphrasing. Convergence occurred when users adopted those same terms in their queries. We will return to this theme in our discussion of "TurvyTalk".

THE TURVY EXPERIMENT

Our research concerns both the technical and usability aspects of programming by demonstration. Previously implemented systems have had problems with competence or usability; we had in mind a more general purpose, easily instructed system that would be nonetheless practical to implement in the near future. The system we envisioned starts with easily coded, primitive knowledge of datatypes and relations, and more specialized knowledge defined by application authors. It then learns higher-level constructs and procedures specific to the individual user's work. For practicality, it must learn under the user's guidance, so it needs an intuitive and flexible teaching interface. We decided that an agent metaphor would help us explore the design issues; the agent is Turvy.

We wanted to see how end users would teach an agent with perfect memory and primitive knowledge. We wanted to see how they structured lessons and whether they could focus its attention correctly. Would they be able to translate cultural concepts (like *surname*) into syntactic search patterns (like *capitalized word after Mr. or Ms.*), and how could Turvy minimize the annoyance of doing so? What kinds of instructions and commands would they use, and what wording? Could verbal input, based on pseudo-natural language or even keyword-spotting, work in conjunction with pointing? We decided to use a Wizard of Oz simulation to investigate these issues, involving end users up front, before making too many commitments in our design.

Using the Wizard of Oz to prototype programming by demonstration makes unusual demands on the Wizard, and we had some special concerns. We wanted to simulate a "buildable" Turvy, so we designed a formal model of the learning system and required the Wizard to obey it. We wanted to separate the agent's behavior (what it can understand and how it can react) from the illusion it presents in the interface, so we let users explore Turvy's abilities through discovery, finding its limits as it responded to their own commands and teaching methods. Finally, we wanted detailed qualitative and quantitative results, so we designed a set of tasks to reveal interaction problems, videotaped sessions, and interviewed the users.

Motivation: previous implementation experience

We had completed a user study of Metamouse, a fully implemented programming by demonstration system personified by an agent named Basil. Basil learns repetitive graphical edits, such as aligning or sorting by height, *provided* the user makes all relevant spatial relations visible by drawing construction lines. In that study we gave users six tasks, all of which Basil could learn in principle. But in practice, Basil frustrated users' attempts to teach it. First, though they discerned the need for constructions, users did not grasp Basil's strict procedural interpretation, and they sometimes used a line to suggest a relation rather than define it. Second, users didn't think it possible to construct some features, like height; they would have preferred to say it in English. Third, they were confused by aspects of the agent metaphor concerning how it searched for sets of objects and created branching procedures. Finally, limitations on Basil's inferencing ability, and occasional crashes due to bugs, compounded their problems and made our observations harder to assess.

We wanted to overcome these deficiencies. We wanted people to converse with the agent through a multi-modal dialog, using demonstrations, English, or property sheets. We wanted people to learn through conversation (verbal or graphical) what the agent could understand. We wanted a metaphor that would present the right illusion. We also knew that a reincarnated Metamouse was not the vehicle for this study, because we would be shackling ourselves to ideas embedded in a big system. We turned to Turvy.

Turvy as agent

We made four key assumptions about the sort of agent we would build, each with consequences for usability. • First, Turvy learns from a user's demonstrations, pointing, and verbal hints. It does not have human-level abilities or knowledge; like Metamouse, it forms search and result patterns from low-level features, and users must refer to them during demonstrations. • Second, unlike Metamouse, Turvy does not equate the user's demonstration with a procedure; actions may be interpreted as focusing attention or extending a pattern, and Turvy can revise an interpretation as more examples are seen. This enables a dialog where users can

Class	From User	From Turvy
Command	√ Watch and learn what I do	√ Show me what to do
	√ End of lesson	√ All done
	Do the next step	√ May I take over?
	√ Do the rest of this example (iteration)	√ May I continue?
	√ Do the next example (iteration)	√ [May I] do the next one?
	√ Do all remaining examples (iterations)	√ [May I] do the rest?
	√ Stop (you've made a mistake)	
	~ Undo [one step] [to the start of this iteration]	~ Undo [last step?] [this iteration?]
	√ Ignore my actions (I'm fixing something)	
	1 Let me take over	You take over
	1 Always let me do these steps	Do you want to do this manually?
Focus attention	1 I'm repeating what I did before	√ I've seen you do this before
	This is similar to [indicates previous example]	√ Treat this like [describes similar item]?
	This case is different	? What is different about this case?
	R Look here (this is important)	? Is this [describes item] important?
	√ Look for [describes item]	√ I should look for [describes item]
	R I did this [conditional branch] because [points at something and/or lists features]	√ Is this [describes feature] what distinguishes the two cases [or new special case]?
	1 I'm changing the way I do this task	You've changed the method here, why?
Response	R OK / yes	√ OK / yes
	R No	√ No
	R I don't know / I don't want to discuss that	√ I don't know, show me what to do

Legend: √ – used without difficulty; 1 – given by only 1 user; R – given only in response to a prompt from Turvy; ~ – not differentiated in usage; ? – questions asked by Turvy that caused user confusion; blank – not used.

Table 1. Messages used in the Turvy study.

teach new concepts on the fly. • Third, an implementable Turvy would not have true natural language capabilities, and our system only recognizes spoken or typed keywords and phrases, using an application-specific lexicon. This poses problems for a user of Turvy's language, for they must learn this lexicon. Verbal inputs are either commands (like *Stop!*) or hints about features (like *look for a word before a colon*), where keywords (*word, colon*) are compared with actual data at the locus of action to determine their meaning. This implies that users must accompany verbal descriptions with editing or pointing actions. • Fourth, Turvy predicts actions as soon as possible, verbalizing them on the first run through. Eager prediction gives users efficient control over learning. Speech output signals the features Turvy has deemed important, without obscuring text or graphic data. As a side effect, the users also learn Turvy's language.

Turvy as system

This section supplies a brief description of Turvy's built-in knowledge, inference mechanism, and interaction model. Maulsby (1993) provides more detail. Turvy learns procedures and data descriptions — specialized types, structures, orderings (cf. Halbert, 1984). Turvy's learning strategy is to make a plausible generalization from one example, then revise it as more are seen or when the user gives descriptive hints. Turvy's tactic is to elicit hints by predicting.

We designed a formal model of the learning system in terms of a grammar for tasks it could learn, then chose an application (text editing), and made a detailed model of Turvy's background knowledge, in the form of an attribute grammar for textual search patterns. The knowledge is more primitive than that in Myers' (1991) demonstrational text formatter. Untutored Turvy recognizes characters, words, paragraphs, brackets, punctuation, and properties like case and font. Turvy learns to search for sequences of tokens with specified properties.

We designed a model of interaction—an incomplete model, formulated as a list of 32 kinds of instructions implied by the learning model. If a person gives all the instructions, the system can learn without inferencing, but we predicted that people would use only the subset listed in Table 1. We believe the user will give incomplete or ambiguous instructions, which the system will complete by making inferences and eliciting further details. Note that we told Turvy's users only that it could understand "some English"; they did not receive a list of instructions or wordings. In turn, the Wizard as Turvy responded only to instructions whose intention corresponded to some message in the interaction model.

Tasks and an example user/Turvy dialog

We made up six tasks and a data set, on the theme of formatting a bibliography:

a. italicize journal titles
b. quote the titles of conference and journal papers
c. create a citation heading with the primary author's surname and year of publication (illustrated in Figure 1)
d. put authors' given names and initials after surnames
e. put book titles in Times-Roman font
f. strip out colons that separate bibliographic fields

Tasks involved domain concepts like "title," "journal vs. book," and "list of authors," concepts Turvy does not understand. Users were shown before and after snapshots of example data for each task; no mention was made of the syntactic features Turvy would learn. This tested Turvy's ability to elicit effective demonstrations and verbal hints.

Philip E. Agre: *The dunamic structure of everyday life.* PhD thesis: MIT Artificial Intelligence Lab: 1988.	**[Agre 88]** Philip E. Agre: *The dunamic structure of everyday life.* PhD thesis: MIT Artificial Intelligence Lab: 1988.
John H. Andreae, Bruce A. MacDonald: "Expert control for a robot body." *J. IEEE SMC:* July 1990.	**[Andreae 90]** John H. Andreae, Bruce A. MacDonald: "Expert control for a robot body." *J. IEEE SMC:* July 1990.
Michalski R. S., J. G. Carbonell, T. M. Mitchell (eds): **Machine Learning II.** Tioga: Palo Alto CA: 1986	**[Michalski 86]** Michalski R. S., J. G. Carbonell, T. M. Mitchell (eds): **Machine Learning II.** Tioga: Palo Alto CA: 1986
Kurt van Lehn: "Discovering problem solving strategies." Proc. Machine Learning Workshop, pp. 215-217: 1989.	**[van Lehn 89]** Kurt van Lehn: "Discovering problem solving strategies." Proc. Machine Learning Workshop, pp. 215-217: 1989.
a. before editing	b. after putting first author and date into heading

Given the document on the left, users were asked to place the first author's name and the date of publication in square brackets with bold styling prior to each reference. When correctly reformatted the bibliography would appear as on the right.

Figure 1. Sample data from Task c, "make citation headings".

Similar concepts were repeated in later tasks, to see whether users adopted Turvy's description.

Figure 1 shows a sample of the data for Task c. The user is supposed to make citation headings for each entry, using the primary author's surname and part of the date. From Turvy's point of view, these items are the word before the first colon or comma in the paragraph, and the two digits before the period at the paragraph's end. Exceptional cases to be learned include a baronial prefix (eg. "van Lehn") and initials after surname (eg. "Michalski R. S.").

```
TaskC: repeat (FindSurname FindDate)
FindSurname: if find pattern (Loc := BeginParagraph
    SomeText Surname := ({0 or more LowerCaseWord}
    Word) [Colon or Comma])
    then do MakeHeading else TurvyAllDone
MakeHeading: select Surname; copy Surname;
    put cursor before Loc; type "["; paste Surname;
    type (DateLoc := Blank "]"); type Return;
    select styleMenu; select "citation"
FindDate: if find pattern (Date := (Digit Digit)
    Period EndParagraph)
    then do CopyDate else TurvyAskUserForDemo
CopyDate: select Date; copy Date;
    put cursor after DateLoc; paste Date
```

Figure 2. Pseudo-code derived by analyzing Task c (from Figure 1) with Turvy's inference model.

Prior to the experiment, we analyzed this task using the inference model and got code similar to that in Figure 2 (we've omitted the code for handling initials after a surname). Coding helps the Wizard act consistently, even if users teach somewhat different procedures. The code loops over all entries in the bibliography. Each step involves finding a pattern in the text (to instantiate data descriptions) and then executing a macro; if not found, a default action is done. Macros are straight-line procedures that refer only to instantiated variables. Mnemonic variable and macro names are shown in italics.

Below is the trace of a dialog between Turvy and "Kate", a composite of two users in our study (the first paragraph was spoken by the one, the rest by the other). Kate is working through Task c: the first example is "Agre" in Figure 1.

Kate: *Take the last name, it will be in some cases before a colon.* [Kate copies "Agre", pastes it into a new paragraph, types the brackets and sets the style.] *This is a list of publications. Now we're going to add the last two digits of the date* [types "88"].

Kate gives hints about the syntax ("before a colon", "last two digits") and the subject matter ("name", "a list of publications"). Turvy interprets only the hints about syntax, and then only in relation to actions demonstrated (so "last two digits" is taken to mean the digits typed, not those at the end of the paragraph).

Kate then signals the end of the lesson and gives Turvy a chance to take over or get another example. Turvy edits the next entry (not shown in Figure 1).

Kate: *Do you want another demo?*
Turvy: I can try if you want. — *Kate: OK.*
Turvy: Looking for word before first colon in next paragraph [picks surname], OK? — *Kate: OK.*
Turvy: [while doing actions] Copy, go to start of paragraph, type return, paste, type brackets, change style to "citation", OK? — Kate: *OK.*

Now Turvy makes a mistake. Kate shows the right action and Turvy echoes back a new hypothesis reflecting her hint from earlier on.

Turvy: Type in blank space "88", OK?
Kate: *No, the last two digits at the end.*
Turvy: Show me please. — [Kate picks digits.]
Turvy: Two digits before period at end of paragraph?
Kate: *Yes.*
Turvy: Continue? — *Kate: OK.*

The next example introduces a new case, multiple authors (see the second entry in Figure 1). Turvy's search pattern selects the last one. Kate tries to correct Turvy by giving a hint, but Turvy insists on a demonstration to go with it.

Turvy: [picks "MacDonald"] — *Kate: No – stop.*
Turvy: Show me please.
Kate: *I want the last name of the first author.*
Turvy I don't understand – show me please.
Kate: *When you have a number of authors, choose the first* [picks "Andreae"]. *It's before the comma.*
Turvy: Word before first comma or colon? — *Kate: Yes.*

A little later on, Turvy comes to an entry in which the surname happens to precede the initials ("Michalski in Figure 1). Turvy's search pattern picks the last initial.

Turvy: [picks "S."] — *Kate: No.*
Turvy: Show me please.
Kate: *I want you to take in this case the very first word* [picks surname], *then continue as before.*
Turvy: You skipped back over words ending in period?
Kate: *Yes.*
Turvy: Continue? — *Kate: OK.*

Finally, Kate teaches Turvy how to parse names with baronial prefixes ("van Lehn" in Figure 1).

Turvy: [picks "Lehn"]
Kate: *Ok, um, stop* [hesitant]. *In this case I want you to select the last two words.*
Turvy: Show me please. [Kate picks "van Lehn".]
Turvy: What's different about this case?
Kate: *Hmm* [confused].
Turvy: You selected these two words, why?
Kate: *Because these two words make up his last name.*
Turvy: Is there some feature of that word [hilites "van"] I should look for? That word exactly?
Kate: *In this case, yes* [hesitant].

As in previous lessons, Kate gives an explanation with her demo, but it fails to distinguish this case. When she is stymied, Turvy suggests a plausible (though wrong) hypothesis, which she accepts for now.

To summarize the dialog: Kate's goal is to get Turvy to do the task (until a new case requires further teaching), and Turvy's goal is to get more specific instructions from Kate. Typically, Kate does an example, then Turvy does the next one. Turvy describes actions when predicting them the first time. When Turvy errs, Kate demonstrates the right steps and gives a verbal hint; Turvy echoes back an interpretation. If it cannot make use of Kate's hint, Turvy proposes a guess, to elicit further explanation.

We realized from the outset that it would be all too easy for the Wizard to slip out of character, despite our formal models. In order to sustain the simulation of algorithmic intelligence, we decided to minimize the amount of new information the Wizard has to cope with. Hence we had users do standard tasks on data the Wizard had prepared. Moreover, all tasks were designed to limit the user's options: there were no "inputs" (data was merely cut, copied, pasted) and few points at which the order of steps could be varied. As mentioned above, each task was analyzed beforehand, so that inferences made from examples (given in a standard sequence) were in effect pre-scripted. Thus the Wizard was able to focus on the user's speech and gestures, and respond consistently. We also made Turvy a little extra stupid (for instance, guessing "van" rather than lowercase), to get more experimental data.

Hypotheses

Formalizing the inference and interaction models revealed the complexity of various kinds of instructions and the information needed to interpret them. This helped us form hypotheses about the way people would construct lessons and the instructions they would use. We had four main working hypotheses.

1 All users would employ the same small set of commands, those given in Table 1, with only minor variations in wording.
2 Users would learn "TurvyTalk" (Turvy's low-level terminology for describing search patterns), but only as a result of hearing Turvy describe things. Moreover, if Turvy uttered (perhaps in the form of a question) some instruction the user had previously given but with different wording, users would thereafter adopt Turvy's wording. This hypothesis is based on verbal convergence (Leiser, 1989), as mentioned in the introduction.
3 Users would teach simple tasks with ease and complex ones with reasonable effort. In other words, the complexity of instruction would depend on the task, not on peculiarities of Turvy's learning method.
4 Users would point to focus Turvy's attention on parts of a search pattern. Pointing avoids the problem of how to name something, but has the disadvantage of being ambiguous; the teacher must rely on the learner to infer the correct granularity and property.

Experimental setup

In our experiment users sat at a Macintosh computer and worked on bibliographic entries using the Microsoft Word text editor. A facilitator sat next to the user. Nearby (but out of eye contact) sat Turvy, played by the system designer, who had a second keyboard and mouse also connected to the Macintosh. The video signal from the computer was split-routed to both the user's and Turvy's screens. This setup allowed them to view the document simultaneously, yet restricted their physical interaction to pointing on screen with the cursor. They competed for control of the cursor, but this caused no problems in practice. For each task, the facilitator opened an on-screen document containing the references, then showed an annotated picture of the reformatted result. The user would practice on several entries until able to reformat them correctly. At this point the facilitator would ask the user to teach Turvy, and the Wizard became active, speaking and acting as if Turvy were present in the software. To reinforce the fantasy, the Wizard spoke in clipped sentences, with rather mechanical intonation. While we did not deceive users, they quickly bought into the illusion. They spoke more curtly to Turvy than to the facilitator, and referred to Turvy and the Wizard as two separate entities.

In this study, we did a pre-pilot, a pilot and a final run. Originally, we planned a Wizard of Oz without speech. The

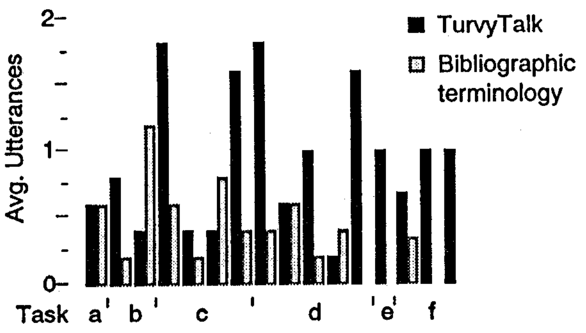

Figure 3. The language spoken by users.

pre-pilot used a menu, with the facilitator-to-be acting as user. When it became clear that our user misunderstood the menu commands, we realized we ought not to predetermine the user's language. In the pilot, four users tested a speech input version of Turvy; they could compose their own commands, but initially Turvy spoke only to ask very general questions like "What's important here?" Generality confused them, but was rectified when questions were made context specific, and when Turvy described its actions and proposed guesses. In the main experiment, eight subjects (five with some programming experience) tested the revised version, whose behavior was held constant.

OBSERVATIONS AND RESULTS

Our data consists of video tapes, transcripts, and the experimenters' subjective observations. We studied comments made by users while working with Turvy and during interviews. We also did content analyses, counting the number of bibliographic terms vs. TurvyTalk in users' instructions, and measuring indicators of confidence, hesitation and confusion at various points of interest during the session.

Command set (Hypothesis 1)

Users gave a close-fitting subset of the instructions we had predicted. From Table 1 we see that nearly all commands were used and caused no difficulty. The actual wordings subjects used were quite consistent, especially after they heard Turvy ask the corresponding question: they would turn it into a command, such as "Do the rest." (See TurvyTalk, Hypothesis 2).

Instructions for focusing attention were problematic. Users almost never volunteered vague hints like "I'm repeating actions," "this is similar," and even "look here." On the other hand, users would give hints in answer to Turvy's questions. The wording of focus instructions was more variable than for commands or responses.

TurvyTalk (Hypothesis 2)

We found that users did learn to describe things like titles and surnames in terms of their syntax. In post-session interviews, all users said that Turvy does not know about bibliographies; few could describe the sort of terminology it does understand, but they could list examples. We did a content analysis of users' speech to confirm these findings. Dividing the entire session into 16 events for different phases of tasks (first example, points where Turvy would err, etc.), we counted the number of user utterances referring to features in terms Turvy understood (TurvyTalk) versus those involving bibliographic terminology (eg. "paste after the author's name"). The analysis confirmed our qualitative findings. In particular, as shown in Figure 3, use of bibliographic terminology tapers off and TurvyTalk increases. We conclude that Turvy's speech quickly trained users to mirror its language—verbal convergence occurs. Even more interesting is users' tendency to juxtapose both languages in one instruction, as if trying to relate their concepts to Turvy's.

Teaching difficulty (Hypothesis 3)

One of our chief aims is to make simple tasks easy to teach, and complex tasks teachable with reasonable effort. In our study, easy tasks (like changing underlined text to italics) were trivially taught by giving a demonstration. The hard tasks (like reversing author's surnames and initials) involved more verbal description of special cases.

All but one user reported that Turvy was easy to teach, once they had realized it learns incrementally and continuously so they needn't anticipate all special cases. However, one user told us at the outset that no computer could be taught without anticipating all cases, and therefore refused to try. No one complained about having to speak TurvyTalk.

We sought to objectify users' impressions of teaching effort in a content analysis of speech characteristics, with ratings measuring high confidence through to confusion. We found that users had a fairly neutral feeling of control; however, dealing with unexpected cases caused anxiety.

Speech versus pointing (Hypothesis 4)

One instructional technique we hoped to find was pointing to focus attention, but we observed almost none (apart from explicit selections required by tasks). When Turvy asked users to explain a new case by "pointing to something in the text," they were confused, if the distinguishing feature was a property rather than a string. We concluded that this query is ineffective; instead, Turvy should propose a guess.

Dialog

We found two distinct styles of interaction, not anticipated in our hypotheses. One type of user is highly interactive, talkative; the other is quiet, non-inviting. A talkative user describes a task before starting it, then does one example and gets Turvy to try the next. Some feel duty-bound to explain expected special cases in advance, but find this hard to do. Soon they learned to wait for special cases to arise.

In contrast, a quiet user works silently through the first example and goes on to the next one without inviting Turvy to try, nor even signaling that the first is done. When Turvy detects repetition, it butts in "I've seen you do this before, may I try?" The user consents, and the rest of the dialog proceeds much as for talkative users, though a quiet one is more likely to tell Turvy to skip a troublesome case than try explaining it.

Talkative users say less as they grow more adept at using Turvy; quiet ones stay quiet. A content analysis showed that talkative users gradually grow more confident, with lapses at special cases, while quiet ones, initially very confident, lose their edge as tasks get harder.

In the post-session interviews (see the Appendix for sample questions), we found that both types of user formed similar models of Turvy's inference and interaction. They recognized that it learns primarily by watching actions, but also understands verbal hints. All users liked the way Turvy is eager to predict after one example, because they believed this gave them more control over learning. All users were concerned about completeness, correctness and autonomy, believing it would be foolhardy to leave Turvy unsupervised until all cases had been covered. Some of the programmers thought that seeing a printout of the procedures Turvy learned would be helpful; the other users disagreed.

All users (apart from the one who refused to teach) said they would use Turvy in their daily work, if only it were real. Most wished that Turvy would learn their language.

To generalize the experiment, we also tried Turvy on drawing and file management with four different subjects. The results were the same, except that we saw some use of pointing (at fields in file listings).

LESSONS LEARNED USING WIZARD OF OZ

Turvy is the most complex Wizard of Oz simulation done to date. The lessons we learned will be of value to those who want to use the Wizard of Oz in their own studies of intelligent interfaces. These lessons differ from other Wizard of Oz experiences, being oriented towards implementable systems, rather than proof of concepts.

Ironically, prior implementation experience is invaluable. The chief danger in using a Wizard of Oz is ascribing to the Wizard powers which no real system could have. This casts doubt on the validity of hypotheses and yields inappropriate, optimistic results. We wanted Turvy to help us design a system of the near future, so the danger is magnified. A prior implementation (like Metamouse) suggests limitations to be put on the Wizard's intelligence, as well as interesting research questions.

The Agent's behavior should be based upon an algorithm. Another way to keep the simulation honest is to base it on an algorithm or at least a fairly detailed formal model. This ensures consistent behavior and experimental repeatability. When the algorithm is too complex for the Wizard to run in real time, it can be used to analyze tasks in order to script the Wizard's behavior. We did this in Turvy by designing a formal learning model, and by "scripting" the Wizard's responses by running the tasks through the model and codifying the results.

The Agent's dialog capabilities should be based upon a constrained interaction model. Similarly, true natural language understanding by computers is far in the future. A realistic dialog must be constrained by an interaction model that explicitly lists the kinds of instructions the system can understand and the feedback it can formulate.

You can build real systems derived from studies of verbal Wizard-human discourse. Even given an interaction model, some would argue that discourse rules derived from studies of verbal dialog cannot be incorporated in today's technology. We offer a glimpse into our current work as an existence proof that this is indeed possible. Moctec (Maulsby, 1992) is a working prototype, limited to simple search and replace tasks, and using a subset of the Turvy instructions without an agent metaphor. Users direct it by menus, voice buttons, and text-based pseudo-natural language hints. Cima (in progress) is a full implementation of Turvy's inferencing and interaction techniques.

The Designer benefits by becoming the Wizard. Perhaps the most important aspect of Turvy was the "training" the designer received by playing the role of agent. Being personally responsible for a user's discomfort and confusion motivates revisions! And simulating an incomplete design reveals its ill-defined aspects.

Qualitative results are the most valuable. By acting as Wizard, facilitator, and interviewer, the experimenters become immersed in the experiment and many important results become obvious. The most interesting experimental questions cannot be answered by statistics, at least in small, cheap studies. Still, measurements are useful: they validate the opinions of experimenters and users, and allow a detailed (if myopic) exploration of particular activities.

Interviews are essential, and video records are useful. We want to know what works, what doesn't, and why. Moreover, we want to know what concepts the users are constructing to explain the system to themselves. Accounts based on running commentaries and interviews are the most efficient means of finding out. Video records permit content analyses of speech and gesture in the users' visual context.

CONCLUSIONS

We prototyped an instructible computer agent using a Wizard of Oz simulation. The agent, Turvy, learns procedures and data descriptions from one or more examples done by the user, combined with verbal and pointing hints. The Wizard of Oz allowed us to test ideas without implementing a system, and thereby involve end-users in several iterations of Turvy's design. The simulation was constrained by formal models of inference and interaction, so that Turvy would have realistic limitations. As a result, we found a natural teaching protocol that we believe we can implement in a real system; in fact, we have implemented parts of it in a second prototype.

In some ways, the Turvy we tested was more stupid than one we would implement. It had no special purpose task knowledge and no ability to apply what it learned in one task to another one later on. Moreover, it learned concepts, but not the user's terminology. Nonetheless, users accept Turvy because it learns new cases on the fly, and makes good use of both demonstrations and verbal hints.

We learned valuable lessons about the Wizard of Oz, in particular the benefits of formal models, detailed task analysis, and direct feedback from users to the designer.

ACKNOWLEDGMENTS

Apple Computer Inc. and the Natural Sciences and Engineering Research Council of Canada provided the funding. Allen Cypher and Shifteh Karimi advised us (Thanks!).

REFERENCES

A. Cypher (1991) "EAGER: Programming repetitive tasks by example." *Proc. ACM CHI'91,* pp. 33-40.

N. Dahlbäck, A. Jönsson, L. Ahrenberg (1993) "Wizard of Oz studies—Why and how." *Proc. ACM International Workshop on Intelligent User Interfaces.*

J.D. Gould, J. Conti, T. Hovanyecz (1982) "Composing letters with a simulated listening typewriter." *Proc. ACM CHI'82,* pp. 367-370.

D.C. Halbert (1984) "Programming by example." Xerox PARC Research report OSD-T8402. Palo Alto CA.

A.G. Hauptmann (1989) "Speech and gestures for graphic image manipulation." *Proc. ACM CHI'89,* pp. 241-245.

W.C. Hill, J.R. Miller (1988) "Justified advice: a semi-naturalistic study of advisory strategies." *Proc. ACM CHI'88,* pp. 185-190.

A. Kay (1984) "Computer software." *Scientific American,* 251(3), pp. 53-59, September.

T.K. Landauer (1987) "Psychology as a mother of invention." *Proc. ACM CHI+GI'87,* pp. 333-335.

R.G. Leiser (1989) "Exploiting convergence to improve natural language understanding." *Interacting with Computers,* 1(3), pp. 284-298.

D. Maulsby, K.A. Kittlitz, I.H. Witten (1989) "Metamouse: specifying graphical procedures by example." *Proc. SIGGRAPH '89,* pp. 127-136.

D. Maulsby (1993) "Simulating an instructible interface: the Turvy experience," in *Watch What I Do: Programming by Demonstration,* A. Cypher et al eds (in press).

D. Maulsby (1992) "Prototyping an instructible interface: Moctec." *Proc. ACM CHI'92,* pp. 52-53.

B.A. Myers (1991) "Text formatting by demonstration." *Proc. ACM CHI '91,* pp. 251–256.

J. Wilson, D. Rosenberg (1988) "Rapid prototyping for user interface design." In *Handbook of Human-Computer Interaction,* pp. 859-875, M. Helander ed., New York, North-Holland.

APPENDIX — INTERVIEW QUESTIONS

Questions asked of users during post-session interviews are listed below. The questions are intended to focus the user's thoughts on key aspects of working with Turvy: translating cultural concepts into syntactic proxies, speech input and output, eager prediction and supervision. We also asked users how they felt about our experimental method. We often reworded questions to refer to specific things that happened during a session. For lack of time, we did not ask all questions of all users.

Did you find some tasks hard to do yourself? Which ones?

How much English do you think Turvy understands? Does he understand whole sentences?

Were you surprised at what Turvy understood? What had you expected?

You gave Turvy some instructions like "select the author's last name." Do you think Turvy understood those?

Do you think Turvy can learn a concept like "name"?
Do you think Turvy understands instructions like "look for the word before the italic text"?

What kinds of bibliography features do you think Turvy can recognize? What did Turvy know before you taught him?

Turvy described what he was doing. Did this help you? Did you find the details distracting?

Can you recall one or two descriptions Turvy read out?

Do you think Turvy can learn from verbal instructions alone — can you simply tell him what to do and then get him to do it?

Could you list some commands Turvy recognizes?

Did you want Turvy to start predicting actions as soon as he did? How many examples do you think Turvy should wait for before he starts predicting?

Sometimes Turvy would change the order of the steps you had demonstrated when he did them. Did this bother you?

Do you think it's best to tell Turvy in advance about all the special cases, or wait for them to arise?

Sometimes Turvy would say "What's different here, can you point to something in the text?" Did you understand this question? What did Turvy want you to do?

Do you think Turvy remembers what he learned in one task when you teach another one? For instance, you did two tasks that involved searching for surnames. Did you expect Turvy to remember?

Did you feel comfortable with the fact that Turvy made a number of mistakes while learning? Would you run out of patience if using Turvy for real?

At some point in every task you would let Turvy "do the rest". Did you watch him closely while he zipped through them? Would you rather watch step by step or review his work afterwards?

Would you trust Turvy to ask you about questionable items, or do you suppose he would just do them, stupidly?

Would you like to be able to see a printed program or script of the task as Turvy learns it? Would this help you?

What did you like about Turvy? What did you dislike?

Would you like to be able to see Turvy (as an icon)?

Would you use Turvy in your day-to-day work?

What additional kinds of knowledge (if any) do you think Turvy should come bundled with?

During the experiment, did you get the impression that Turvy was real, or were you constantly aware that you were talking to a person?

Did the experiment cause you any problems? Were the tasks too difficult? Did it go on too long? Did you feel pressured, or that you were being tested?