1. INTRODUCTION

Developments in the last decade have given us computers that are faster and more versatile, yet smaller and more affordable. These machines are no longer restricted to scientific and educational institutions; nor is their accessibility limited to a privileged and well trained group of people.

With the improved hardware and a greater diversity of users, software products have gained functionality and complexity. One of the many problems of complex systems is the difficulty of training beginners to become proficient users [1].

1.1 Forms of Online Aid

Many methods of online aiding have been proposed, implemented and tested. The simplest and most common form is the online manuals. Online manuals are usually nothing more than electronic versions of the traditional user manuals. An example of this form of online aid is the `man` command in UNIX.

More sophisticated help facilities are intelligent tutoring systems (ITS) or intelligent computer-aided instructions (ICAI) like the *Lisp Tutor* [2] and *Proust* [3]. They challenge users with relatively simple tasks to work on and provide guidance when the users make mistakes. Even more sophisticated facilities provide users with assistance as they do their everyday work. They allow users to "learn by doing." A good example of such online advising facilities is the *UNIX Consultant* (UC) [4].

Many other novel approaches to online help have been proposed and developed [Carroll and McKendree, 5]. Examples are Carroll and Carrithers' "training wheel's approach" [6] and Owen's *DYK* (Do-You-Know) [7].

1.2 Problems of Online Aid

Most of the above forms of online aid are designed to assist users in the completion of their current tasks. Indeed, "current work has focused on system-initiated advice given in the context of error recovery" [5]. However, research has shown that advice on error recovery is particularly difficult as users are always directed to follow the advice immediately. If the advice is misunderstood or incorrect, users become angry and are likely to ignore future advice.

In an experiment conducted by Hill and Miller [8], subjects were asked to play the role of statistical analysts who are analyzing data for *Consumer Reports*. They were taught to seek advice by pressing a help key. After hitting the help key, the subjects could send English questions to a hidden-operator advisory-system. The operator then provided the subjects with the answers to their questions. Detailed analysis of the subjects' actions "indicates that clients followed prescriptive advice[1] effectively and efficiently in slightly more than half the cases. For other cases, clients performed twice as many actions as needed in three times as much time without ever reaching prescribed states" [9]. These findings are attributed to the incorrect presumptions made by the hidden operator with regards to what the subjects knew. This experiment highlights the importance and necessity of having a good understanding of the user.

Carroll and Aaronson [10] added a simulated active help system to a database and report application. This study indicates that "although intelligent help can support users, there are also specific potential problems." Two of the problems noticed are: (1) participants do not always notice what is on their display; (2) users' goals are constantly changing and rather unpredictable.

1.3 Motivation for *UNIX NOTICES*

UNIX Notices (UN) was developed to study the concept of providing non-error recovery help to computer users. Such a facility will hopefully avoid most of the problems related to error recovery help and thus be available as a simple but useful tool for novices. It aims to satisfy a secondary (yet still important) objective of online aid facilities:

[1] *Prescriptive advice* are prescriptions of what to do at the interface; as opposed to *descriptive advice* what are descriptions of the interface.

to help users learn to complete future tasks as efficiently and effectively as possible.

UN provides help to its user by 'posting' helpful notices on the user's screen whenever appropriate. In other words, it 'advertises' new concepts or commands in the hope that the user will find some used for the concepts or commands in the future. UN is related to David Owen's *DYK* [7] system as it adopts the "answer first, then questions" paradigm. However, UN is active, context-sensitive and user adaptive; it has been designed to 'volunteer' helpful notices even at the risk of interrupting users and to post only notices that are relevant to the users and their current activities.

2. ABOUT *UNIX NOTICES* (UN)

UN is a simple active advisory facility designed to help UNIX novices expand their mental model of the operating system. The two main goals of UN are:

(1) introduce users to more efficient ways of doing their work. For example, if the user is using the commands `cp` and `rm` to rename files, UN will advise the user to use the command `mv`.

(2) introduce new commands/concepts that may be of use to the user. For example, the command `head` will be introduced to users who are continuously `more`-ing files. And if the user is already aware of the `head` command (though not totally familiar with it), UN may inform the user that `head` has an option that will change the number of lines that are displayed (instead of the default of 10).

2.1 Implementation of UN

Theoretically, any advisory facility should be designed together with the actual system. However, this is usually not possible. As noted by Carroll and Aaronson [10], "developing advisory solutions for the leading edge of human-computer interaction is hampered by the fact that the leading edge must have already been codified and deployed before advisory problems can even exist."

Thus, UN was built on top of the UNIX operating system. However, care was taken to make sure that the new routines do not interfere with the 'normal' performance of the system. In particular, the program must not take up too much memory nor decrease performance. Hence, the *detector-verifier* mechanism for detecting situations suitable for posting advice notices was developed (figure 1).

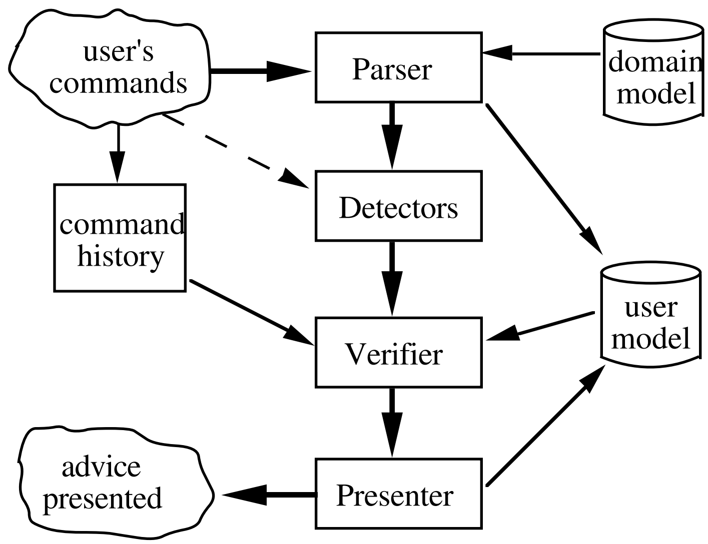

Figure 1: Flow diagram of UN

Every command entered by the user is logged and passed to the *parser*. The *parser* breaks the command line down into the command name (e.g. `cp`, `rm`, `mv`), its options, its arguments (e.g. file or directory names), and UNIX shell symbols (e.g. output re-direction: <, >; history command: `!$`) if any. This information is then passed to the *detectors*. It is also used to update the user model.

The *detectors* identify situations that *may be* appropriate for posting helpful notices. There are two main *detectors*. The first *detector* watches for patterns in file usage. For example, if a file was used as the input to a `cp` command and then used as an input to a `rm` command, it is likely that the file was renamed by the user. The *detector* will note that a "use `mv` to rename" notice may be appropriate.

The second *detector* watches for patterns in command usage. For example, if the user uses two `rm` commands consecutively, it is likely that the user does not know how to delete more than one file with a single `rm` command. A possible situation for posting a notice is thus detected.

The *verifier* verifies that the advisory situations detected by the *detectors* are really correct. In other words, it filters the detected notices so as to prevent users from receiving inappropriate notices. For instance, if the `mv` command has been used by the user recently, the "use `mv` to rename" notice will be rejected as inappropriate. Notices validated by the *verifier* are then passed to the *presenter* to be given to users.

2.2 Problems Studied

A central concern was the design of the *presenter* since it determines the overall effectiveness of UN. Two specific questions that we tried to answer were: "How to present the advice notices?" and "When to present the notices?"

<u>How to present the advice notices?</u>

The *presenter* displays the advice notices in a separate window. This method of 'advertising' information is analogous to roadside posters; we hoped that the users will glance at the display while they are waiting for something else to happen. We compared the multi-window presentation method with two others:

(1) when an advice is considered appropriate, the user's prompt is changed and the user is required to enter a command in order to read the advice notice.

(2) the advice notice is shown in the user's window as soon as an advisory situation is detected and verified.

The experiment is described in detail in section 3.

<u>When to present the notices?</u>

Our initial belief was that the feedback should be immediate. This way, users will be able to relate the advice with their current activities. They will also be in a position to try out the information given in the notice immediately. However, the problem with immediate feedback is that it may disrupt the user's work.

Corbett and Anderson [11, 12] conducted several experiments that examined the effects of feedback control on learning to program with the Lisp Tutor. They studied three feedback conditions along with a control group (no feedback). The three conditions are:

(1) Immediate feedback -- the feedback message is given as soon as the user makes an error.

(2) Error flagging -- errors are displayed in bold; the user can ask for more information or continue coding.

(3) Feedback on demand -- the user must take the initiative and ask the tutor to check over the code.

The results showed that the three forms of feedback conditions did not affect how well the students learned the Lisp lessons. However, in the second experiment [12] where the lessons were made more difficult, subjects with the first two conditions were able to complete their exercises faster. Among Corbett and Anderson's observations are:

(1) Subjects generally preferred more control over the Tutor. However, they don't take full advantage of the added flexibility.

(2) Feedback conditions affect the subjects self-perception. "Students who received less assistance from the tutor seemed more confident of the skill . . . Students who received immediate feedback seemed to assess their knowledge more realistically" [11].

Our second experiment on the effects of feedback delay considered two feedback conditions:

(1) immediate feedback -- the user is informed of advice notices as soon as they are found appropriate;

(2) feedback at end of session -- all appropriate advice notices are shown at the end of

the user’s work session.

An additional treatment with no feedback was added to form the control group. The experiment is described in detail in section 4.

3. EXPERIMENT 1: PRESENTATION STYLE

3.1 The 3 Presentation Styles

Separate window

Once an advice is identified as possible and then validated, a notice will be displayed in a window to the upper-right of the users’ window. Figure 2 shows what the advice window may look like in the middle of a session. At the start of the session, when no advisory situation has been detected yet, the advice window displays a quotation. This is considered better than leaving the window blank.

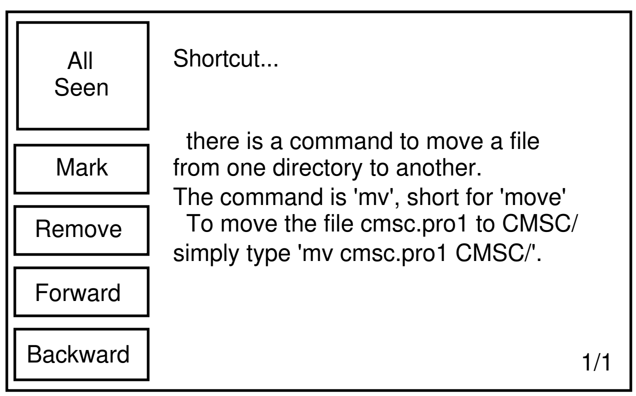

Figure 2: Advice window display

The advice notices that are given to the users are not automatically removed after they are seen. Instead, they are maintained by UN. The users can read previous notices by using the mouse to click the *Forward* and/or *Backward* buttons. Sometimes a few advisory situations may appear so close together that last new notice over-laps the other new notices before they had a chance to be read. If this situation should happen, a *See New* button will be shown and users can click on this button to see the ‘hidden’ notices.

In addition to the three buttons mentioned above, there are the *Remove* and the *Mark* buttons. If users click the *Remove* button, the notice that is currently displayed is removed from the list of advice notices maintained by UN. The notice will never appear again in the on-going session. This button is provided to allow the users to get rid of notices they do not like.

The examples provided with the advice notices will change if UN detects new situations where the same notices are appropriate; in other words, the examples are kept up-to-date with what the user is doing. To stop an example from changing, the users have to click the *Mark* button when viewing the advice notice. This button is also provided to help users organize the notices.

Change of prompt

As soon as an advisory situation is validated by the *verifier*, the users' prompt is changed (a smiley face is added to the front of the prompt, see figure 3) to alert them to the existence of an advice notice. The users have to type the command `hello` to read the notice (see figure 4).

```
ADDRESS/   FALL91/         RESEARCH/       RESUMES/        SPRING91/
<JOHN> cd SPRING91
<SPRING91> ls
ARTH/           arth.final      cmsc.pro1       cmsc.term
CMSC/           arth.rep1       cmsc.pro2
<SPRING91> cp cmsc.pro1 CMSC/
<SPRING91> cp cmsc.pro2 CMSC/
<SPRING91> cp cmsc.term CMSC/
<SPRING91> rm cmsc.pro1
rm: remove cmsc.pro1? y
:-) <SPRING91>
```

smiley face

Figure 3: Smiley face

```
Shortcut...

            there is a command to move a file
            from one directory to another.
            The command is `mv', short for `move'.

            To move the file cmsc.pro1 to CMSC/ ,
            simply type `mv cmsc.pro1 CMSC/'.

                                             ...message 1 of 1


    All messages have been seen.

Type    <F> to go forward;          <B> to go backward
        <M> to mark this advice;    <N> to see new
        <R> to remove this advice;  <Q> to quit
```

Figure 4: Sample notice

They can read previous notices by typing the characters 'F' (for forward) and 'B' (for backward). To read a previously unread notice, they can type 'N' (for new). The users can also 'R'emove or 'M'ark a notice. When they are done reading the notices, they have to type 'Q' (for quit) to return to the normal UNIX screen. The prompt is also returned to normal.

The users are allowed to enter the `hello` command even when the normal prompt is showing. This allows users to read previous advice notices at any time. If UN has no notices for users, it will simply display a quotation.

Immediate display

```
CMSC/           arth.rep1       cmsc.pro2
***************   ;-)  *****************
Shortcut...

 you can change from one sub-directory
to another directly without going to
the main directory first.
 Simply type `cd ~/ SPRING91'.
****************************************
<SPRING91> cp cmsc.pro1 CMSC/
<SPRING91> cp cmsc.pro2 CMSC/
<SPRING91> cp cmsc.term CMSC/
<SPRING91> rm cmsc.pro1
rm: remove cmsc.pro1? y
***************   ;-)  *****************
Shortcut...

 there is a command to move a file
from one directory to another.
The command is `mv', short for `move'.
 To move the file cmsc.pro1 to CMSC/ ,
simply type `mv cmsc.pro1 CMSC/'.
****************************************
<SPRING91> █
```

Figure 5: Sample session of user using Immediate display

The advice notice is shown in the user's window as soon as it is validated. To make sure that the notice does not appear while the user is listing or editing a file, the appearance of the notice is delayed until immediately before the next prompt. Only one notice is shown at a time before a prompt; other advice notices are put in a queue to wait for the next prompt. Figure 5 shows what the user's window may look like after the presentation of two advice notices. Unlike the other two methods of presentation, it is not possible for users to read previous notices.

3.2 Experimental Design

Subjects with no UNIX experience were first given a tutorial on the basics of UNIX. They were then asked to perform a series of tasks with UN executing in the background. Each subject is only required to work with one of the three presentation styles. Hence the experiment is a between-groups, 1 by 3 factorial design.

The three presentation styles mentioned above are the different treatments of the independent variable. The dependent variables are (1) subjective preference and (2) advice effectiveness. The methods for measuring the two dependent variables are described below in the section on *Calculation of Scores*.

3.3 Hypotheses

Before the experiment was run, we hypothesized that the third method of presenting advice would perform significantly poorer (at the 10% level) than the other two methods in both subjective preference and advice effectiveness because it interrupts the user's train of thought. The participants were also hypothesized to prefer the first method rather than the second method (also at the 10% level of significance). However, no conjectures were made on how the two methods would compare with each other in the context of advice effectiveness; the second method gives the users more control over the system but the first method has the advantage of allowing the users to glance at a notice while they are waiting for the system to complete a long process.

3.4 Subjects

Twenty-four students from an undergraduate *Introduction to Computing* class were used as subjects. The class is mainly for non-computer science majors and teaches the students to use only applications on the IBM personal computers. Hence, all the subjects were familiar with DOS but had no experience with UNIX. The subjects were highly motivated students; almost all of them volunteered for the experiment because they wanted to learn more about computers.

3.5 Materials

The computers used were DEC2000 workstations running UNIX and X Windows. The software for the experiment is described in the previous section.

Each subject was given a set of paper printouts consisting of five parts:

(1) experimental consent form;

(2) questionnaire on the computer background of the subject;

(3) tutorial;

(4) list of tasks; and

(5) post-experiment questionnaire.

3.6 Procedure

Students who signed up for the experiment were permitted to choose the date and time most convenient for them to take part in the experiment. They were randomly assigned to the different presentation styles: the first subject was assigned to the first presentation style; the second subject to the second style; the third subject to the third style; the fourth subject to the first style again and so on.

As subjects arrived and settled down, they were given the materials listed above. A brief explanation of the experiment was provided before the subjects were asked to sign the consent form. They were then asked to complete the pre-experiment questionnaire and directed to start on the tutorial.

The tutorial was designed to teach only the basics of the file system, five simple UNIX commands (`cd`, `ls`, `cat`, `cp`, `rm`) and the concept of output redirection. Subjects were allowed to take as much time as they needed for the tutorial. The tutorial session was not monitored and no computer-generated advice notices were given. We helped the subjects during the tutorial by correcting their mistakes and/or misconceptions.

After the subjects indicated that they were done with the tutorial, the UN program was started and the appropriate directory was set up. The subjects were given six tasks but were told that it is not necessary to do them in the given order or even to complete all of them. The tasks are worded such that only the goals are stated; the choice of commands necessary to reach the goals was left to the subjects. However, minor hints on how to go about doing things were provided just in case the subjects could not relate the goals to the material taught in the tutorial.

Interaction with the subjects during the experiment was kept as limited as possible. However, sometimes it was considered necessary to clarify some of the tasks. At times, the meaning of error messages had to be explained to the subjects. Nevertheless, no UNIX information not already given in the tutorial was volunteered.

Finally, before the subjects left, they were asked to complete the post-experiment questionnaire.

3.7 Calculation of Scores

The post-experiment questionnaire (Appendix A) is made up of four parts. Only the first part is used in the measurement of subjective preference. This part of the questionnaire consists of eight short questions. To answer these questions, the subject has to choose a number between 1 and 7 where 1 means 'Yes', 4 means 'Neutral' and 7 means 'No'. Subjective preference is calculated using the following formula: $SP = a + b + c + d - e - f - g + h$ where a refers the subject's answer to question (1a), b refers the subject's answer to question (1b), etc. Hence, a lower score means that the subject prefers his/her presentation style more.

The calculation for advice effectiveness is much more complicated. Four main rules for assigning values were followed. For every advisory situation that was detected and validated:

(1) if the advice is relevant to a task that the subject was working on *after* the advice notice had been detected, but *not* applied, one point is taken off the score.

(2) if the advice is relevant to a task that the subject was working on *after* the advice notice had been detected, and applied correctly, one point is added to the score.

(3) if the advice notice is *not* relevant to any task that the subject was working on, and used *incorrectly*, half a point is taken off.

(4) if the advice notice is *not* relevant to any task that the subject was working on, but used *correctly*, half a point is added.

In contrast to subjective preference, a higher *effectiveness* score means that the corresponding presentation style is better.

3.8 Results

		Advice Effectivenesss	Subjective Preference
Group's **mean** and *standard deviation*	Notices in Separate Window	**-2.9** *4.3*	**-6.5** *8.8*
	Notices only on Request	**-0.7** *6.1*	**-6.0** *9.9*
	Notices posted Immediately	**3.0** *4.1*	**-2.5** *5.0*
One-way ANOVA	F	2.94	0.57
	df	2, 21	2, 21
	p	0.08	0.57
Two-sample t-tests df=14 (**t-value** and *probability*)	Separate Window vs On Request	**-1.36** *0.19*	**-0.11** *0.92*
	On Request vs Immediate	**-0.89** *0.39*	**-0.89** *0.39*
	Immediate vs Separate Window	**2.82** 0.01	**1.12** *0.28*

Table 1: Experiment 1 statistics

The *SPSS* statistical package was used to analyze the data collected. The one-way analysis of variance (ANOVA) technique was applied twice to examine if different presentation styles had any overall effects on subjective preferences and advice effectiveness. Two-sample t-tests were then done to study differences between individual presentation styles. The results of the tests are summarized in table 1.

Since aggregate subjective preference was not statistically different, each individual component of the *preference* score was analyzed. Table 2 summarizes the mean and standard deviation of each of the test questions. ANOVA and t-tests were performed on the individual question scores but all the results -- with the exception of one -- proved to be insignificant. The only significant result found was for the seventh question: “Are the messages disruptive?” Two one-directional t-tests were performed (the null hypothesis is: the third presentation style is not more disruptive than the first and the second presentation styles) on this question and the results are shown in table 3.

Presentation style	Do you like Unix?	Do you like the advice?	Do you find the advice helpful?	Do you like the advice messages?	Are the messages misleading?	Are the messages threatening?	Are the messages disruptive?	Do you like the idea of a computer providing advice?
1) display advice in separate window	**2.8** *2.2*	**2.1** *1.8*	**2.5** *2.1*	**2.5** *1.4*	**5.5** *2.0*	**6.8** *0.5*	**5.6** *1.1*	**1.5** *0.8*
2) prompt user and wait for *hello* command	**2.6** *1.8*	**2.2** *1.7*	**1.9** *1.0*	**2.0** *1.3*	**5.4** *1.6*	**5.5** *2.0*	**5.4** *2.2*	**1.5** *1.1*
3) display advice immediately in the user's window	**3.2** *1.2*	**3.1** *1.6*	**2.4** *0.9*	**2.4** *0.7*	**4.8** *1.7*	**6.2** *1.4*	**3.9** *2.0*	**1.2** *0.5*

bold numbers represent the mean.
italic numbers represent the standard deviation.
A smaller value represents a more positive answer to the question: 1 - Yes, 4 - Neutral, 7 - No.

Table 2: Statistics on answers to questionnaire

	display advice in separate window	prompt user and wait for command
display advice immediately in user's window	t-val = 2.22 prob = 0.022	t-val = 1.44 prob = 0.086

df = 14 for both tests; one-tail probability is given

Table 3: Two sample t-test results for “Are the messages disruptive?”

3.9 Analysis

The results obtained from the experiment did not agree totally with our initial hypotheses. The ANOVA tests showed that different presentation styles affected *advice effectiveness* ($F(2,21)=2.94$, $p = 0.08$) but not *subjective preferences* [2] ($F(2,21)=0.57$, $p = 0.57$).

Surprisingly, the third presentation style -- which was hypothesized as the worst of the three styles -- turned out to be the best in the context of advice effectiveness; it is significantly better that the first style ($t(14)= -2.82$, $p = 0.01$) though not significantly better than the second style ($t(14)= -1.36$, $p = 0.19$).

The initial belief that the third presentation style is more disruptive was supported. The experiment showed significantly that the subjects judged the third style more disruptive than the second style ($t(14)= 1.44$, $p = 0.09$), and the first style ($t(14)= 2.22$, $p = 0.02$).

The above findings suggest that it is proper to disrupt the users in order to give them useful advice. However, from the subjects' comments and observations made during the experiment, other explanations are possible. It is possible that the first and second presentation styles did not perform as well because:

(1) the subjects did not notice changes in the advice window / subjects did not notice the change in the prompt.

[2] The insignificant ANOVA result could be because of the way the *preferences* score was calculated.

(2) the subjects were not familiar with advisory facilities and hence unwilling to experiment with ours.

3.10 Subjects' comments and observations

From personal observations and the comments collected, we learned that each one of the three presentation styles has its problems.

Problems with the first style (separate window)

The main disadvantage with the first presentation style (notices displayed in separate window) is that the subjects do not seem to notice the advice window. Subject 22 wrote, "Sometimes (actually most of the time) I missed [the notices] because I was concentrating on what I was doing . . ." Subject 7 suggested having "a sign on the middle screen saying that there is advice to be given."

Most of the subjects were observed not using the buttons to read previous advice notices or to organize the notices. Hence, when they looked at the advice window, they only noticed the latest notice which may be concealing a more relevant notice.

Problems with the second style (change of prompt)

The subjects seem unwilling to use the UN with this presentation style. One subject was observed working through the entire experiment with a smiley face in front of every prompt. When asked why he did not type hello and read the advice notices, he replied that he knew he was doing the right things and did not believe that the advisory facility has anything relevant for him.

Another problem is the fact that users have to stop whatever they are doing to read an advice notice. Subject 8 commented that the advice notices were "above/separate from what I was doing. [So] I had to look at it [and] then go back to what I was trying to do."

Problems with third style (immediate display)

The biggest problem with this style is that it is very disruptive. Subject 15 said this about the style: “In the beginning, the advice was helpful. But often the same advice showed up even though I had understand it and it was disruptive . . .” Subject 18 said that “I dislike the fact that it scrolls off the screen information previously appearing on it that I had wanted to see . . .”

Another problem is that subjects could not re-read previous advice notices. Subject 9 wrote “Sometimes they are not available when you need [the messages], . . . they are difficult to recall once they are gone.”

General problems

The concept of advisory facilities is relatively new and it is understandable that the subjects do not fully comprehend what UN does. A few of the subjects were observed trying to use the advisory facility only when they were stuck with a problem. They appeared frustrated when they did not see the advice notice they were hoping for. One of the subjects suggested incorporating error recovery advice into the system.

Several subjects suggested that a help key/menu would be better. Subject 10 commented that he does not “like the advice to come after I needed them, it would be easier to have a help menu.” Subject 3 would like to “have a help key to look for the advice rather than just waiting for advice to come up.”

3.11 Improvements

These comments suggest that all three presentation styles studied can still be improved. The first style should become more effective if users are alerted (in a non-disruptive way) whenever a new advice notice appears in the advice window. The third presentation style could be made less disruptive by limiting the number of advice notices given in any time period; notices should also not be repeated unless they are really relevant to what users are doing. And both the first and the second styles will benefit if the advice notices previously seen can be arranged in order of relevance.

A possible improvement might be to combine presentation styles. For example, an notice may be displayed using the third presentation style when it is first deemed appropriate. If another situation requiring the same notice is detected, the first or second presentation style may be used instead. Subject 15 had this suggestion: ". . . have the advice on-screen in the window for beginners. But for intermediate users it would be better [to have the advice messages] on-screen outside the window, and for advanced users, just a notation that advice is now available [should be enough]." This interesting suggestion agrees with what was discovered in this study, and should be studied further.

4. EXPERIMENT 2: FEEDBACK DELAY

4.1 Independent Variable

The independent variable in this experiment is the delay in feedback. Three treatments of this variable are studied.

<u>Immediate feedback</u>

The presentation style of this treatment is a combination of the second and third styles of the previous experiment[3]. All the advisory situations are first judged for their level of importance. Advice notices considered more important are presented using the third style (immediate display) while the rest are presented using the second style (change of prompt). However, all the advice notices (including those displayed immediately) are kept on an internal list. The user is allowed to refer to the list at any time by using the `hello` command.

<u>Feedback at end of session</u>

```
Shortcut...

there is a command to rename a file. The command is `mv', short
for `move'.

To rename the file apple to orange, simply type `mv apple orange'.


                                        ...message 1 of 1



All messages have been seen.

Type    <F>  go forward;          <B>  go backward
        <Q>  quit
```

Figure 6: Sample advice notice

In this treatment, advisory situations are detected and validated as normal.

---

[3] The first style of the previous experiment (messages in a separate window) is not used because the subjects are likely to do their work on terminals that do not support multi-windowing.

However, all the advice notices immediately go onto an internal list. Users are not aware of the existence of any notices until they try to quit the session. They cannot even access the advice notice with the `hello` command. When they logout, they are placed in a routine to read advice messages. (The routine is similar to that used by the second style of the first experiment except that users are not given the *Remove*, *Mark*, and *New* options. See figure 6.)

<u>No feedback (the control group)</u>

A control group is included to test whether the differences between the two previous treatments (if any) are caused by UN. To ensure that system performance is the same among all three groups, subjects in the control group were given a version of UN that had the *presenter* disabled.

4.2 Dependent Variables and Hypotheses

Four dependent variables are studied in this experiment.

<u>Subjective preference</u>

At the end of the experiment, the subjects were asked the following questions.

a) Do you like UNIX?

b) Do you consider yourself a proficient UNIX user?

c) Do you like the idea of a computer providing advice?

d) Do you like the advice messages?

e) Do you find the messages helpful?

The answers are numbers from 1 to 7 where 1 means “Yes, very much”; 4 means “Neutral”; and 7 means “No, not at all”. I performed one-way ANOVAs on the first 3 questions. It is not possible to do ANOVAs on the last 2 questions as the control group did not get any advice notices. However, two-sample t-tests were attempted on these questions. Two-sample t-tests were also done on the results of the first three questions. These tests study differences between pairs of groups.

We believed that the participants who are given advice would like UNIX better and consider themselves better users of UNIX (at 10% level of significance). Users of the immediate feedback method were hypothesized to like the advice notices and find them more helpful that those users using the delay feedback method.

Commands familiarity

At the start of the experiment, the participants were required to list (in five minutes) all the commands (and options) they could remember at that moment. They were required to do the same at the end of the experiment. Differences between each subject's lists are attributed to changes in the users' mental model. The difference in the length of each subject's lists was calculated. An one-way ANOVA and three two-sample t-tests were performed on the data.

The initial hypothesis was: the participants who are given UN notices would gain familiarity with more commands than those without the notices (10% level of significance).

Commands used

The user modeling component of UN records the date that a user uses a command that is in the domain model. From this information, we counted the number of different commands[4] (and their related options) used by each participant during the experiment[5] . An one-way ANOVA and three two-sample t-tests were performed on this number.

It was initially believed that users of the immediate feedback method would use more different commands than users of the delay feedback method. Users of both these methods were also hypothesized to use more commands than the control group.

Inefficient command usage

4 Commands that are not in the domain model are not counted.

5 Because I represented each date as a special code, I found it much easier to count only those commands used after the 19th of June.

Among all the advisory situations, there are a few that will only be detected if a user performed a very specific sequence of non-optimal commands. UN was modified to count the number of times these advisory situations were detected. A high value implies that the participant was not following the advice notices (if notices were provided). An one-way ANOVA and three two-sample t-tests were also performed on this variable.

The hypothesis for this variable was: immediate feedback would result in more notices being followed. Participants in the control group were believed to have the highest detection count as they have no notice to follow.

4.3 Procedure

The lecturer of a Summer 1992 undergraduate computer science class gave us the last fifteen minutes of the first class to explain my experiment and solicit volunteers. Almost the entire class signed up, hoping that the system would help them with the course. The volunteers were asked to sign a consent form and answer a short questionnaire.

The necessary changes[6] to the students' accounts were made before the accounts were assigned. Two accounts (numbers 2 and 3) were left unchanged for those students who decided not to participate in the experiment. The rest of the accounts (numbered 4 to 38) were assigned one of the three possible treatments in a modulo-3 fashion. Accounts 4,7,10 and so forth were given the first treatment (immediate feedback); accounts 5,8,11 and so forth were given the second treatment (feedback at end of session); the rest of the accounts had no feedback.

Each student was randomly assigned to an account by the teaching assistant. The non-volunteers were given their accounts first while the volunteers were handed a set of materials that included their account number, the temporary account password and a sheet

6 Certain system parameters were changed using the *.cshrc* file; the *.login* file was modified to execute a version of my program.

of paper explaining how to use UN. The explanations varied depending on the version of the program that the student would be using. Due to the random nature of the account assignment, accounts 6 and 32 were not assigned.

Because most of the subjects were new to UNIX, the teaching assistant and the lecturer asked that the UN program be disabled for the first two weeks while they taught the students the fundamentals of UNIX. They felt that this was necessary as they did not want the students to think that UN (with its possible side effects) was a standard part of UNIX. Hence, data collection for the experiment started only on the fifteenth of June.

The experiment was run over a period of about one and a half months. During this time, the data necessary for the calculation of the dependent variables were captured automatically by the program. At the end of the experiment, the subjects were again asked to answer a short questionnaire .

4.4 Subjects

The subjects in the Summer experiment were students taking the undergraduate computer science course CMSC112 (*Computer Science I*). They are first year students with almost no experience in UNIX. A majority of the volunteers could not list a single UNIX command on the first day of class. Thirty-three students volunteered for the experiment when first approached.

During the experiment, one of the subjects (account number 30) dropped the course and another two subjects (numbers 5 and 35) opted out of the experiment by undoing the changes that I had made in their accounts. Of the remaining subjects, we were able to get only sixteen to answer the end-of-experiment questionnaire. The rest did not reply to our electronic mail nor attend the last regular class.

By pure coincidence, ten of the sixteen who answered the final questionnaire were from the group given the first treatment (immediate feedback), four were from the third group (no feedback), with only two subjects from the second group (feedback at end of session). This unfortunately makes it more difficult to draw conclusion for the first two dependent variables (subjective preferences and command familiarity).

The data for the other two dependent variables (commands used and advice followed) is complete as it is collected whenever the volunteers logged in. However, the data of one of the subjects was rejected because (s)he logged less than two hundred commands during the entire course (subject 21 used only 57 commands). The cutoff value of two hundred was decided before the experiment was conducted. Hence, data of twenty-nine subjects was used in the calculations of the last two dependent variables.

4.5 Results

The twenty-nine subjects were monitored using a total of 48401 commands. During this time, a total of 2665 advisory situations were detected **and** verified. However, it is unknown how many of these notices were actually read.

		Do you like UNIX?	Do you consider yourself a proficient UNIX user?	Do you like the idea of a computer providing advice?	Commands familiarity	Commands used	Inefficient Usage
Group's **mean** and *standard deviation*	Immediate feedback	**3.4** * *1.4*	**4.6** * *1.8*	**2.4** * *1.9*	**7.7** *3.7*	**17.1** *3.6*	**17.7** *13.2*
	Feedback at end of session	**1.5** *0.71*	**3.5** *0.71*	**3.5** *0.71*	**9.0** *1.4*	**16.4** *3.0*	**21.3** *16.7*
	No feedback	**1.5** *0.58*	**3.5** *0.58*	**1.8** *1.0*	**6.8** *4.2*	**16.3** *4.9*	**35.5** *27.3*
One-way ANOVA	F	4.92	0.97	0.75	0.26	0.13	2.23
	df	2, 13	2, 13	2, 13	2, 13	2, 26	2, 26
	p	0.03	0.40	0.49	0.78	0.88	0.13
Two-sample t-tests (**t-value**, *df* and probability)	Immediate feedback vs Feedback at end ...	**1.89** *10* 0.09	**0.84** *10* 0.42	**-0.78** *10* 0.45	**-0.48** *10* 0.64	**0.44** *19* 0.67	**-0.56** *19* 0.58
	Feedback at end ... vs No feedback	**0.00** *4* 1.00	**0.00** *4* 1.00	**2.24** *4* 0.09	**0.70** *4* 0.52	**0.10** *15* 0.92	**-1.31** *15* 0.21
	No feedback vs Immediate feedback	**-2.91** *11* 0.01	**-0.96** *11* 0.36	**-0.55** *11* 0.60	**-0.84** *11* 0.42	**-0.44** *18* 0.67	**1.96** *18* 0.07

	# of Subjects
Immediate Feedback	10
Feedback at end . . .	2
No Feedback	4
TOTAL	16

	# of Subjects
Immediate Feedback	12
Feedback at end . . .	9
No Feedback	8
TOTAL	29

* A smaller value represents a more positive answer: 1 - Yes, 4 - Neutral, 7 - No.

Table 4: Experiment 2 statistics

Using the *SYSTAT 5* statistical package on the Macintosh machines, one-way ANOVAs were preformed on the *advice followed*, *commands used* and *commands familiarity* dependent variables as well as on the answers to the first three questions on the end-of-experiment questionnaire. Two-sample t-tests were also done on all these variables and questions to study the relationship (if any) between any two treatments (table 4).

ANOVAs were not performed on the last two questions on the end-of-experiment questionnaire ("Do you like the advice messages?", "Do you find the messages helpful?") because these questions are meaningless to those subjects given the no-feedback treatment. We performed t-tests using only the answers by the groups given feedback but were unsuccessful because of insufficient data (the feedback-at-end-of-session group has only two data points).

4.6 Analysis

As expected, few statistically significant differences were found in the results. The expectation arises because this type of experiment has many uncontrollable factors. The difficulties are described below.

The only significant ANOVA result is from the question "Do you like UNIX?" ($F(2,13)=4.92$, $p = 0.03$). This shows that the feedback delay factor does affect the users' feelings towards UNIX. However the data collected directly contradict the initial hypothesis: the group given immediate feedback likes UNIX significantly less than the group with delayed feedback ($t(10)=1.89$, $p = 0.09$) and the group with no feedback ($t(11)=2.91$, $p = 0.01$).

Another significant result is that the group with immediate feedback performed significantly better -- as measured by *inefficient command usage* -- than the group with no feedback ($t(18)=-1.96$, $p = 0.07$). This result agrees with the initial hypothesis. The subjects from the group with no feedback were given no help from UN and thus produced more occurrences of inefficient usage. The group with delayed feedback performed poorer than the group with immediate feedback but better than the group with no feedback. However these results are not significant at the 10% level.

The final significant result is from the second t-test of the question "Do you like the idea of a computer providing advice?" Surprisingly, the group with no feedback likes the idea significantly more ($t(4)=-2.24$, $p = 0.09$) than the group with feedback at end of session. We do not have an explanation for this result. It is very possible that the small number of valid data points in these two groups affected the result.

4.7 Subjects' Comments and Conclusions

The subjects have two main comments to make about the system. First, some of them noted that they hardly saw the program in action. Subject 7 wrote "I didn't get to use your program very much; but the few times I saw it, it gave me some helpful advice." This deficiency arises because of the limited number of notices that are currently handled by UN. If the subject has sufficient experience with UNIX, none of the advisory situations would be detected and hence UN will have no opportunity to activate its *presenter*.

The second comment is from subjects who were given immediate feedback. Some of them found the advice notices too repetitious. Subject 34 commented "It helped me several times and showed me easier ways of doing things, but some messages were repeated too much." It could be this reason (too repetitious) that led the first group (immediate feedback) to like UNIX less than the other two groups.

However, we received quite a few favorable comments. When advice notices do appear, the subjects seemed to appreciate them and learn from the notices. This "learning" result showed up in the analysis of the *inefficient command usage* variable.

This experiment demonstrates that immediate feedback helped users to improve their UNIX skills but it may have caused them to dislike UNIX.

4.8 Limitations

In conducting any complex experiment with computer training effects over a period of weeks, there are many factors that may affect the outcome.

First of all, it is not possible to control what the subjects do when they logged on. Some of the students used their accounts to do other activities (like `talk`-ing to other computer users) in addition to their course work. In other words, some subjects used their accounts more intensively than others. This would have affected the type and frequency of advisory situations that are detected. Subjects who used their accounts solely for their course work will not appreciate UN as much as the other users.

It is also difficult to judge if UN does teach anything. Even if a subject learns a new concept or command during the experiment, it is not possible to conclude that the subject indeed learned it from UN and not somewhere else (like a friend or a book). The only way to be sure of UN's effectiveness is to restrict the subjects' interaction with the outside world. However, this is impossible due to of the length of the experiment (one and a half months).

5. CONCLUSIONS

UN shows that an effective advisory facility can be developed for a complex system without the problems of error-recovery help. Instead, UN guides the users in their learning process, slowly expands their mental models and encourages exploration of the underlying system.

However, there are many questions that must be answered before such help facilities can be put to widespread use. We have tried to answer two of those questions -- "how to present the advice notices?" and "when to present the notices?" -- via experimentation. The two experiments suggest the following guidelines for designers of new advisory facilities:

1) Notices should be posted such that they are easily noticeable by the user, even though this style may be very disruptive.
2) Notices should be posted as soon they are deemed appropriate.

Most importantly, the experiments showed that people like the concept behind UN. Twenty-three out of the twenty-four subjects from the first experiment (presentation style) said that they like the idea of a computer providing advice; the only remaining subject was neutral. And of the sixteen who completed our end-of-experiment questionnaire in the second experiment, only one student replied in the negative while another was neutral; the rest of the subjects like the idea.

5.1 More Experiments

More experiments are needed to better understand the design issues and confirm our intuitions. The most important but most difficult experiments to conduct will be on advice presentation:

1) How does users' level of expertise affect their opinions of the different presentation styles in the first experiment? Do beginners prefer a particular style while experts prefer another style? Subject 15 of this experiment suggested a way of assigning the different presentation styles to users of different expertise (see section 3.12). This suggestion is worth looking into.
2) Are the opinions about presentation styles (first experiment) affected by the tasks the users do? For example, if the subjects were asked to use commands that are slow to execute, the subjects may have more idle time and may glance at the advice window more often. Thus, the subjects may have a more favorable opinion of the 'Separate Window' presentation style.
3) Is it really desirable to repeat advice? How many repetitions are acceptable before users become annoyed? During the design phrase, we felt that repetitions are necessary as they remind users of the notices they had received. We hoped that the constant reminders would discourage improper usage and encourage correct command usage. Is this intuitive thinking correct?

5.2 Future Directions

UN is still very much in its infancy. It is currently useful only to people who are starting to learn UNIX. Once these novices have learned the basics of UNIX, they will have little use for the advice notices provided by UN. To make UN more useful, more advice notices have to be added. This can be done with more *detectors* and a better *verifier*.

The method of providing help advanced by UN need not be restricted to the UNIX operating system. Any complex system that has lots of options and many variety of users should become more popular if it has this type of help facility.

An example of another operating system that may benefit from this form of help facility is DOS. DOS is quite similar to UNIX and thus it should be easy to modify UN such that it will provide advice notices for DOS. The most difficult part of this task is in the development of another method for monitoring the user.

The help-via-advice concept advanced by UN can also be applied in systems with graphical interfaces (for example, Macintosh machines). A new monitoring method and parser will be needed as menu selections with the mouse (or other pointer devices) must be monitored and differentiated. However, if we treat each menu selection or each mouse action as an individual command, it is possible to use UN's method of detecting and verifying advice notices.

With graphical interfaces, more styles of presenting the advice notices are possible. The notice-in-separate-window style considered in the first experiment should be studied again. Another style worth examining is to have a blinking icon or a beep to indicate the presence of an advice notice.

In general, to develop a similar advisory facility for another system, studies must first be done to determine common errors and collect useful advisory situations. The designer of the new help facility should then decide on a method of detecting, verifying and presenting notices. Designing and developing help facilities of this form is not easy. But as more studies are done and more experiments conducted, the task will be progressively simplified.

REFERENCES

1. Nickerson, R. S. (1981). Why interactive computer systems are sometimes not used by people who might benefit from them. *Int. J. Man-Machine Studies 15*, 469-483.

2. Anderson, J.R. (1985). The lisp tutor. *Byte 10*(4), 159-175.

3. Johnson, M.L., and Soloway, E. (1985). PROUST: an automatic debugger for Pascal programs. *Byte 10*(4), 179-190.

4. Wilensky, R., Arens, Y., and Chin D.N. (1984). Talking to UNIX in English: an overview of UC. *Communications of the ACM 27*(6), 574-593.

5. Carroll, J.M., and McKendree, J. (1991). Interface design issues for intelligent advisory systems. In Kent, A. and Williams, J.G. eds. *Encyclopedia of Computer Science and Technology*, 111-142. New York, NY: Marcel Dekker, Inc.

6. Carroll, J.M., and Carrithers, C. (1984). Training wheels in a user interface. *Communications of the ACM 27*, 800-806.

7. Owen, D. (1986). Answers first, then questions. Norman, D.A., and Draper, S.W., eds. *User centered system design: new perspectives on human-computer interaction*, 361-375. Hillsdale, NJ.

8. Hill, W.C., and Miller, J.R. (1988). Justified advice: a semi-naturalistic study of advisory strategies. *Proceedings of the CHI'88 Human Factors in Computer System*, 185-189.

9. Hill, W.C. (1989). How some advice fails. *Proceedings of the CHI'89 Human Factors in Computer Systems*, 85-90.

10. Carroll, J.M., and Aaronson, A.P. (1988). Learning by doing with simulated intelligent help. *Communications of the ACM 31*(9), 1064-1079.

11. Corbett, A.T., and Anderson, J.R. (1990). The effect of feedback control on learning to program with the Lisp Tutor. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, 796-803.

12. Corbett, A.T., and Anderson, J.R. (1989). Feedback tutoring and student control in the LISP Intelligent Tutoring System. *Artificial Intelligence and Education*, 64-72.

APPENDIX A: Experiment 1 (Presentation style) post-experiment questionnaire

Before-you-leave questionnaire

We know you are tired and would like to get out of here. But before you do so, we would like your opinions on a few matters. There are no right answers to these questions.

	Yes			Neutral			No
1a) Do you like UNIX?	1	2	3	4	5	6	7
1b) Do you like the advice?	1	2	3	4	5	6	7
1c) Do you find the advice helpful?	1	2	3	4	5	6	7
1d) Do you like the advice messages?	1	2	3	4	5	6	7
1e) Are the messages misleading?	1	2	3	4	5	6	7
1f) Are the messages threatening?	1	2	3	4	5	6	7
1g) Are the messages disruptive?	1	2	3	4	5	6	7
1h) Do you like the idea of a computer providing advice?	1	2	3	4	5	6	7

2) What do you like or dislike about this method of presenting advice? Why?

3) How would you improve on the advice presentation?