Carnegie Mellon University
Research Showcase @ CMU

Department of Psychology | Dietrich College of Humanities and Social Sciences

November 2008

The Curriculum and the Learning Environment: A Componential Analysis of the Learning Curve

Kevin A. Gluck
Carnegie Mellon University

Marsha C. Lovett
Carnegie Mellon University

John R. Anderson
Carnegie Mellon University

Valerie J. Shute
GKIS, Inc.

Follow this and additional works at: http://repository.cmu.edu/psychology

Recommended Citation
.

This Article is brought to you for free and open access by the Dietrich College of Humanities and Social Sciences at Research Showcase @ CMU. It has been accepted for inclusion in Department of Psychology by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

Running head: COMPONENTIAL ANALYSIS OF THE LEARNING CURVE

The curriculum and the learning environment: A componential analysis of the learning curve

Kevin A. Gluck, Marsha C. Lovett, and John R. Anderson
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Valerie J. Shute
GKIS, Inc.
7719 Adagio Ave.
Houston, TX 77040 USA

Please address correspondence to: Kevin Gluck
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213 USA
Email: kgluck@cmu.edu
Phone: (412) 268-8117

Abstract

This paper presents a quantitative assessment of the importance of adaptation to the learning environment as a component of the learning curve in performance data from a computer-based tutor. In Experiment 1, verbal protocols are used to investigate the nature of changes in low-level interactions that take place during learning with a computerized tutor called *Stat Lady* (Shute & Gluck, 1994). The data show consistent behavioral changes in the distribution of attention, which account for a substantial portion of the learning curve, independent of error rates. These changes primarily are decreases in the verbalization of on-screen text, although the elimination of interface confusion also contributes to the efficiency gain. Experiment 2 tests the generalizability of the results in a larger population of learners. It is shown that adaptation to the learning environment accounts for a comparable proportion of the learning curve in this new population. More than half of the learning curve could be accounted for by these changes in low-level interactions. These results suggest that more accurate learning models should include a representation of increasing knowledge of the instructional environment as the model interacts with that environment. An ACT-R (Anderson & Lebiere, 1998) model is provided that reproduces the qualitative and quantitative data from the verbal protocol participants. The model reproduces these behaviors via (1) the acquisition of declarative knowledge for the structure of the problem scenarios, and (2) subsymbolic procedural tuning for more efficient goal completion.

The curriculum and the learning environment: A componential analysis of the learning curve

Card, Moran, and Newell (1983) made the point that the study of the use of computers is an especially important applied research topic, due to the proliferation of computers in every aspect of contemporary society. A special case of computer use, which is becoming increasingly prevalent in education and training settings, is the interaction of students with a computer-based tutoring system. The primary research concern for those who design and deploy these systems is typically to maximize the degree to which students acquire the curriculum the tutor is intended to teach. There is, however, another sort of learning that is going on simultaneously - the learning of the learning environment in which the curriculum is presented.

Conventional wisdom tells us that people learn something about the computer interfaces[1] they use and, with experience, become more facile at navigating through them. There is little that is new in such a claim. However, systematic data that document the development of such interface knowledge and the impact it has on performance are considerably less prevalent than this assumption. That is the contribution offered here. This paper describes two empirical investigations dedicated to producing a quantitative assessment of interface learning in a particular computer-based tutoring system.

Why Use a Computer Tutor?

There are a number of different types of computer-based software applications that we could have chosen to study. Text editors, spreadsheets, programming languages, and even games are all valid contenders for the study of interface learning. Why pick a tutoring system, where one also has issues of curriculum learning to worry about? First and foremost is that, as cognitive psychologists and instructional designers,we are interested in issues of curriculum learning, and a desire to better understand the processes and products of learning runs deep.

Second, computers are often (and increasingly) used as instructional tools, intended to enhance the learning of the curriculum objectives (Lajoie & Derry, 1993). In some cases, these computerized learning environments lead to improved performance, suggesting that students have learned something from the experience (e.g., Anderson, Corbett, Koedinger, & Pelletier, 1995; Lesgold, Eggan, Katz, & Rao, 1992; Shute, 1995). Two measures are used to assess learning in these studies. The most universal is learning gain, generally measured as pretest-to-posttest improvement, although there are other variants. To the extent that there is evidence for improvement on the posttest, there is evidence for acquisition of the curriculum.

Sometimes researchers include another dependent measure, as well, which is problem solving time. The claim is that, as knowledge of the curriculum grows stronger, problem solving time will decrease (e.g., Anderson, 1993; Anderson, Conrad, & Corbett, 1989). This habit originates from the tradition in cognitive psychology of using time as an indicator of degree of learning, and plotting this change over time. The resulting graph is best described as a power law learning curve (Newell & Rosenbloom, 1981). However, in the context of problem solving with a computer-based tutor, the questions of how much students are "learning the curriculum" versus "learning the environment," and what effect this has on interpretations of the learning curve, often go unasked. In the absence of attention to these questions, decreases in solution times are generally assumed to result from the former. This assumption is almost entirely untested. The research described here provides an example of how careful analysis of verbal protocols and performance data can reveal components of the learning curve in addition to that part which is attributable to the development of cognitive skill.

A Growing Need

Almost a decade ago, a paper was published which foreshadowed the need to more

deeply explore the composition of the learning curve in the context of learning from computer tutors. In their paper on student learning with the Lisp Tutor, Anderson, Conrad, and Corbett (1989) left open the possibility that some portion of the improvement they saw in coding time actually reflected mastery of the interface. They found that two important variables predicted students' solution times. One was related to the amount of prior practice. Anderson et al. represented the acquisition of skill in Lisp coding as the acquisition of production rules for that skill, and their analysis showed that the number of prior production firings (amount of prior practice) was a good predictor of future problem solving time.

A second variable that predicted problem solving time was how far a student had progressed through the curriculum (measured by lesson number). They made the following statement regarding this latter result:

> The effect of lesson number … may just reflect an increased familiarity with the tutor interface. The fact that the same variable shows up for old productions as for new productions suggests that at least part of the phenomenon is a matter of general interface learning. It is also the case that lesson number is not significantly related to error rate. This is further evidence that the effect may be an interface effect and not reflect any real proficiency in coding. (p. 484)

This acknowledgement of the role of "interface learning" suggests that focusing on curriculum learning alone does not portray the complete picture of student learning in computer environments.

In this paper, we will examine the performance improvements of students working in a computerized learning environment called Stat Lady, which teaches introductory descriptive

statistics (Shute & Gluck, 1994). There is a good deal of evidence from previous assessment studies involving the Stat Lady tutor that people using the system develop improved skill on the curriculum objectives (Shute, 1995; Shute, Gawlick, & Gluck, 1998). However, is the learning of curriculum objectives the only learning that is taking place?

Our goal in this paper is to take the analysis of learning from the Stat Lady tutor well beyond a cursory glance at learning gain and broad measures of problem solving time. We will provide a description of how students interact with this computerized learning environment and how the interactions change as students learn. In two experiments, the careful decomposition of student behavior will show quantitatively that interface learning can account for a significant portion of the learning curve.

As will be seen, consistent trends in verbalizations over practice opportunities suggest that learners quickly fine-tune their low-level interactions with the tutor in a manner that allows for more efficient problem solving. These adaptations at the level of interactions with the tutor account for a substantial portion of the change in problem solving time across practice opportunities. This paper describes those results, provides an interpretation in terms of the acquisition of declarative knowledge of the interface and the structure of the problem scenarios, and concludes with a discussion of the implications of these results for tutor design and student modeling.

Experiment 1

The goal of Experiment 1 was to collect data that would provide a rich picture of students' learning - both of the curriculum and of the interface - as they interacted with the Stat Lady tutor. In particular, we were interested in how students' attention to different parts of the interface changed as they gained experience with the tutor. One source of such data is verbal

protocols.

Verbal protocols have played a significant role in contemporary studies of learning in many different domains (Ericsson & Simon, 1993; Newell & Simon, 1972). One of the fundamental assumptions that underlies the use of verbal protocol data is that the verbalizations reflect some subset of what is currently held, or was very recently held, in working memory (Ericsson & Simon, 1993)[2]. It seems reasonable to propose that the particular subset of working memory contents that gets verbalized would be that portion that is, or was very recently, the focus of attention. Thus, the verbal protocols provide information on the distribution of attention across the Stat Lady interface, and we use this methodology in Experiment 1.

Method

Participants

A significant challenge in any verbal protocol analysis project is that it is an inherently time-consuming methodology. This reality, exacerbated by the length of the tutor, motivated us to limit the sample to four participants. These participants were explicitly chosen on the basis of different degrees of domain-relevant prior knowledge and divergent educational backgrounds, in order to increase the generalizability of our conclusions. Two of the participants, one male and one female, were graduate students in the Psychology Department at Carnegie Mellon, and had advanced mathematics, statistics, and computer science classes at the college and graduate level. The other two, both female, were staff people on the CMU campus, and were relative novices in this domain, as they reported no formal education beyond high school in math, statistics, or computer science.

Materials

Equipment. The tutor was administered on an H&D 486/100 personal computer with a

17" color monitor, extended keyboard, and a standard mouse. An audio recorder sat next to the keyboard. Additionally, an S-VHS camcorder, which was mounted on a tripod directly behind the participant, videotaped everything they did on the computer screen. Participants wore lapel microphones for the audio pickup to the camcorder.

Software. The tutoring system used in this study is the *Stat Lady Descriptive Statistics Tutor: Data Organization and Plotting Module* (Shute & Gluck, 1994) developed at Armstrong Laboratory's TRAIN Lab. The purpose of the tutor is to teach the skills of organizing a dataset and representing it in both tabular and graphical formats. There are a total of 77 declarative and procedural instructional objectives, which together comprise the tutor's curriculum. The curriculum was divided into five sections, each containing a subset of the 77 objectives. Each section involves about 15 minutes of instruction, followed by a series of five problem scenarios which test the student's skill on the curriculum objectives from that section. Figure 1 provides an overview of the structure of the Stat Lady tutor that may be useful in understanding the relationships among sections, scenarios, and curriculum objectives. The curriculum objectives are tested in the portion of the scenarios labeled "Problem Activities/Q's." The terms "Context" and "Number Factory" deserve further explanation, since they both occur in the scenarios, and it is in the scenarios where we will be looking for changes over practice opportunities.

The first thing that happens when the student picks a scenario is that text appears which establishes a problem-solving context. It offers a rationale for collecting some data. Here is an example of the context statement from one of the scenarios:

> Let's say you were recently promoted to Chief of Operations at a huge amusement park. You're concerned about the park's liability, because of a new rollercoaster named Screaming Death Rocket.
>
> In order to determine whether the new ride is too dangerous for continued use, you choose to make up a frequency distribution of the number of injury complaints that are related to the Death Rocket every

day for two weeks (14 days).

Go to the Number Factory and get data with the following parameters:

N = 14 days
Min = 0 injuries
Max = 9 injuries

Immediately after reading the context, the student goes to the Number Factory. The Number Factory is intended to represent the data collection process. It is necessary (as suggested in the example above) to enter the sample size and range of the data. The Number Factory then generates data for use in later problem-solving activities. For a better understanding of the tutor's interface, see Figure 2, which displays a screenshot of the Number Factory. Having acquired the requisite data, it is then "shipped" back to the problem-solving area, where work is done on the declarative and procedural curriculum objectives in that section (e.g., complete the worksheet, answer questions). This pattern holds across all scenarios. In this study, participants were required to complete four of the problem scenarios in each section, but within a section they could do them in any order they wished.

In addition to the tutor, participants completed a comprehensive pretest and posttest. Both tests were designed to assess every curriculum objective in the tutor twice, across different response formats, in order to provide a valid assessment of current knowledge and skill.

Design and Procedure

Participants were briefed on the procedure of the experiment and the rationale for verbal protocol collection. It was explained that verbal protocols are often used to determine what it is students are paying attention to and thinking about, and that for the duration of the study, if there was something they were paying attention to and/or thinking about, they should verbalize it. It was emphasized that this included reading text on the screen. They were told to read only what seemed natural for them to read, but if they were reading it, they should be verbalizing it.

Participants practiced giving verbal protocols using the standard warm-up exercises suggested in the Appendix of the Ericsson and Simon (1993) text on protocol analysis, namely the "Windows in the House" exercise and mental addition of two-digit numbers. It was made very clear during these practice opportunities that we wanted participants to verbalize exactly what they were thinking, so that they established the habit of doing so before starting the tutor.

The next activity was the pretest. All participants were required to complete the pretest during their first session. After that, they had to do at least one section of the tutor each day until done. On the final day, the posttest immediately followed completion of the last section of the tutor. Each participant provided concurrent talk-aloud protocols during the entire pretest, tutor, and posttest, and these were recorded on video and audio tape. An experimenter sat behind the participant and prompted him or her to “Please keep talking” whenever verbalization stopped for more than a few seconds. Participants required an average of about 8 hours to complete the study, spread out over 2 to 5 days.

Verbalization Analysis

Given the length of each participant's participation, as well as the density of verbal protocol data, it was necessary first to pare down the protocols into a manageable subset from which to begin the analysis. We decided to focus our efforts on sections 1 and 3. Section 1 is a fairly obvious choice, as it allows us to study changes in learner behaviors at their earliest point in exposure to the tutor. Section 3 was chosen based on the fact that pretest data indicate this section is one of the most difficult, and our participants showed more evidence of curriculum learning in this section than any of the others. Since we were interested in how student interaction with the tutor changed during the learning process, Section 3 was a good candidate for inclusion in the protocol analysis.

As this was the first protocol study involving this particular tutor, there was no *a priori* set of procedures in place for segmenting the protocols and no established rules for coding those segments. We used Participant 1's protocols from Section 1 as the testbed for creating segmentation rules and developing a coding scheme.

Segmenting. For the purposes of this study, a segment is defined as a verbalization that indicates a new focus of attention. Thus, each segment represented a verbalized change in cognitive process or a switch in the portion of the visual field (on-screen) to which the participant is attending. The latter is identifiable, for instance, as a change in the location of on-screen text that the learner is reading.

Coding. The coding categories were designed primarily for distinguishing the different kinds of activities students might engage in while working with Stat Lady. This helped us identify where their attention was while they used the tutor. In all, 54 mutually exclusive categories were developed and used in the coding, but they are not all relevant to the results presented in this paper. For current purposes, the critical coding categories are those related to attention to different parts of the screen. Examples of these are presented in Table 1.

To facilitate consistency and objectivity in the coding, and also to establish whether the general scheme was reliable, we hired a research assistant to code a subset of the protocols. After a training period, he coded 60% of the Section 1 and Section 3 protocols. We had inter-coder agreement on 90% of the 3,266 segments. Such high inter-coder reliability suggests that our coding procedure was robust and we can be confident that the segments were coded accurately.

Results

We will first examine the verbal protocol participants' results using coarse measures like pre-to-posttest improvements, as well as changes in scenario completion times and error rates

with practice. Then we will move on to show how attending in more detail to the way people use the tutor allows for insight into other kinds of learning taking place. Specifically, we will provide evidence for adaptive changes in participants' low-level interactions with the tutor and the substantial role this learning plays in improvement with practice.

Coarse-Grained Analyses

Pretest-to-Posttest Improvements. At the broadest level of analysis, we would be interested in whether our four protocol participants show any evidence of having learned the curriculum objectives. Test data show that overall they improved from a score of 76.9% on the pretest to 94.6% on the posttest - a gain of 17.7 percentage points. Thus, they did indeed acquire curriculum-related knowledge and skill while using this tutor.

As mentioned above, however, the verbal protocol data were analyzed strictly from sections 1 and 3. How much did they learn in these sections? The average pretest score for curriculum objectives relevant to just these two sections was 79.2%, and the average posttest score was 95.8% - a 16.6 percentage point gain that is quite comparable to the overall mean for the tutor.

Scenario Completion Times. One would expect that, since there is evidence of curriculum learning taking place, there should be some speed up in performance over practice opportunities. Table 2 contains mean scenario completion times averaged over sections 1 and 3. As expected, the verbal protocol subjects show substantial decreases in the time needed to complete the scenarios. These data are also graphed as the "Total Time" curve in Figure 3.

Error Rates. One possible explanation for this decrease in time is that participants are making fewer errors with practice, which would reduce the amount of time spent on each

subsequent scenario. Error frequency data are available in Table 2. Interpreting these error rates is challenging, of course, without some sense for the total number of possible errors a learner could make. If we assume, due to the design of the feedback in the tutor,[3] that the highest number of errors a person could make on any given test item is three, then the maximum number of errors in each scenario (averaged across sections 1 and 3) is 60. As is evident in Table 2, despite the fact that the average error rates from our protocol participants are low, their errors do decrease across scenarios. Thus, the slight but gradual improvement in accuracy across scenarios could account for some of the speedup observed in scenario completion times.

Error state time. It is one thing to note a correlation between error rates and overall completion time, but the issue of how much of the improvement in completion time across scenarios is directly attributable to decreasing error time is another question altogether. Fortunately, we had the videotapes at our disposal. Using the videotapes, we constructed error time profiles, which represent the amount of time participants spent in an *error state* during each scenario. *Error state* is defined as the period of time (a) starting when the participant first makes a comment that indicates confusion and starts down a path leading to an incorrect solution and (b) ending when the participant verbalizes a realization of the correct solution or starts down a path leading to the correct solution. Mean Error Time for the four protocol participants, averaged over sections 1 and 3, was 70 s, 61 s, 35 s, and 58 s for scenarios 1-4, respectively. If we then subtract Error Time from Total Time, we are left with time data on the error-free completion of scenarios, as shown in the second curve from the top in Figure 3. There still is a substantial decrease across scenarios (109 s), suggesting that even when improvements in accuracy of performance are accounted for, something else is being learned.

Peripheral activity time. To further decompose what explains the speedup across

scenarios, we can subtract from the scenario completion times those activities which really are "peripheral" to the curriculum objectives. Activities like reading the Context at the beginning of the scenario and going to the Number Factory might be considered peripheral.[4] Mean Peripheral Activity Times were 68 s, 59 s, 73 s, and 64 s for scenarios 1-4. Note that peripheral activity times do not show a consistent decrease. This is because our protocol participants almost always read the Context portion of each scenario thoroughly, and the time they spent in the Number Factory remained fairly constant. Time data with peripheral activities removed are displayed as the third curve in Figure 3, labeled "Curriculum Objective Time: Coarse." There is still a 105 s change in completion time from the 1st to the 4th scenario.

Fine-Grained Analyses

By subtracting out error time and peripheral activity time, we have slowly chipped away at the activities that are components of the learning curve, but there still remains substantial improvement (105 s) from the first to the last scenario. What additional changes in student performance, aside from increasing skill at performing the curriculum objectives, could explain this speedup?

"Lost in the Interface" Time. It turns out there are a handful of critical junctions in the completion of these scenarios where students are more likely to get "lost" in the interface and are not sure how to proceed. Often this occurs when a critical interface component, like a button, appears suddenly on the screen and the learner does not see it.

An example of this, from Section 1, comes after the learner fills in the frequency column of the distribution. The tutor requires that students fill in five cells of the frequency column correctly, then a Fill Frequency Column button appears below the table, which will complete the job when it is clicked. The complication is that students do not know beforehand that they only

are required to fill five cells themselves. When the button appears, the tutor locks the frequency column so that it is no longer possible to fill in any more cells. If the student does not see the new button, which happens with some regularity, confusion ensues and the student is not sure how to proceed. The student is “lost” in the interface, looking for a way to make progress, but often not seeing the right path immediately.[5] Other times, everything the student needs in order to proceed with a certain goal is available on the screen right from the start, but the student is not sure how to carry out the task set by the tutor, and the result is hesitation and inactivity. In such a case, the student is having difficulty interpreting, the first time through, how to implement the instructions in the tutor’s interface.

We tracked these instances of being “lost” and unsure how to proceed in both Section 1 and Section 3. Two dependent measures bear on this issue. The first one has to do with the verbalization of procedural confusion. Generally, the only thing learners did do when they were “lost” was to ask a question regarding how to proceed. When this happened, the questions were typically considered to be rhetorical, or at least the experimenter made an effort not to respond, in order to avoid interfering with the normal course of tutor use. We coded segments that indicated procedural confusions and tallied them up for each participant in each scenario. Our protocol participants averaged 2.6, 1.5, .9, and .5 procedural questions in scenarios 1- 4. Note the decreasing trend.

Our second measure relevant to this issue is the amount of time spent inactive, or lost. This was measured by watching the video for instances where our protocol participants stopped doing things with the tutor and essentially "froze" in place. Averaged across participants, these data are: 14.8s, .6s, .25s, and .25s for scenarios 1-4. There is a decrease of more than 14s in time spent lost, and virtually all of that time savings comes from the first to the second scenario.

Verbalization Trends

The process of completing a scenario can be roughly broken down into: (a) things that you *do* (e.g., make a table, respond to a question), and (b) things that you *read* (e.g., instructional text, feedback). There is an interesting dichotomy here. The things that one does in order to complete a scenario are mandatory. They have to be done. You can not complete the scenario without sorting the data, filling in the column headings, filling in the frequency column, and so on. For the most part, however, it is not mandatory to read the text on the screen. All of the scenarios in a given section have the same structure, and generally speaking, the same text appears over and over again, in the same location, for every scenario in each section. As we will see, this characteristic of the tutor allows for adaptation away from reading text, either learning to ignore it because it is not useful, or because it is possible to retrieve from memory a declarative chunk for the semantic content of the text, rather than invest the time necessary to re-read it.

Table 1 indicates that we coded the protocols in such a way as to represent verbalization of (a) question text, (b) instruction text, (c) instructional text that appears with positive feedback, and (d) positive feedback text. A description of all four of these text types follows. Figure 4, which is a skeletal representation of the Stat Lady interface, will help illustrate the locations of particular text types.

Questions. In the 2nd half of each scenario, having already made a frequency distribution with their data, participants were instructed to answer a series of general definition questions and questions about the distribution. Examples of such questions would be: “What is the symbol for sample size?” and “How many days did you have 11 complaints?” Each question tests a declarative or procedural curriculum objective. Within a given section, these questions recur in exactly the same order in each scenario, and they appear in the area labeled “Activity Window” in

Figure 4. Adding the number of words of this type that participants verbalized in each scenario results in a raw frequency count of the verbalization of question text, and dividing this by the total number of possible question word verbalizations results in a "proportion of words verbalized" measure for each scenario. Figure 5 shows the verbalization rates for these questions. The most obvious difference between this measure and the others (still to be described) is that students verbalized a considerably higher proportion of the Question text. This is not surprising, since at least some portion of each question must be read in order to know what it is that you are answering. Another striking thing about these data is that the verbalization rate for the questions is *more* than 1.0 during the first scenario. This is due to occasional confusion regarding the intent of the question, which results in re-reading. By reading portions of some of the questions more than once, the protocol participants managed to push the verbalization rates for this measure over 1.0 in the first scenario. In subsequent scenarios, they read about 25% less text in the questions.

Instructions. Instructional text sentences are those which tell the learner what to do and how to do it, or reiterate a point relevant to a curriculum objective. Sometimes it is the case that instructional text appears in the Activity Window, as well, but the majority of it occurs in the Instructional Window. An example of instruction text can be seen in Figure 4, where it begins "Step 1: Click on the ..." During the protocol coding, we identified every instance of verbalization of instructional text in these two windows. Figure 5 presents the proportions of instruction text verbalized across scenarios, as the 2nd curve from the top. There is a very clear trend of decreasing verbalization of instructional text with practice. It is important to emphasize that these instructions repeat verbatim from one scenario to the next within a section. The same instructional text appears in the same locations in all four scenarios in Section 1.

Instructions in positive feedback. It is often the case that, following a successful action on the part of the student, some text appears along with the positive feedback which either restates a piece of declarative knowledge or tells the learner what to do next. Given the nature and location of this text, we coded it as instructional text that appears in the positive feedback window. This text type is basically indistinguishable from the instructions in the Instructional Window, aside from its location. An example would be, "Click on the next red ? and enter the correct symbol/formula."

We used the same procedure described above to compute a "proportion verbalized" for this variable, which resulted in the data shown as the third line in Figure 5. The same trend in the verbalizations is evident, but with a much shallower slope, largely due to the floor effect from two of the participants. Note that the initial verbalization rate for this variable was very low, indicating that learners seem not to attend to this text type very much regardless of practice. Since it does carry some information value, this may seem surprising, but there are two possible explanations. First, it is often the case that the instructional text that appears in the positive feedback is redundant with instructional text available elsewhere on the screen, so the student can go elsewhere for the same information. Second, instruction text in this location is at a bit of a visuo-spatial disadvantage, since it occurs in the same block of text as the positive feedback, which, as we shall see next, garners a small amount of attention.

Positive feedback.[6] The primary location for positive feedback throughout the tutor, as suggested in Figure 4, is the Feedback Window. It is the case, however, that positive feedback does sometimes occur in the Instructional and Activity Windows, as well. We coded positive feedback without respect to its location. It is distinctive and easily coded, regardless of location, due to the fact that we restricted our Positive Feedback code to be applicable only to that text

which carries confirmatory information and does not contain elaborative curriculum-relevant information (like that found in instructional text). For example, the two segments: "Congratulations!" and "You are correct." would both be coded as positive feedback.

Figure 5 also shows "proportion verbalized" results for positive feedback text. There is a now-familiar decreasing trend in the verbalizations of positive feedback. As noted previously, one striking characteristic in these data are that they indicate a generally low level of overall attention paid to the positive feedback.

There are a couple of contributing factors which may account for this. First, the positive feedback is simply that. In other words, it merely indicates that the student was correct and generally does not include additional information that is of very high utility (e.g., useful, relevant declarative knowledge that could add to the learner's knowledge base for the domain). This lowers the expected utility of attending to that text, since the payoff is not large. Nevertheless, one might argue, it is useful for the learner to know whether the response was correct, and one should therefore be likely to seek out that information. This is true, but it turns out that, in all but a handful of exceptions, the student has this information before the positive feedback even appears on the screen. One of the features of the tutor is that it provides auditory feedback on performance, in addition to the text-based feedback. Due to a design constraint, this auditory feedback always appears before the text feedback, so the student actually *hears* whether the answer was correct, and can subsequently afford to ignore the text.

Summary of Text Verbalization Results

What we have found is that there are a number of changes in reading behaviors taking place which, in sum, explain a good deal about how students might increase their efficiency in a learning environment. Our interpretation of the result involving Question Text and Instruction

Text is that during the first scenario learners deliberately read through the instruction and question text in order to correctly perform the desired activities. As a by-product of this, they develop a representation for information regarding (1) the order in which to do the tasks required by the scenario and (2) how to do them. Those two categories subsume the semantic contents of both the questions and the instructional text. Upon starting the 2nd scenario, it is immediately apparent to students that they are dealing with the same curriculum-objective structure (i.e., order in which the objectives occur) and interface design as the previous scenario, and so can rely on these declarative chunks to guide them through the process of completing the scenario. Retrieving these chunks from memory is almost certainly more efficient than attending to and reading text, and can therefore be considered an adaptive behavior, from the standpoint of learning efficiency.

In addition to the verbalization trend data, there are also some specific student comments, offered spontaneously as they worked on the tutor, which support this conclusion. For example:

"I've now abstracted over all these problems." - Participant 2, Section 1

“I sense a sequence.” - Participant 2, Section 3

"I didn't read all that . . . you know what it says." - Participant 3, Section 1

Thus, we also have evidence that these participants eventually became aware themselves that they had developed the sort of declarative chunks that we posit here.

A similar cost/benefit "adaptation" interpretation seems appropriate for the results involving verbalization of positive feedback. The positive feedback text in this particular tutor has very low informational value because of the auditory feedback messages that come first. There is very little reason to attend to on-screen feedback text, because the student already knows the answer is correct by the time the text appears. Learners are sensitive to these kinds of

utilities, as suggested by the fact that reading rates for text that appears in the feedback window are low after just one instructional section.

That is not to conclude, however, that positive feedback more generally has no utility. The feedback sound files, both positive and negative, that are used in the Stat Lady tutor tend to be witty and sometimes humorous. Perhaps better than any other feature of the tutor, they capture the "personality" of Stat Lady, which might best be described as sassy. There are many examples in the protocols in which students actually responded verbally to the auditory feedback, as if they were talking to a person. They seemed to enjoy the feedback, especially early on. Later in the tutor, however, after they had heard the entire pool of positive feedback sounds many times over, students sometimes got impatient waiting for the feedback sound to finish playing so they could get on with the business of doing the tutor. Thus, what our data really have to say about the positive feedback is that students seem to enjoy the auditory feedback, especially early in the tutor, but come to ignore the redundant positive *text* feedback.

Reading Time

Given that participants tended to show decreases in the proportions of total text they read as they had more exposure to the tutor, we speculated that reading time could account for a substantial portion of the remaining learning curve. Reading time was computed by estimating a reading rate for each participant. In order to do this, we needed some text they were consistently reading and for which we could obtain reliable time data. The Context portion of each scenario served this purpose well, since our verbalization data showed that it was consistently read, and also because each Context section is long enough (usually 3 or 4 sentences) to allow participants to establish a reading pace. We divided the time spent reading Context by the number of Context words actually read to arrive at a reading rate figure for each learner in each scenario, then

averaged across scenarios to get all four participants' individual reading rates. These were 250 ms, 300 ms, 391 ms, and 437 ms per word for the four participants.

By multiplying those reading rates by the number of words of each type of text verbalized, we were able to estimate how much time each learner spent verbalizing on-screen text during every scenario. These data, averaged across participants, are 94.5 s, 70.7 s, 54.6 s, and 49.7 s for the four scenarios, and they show that there is approximately a 45 s decrease in the amount of time learners are reading text from the first to the last scenario. Looking back at the data in Figure 3 marked in the legend as "Curr.Obj. Time: Coarse," we see a 105 s decrease in completion time from the 1st to the 4th Scenario. Thus, it turns out that the trend towards decreasing text reading over scenarios is responsible for about 44% of the total improvement in time required to complete the curriculum objectives.

The coarse measure of curriculum objective time in Figure 3 indicates a 105 s decrease in completion time from the first to the fourth scenario. That improvement across scenarios is reduced to 46 s in the bottom curve (Curr.Obj. Time: Fine), which has "lost in the interface" time and reading time subtracted out. This means that 59 s (56 % of the change in the coarse-grained learning curve) can be attributed to a combination of learning the interface (14 s), which is reflected in changes in "lost" time, and adaptive changes in reading behaviors (45 s). By subtraction, then, only 44% of that coarser measure is attributable to actual improvement in the execution of the curriculum objectives.

Experiment 1 Discussion

The empirical work completed for Experiment 1 provides evidence for the dramatic impact that the acquisition of knowledge of the learning environment and the pattern of activities in the tutor can have on a performance measure like scenario completion time. Verbal protocol

data showed that more than half of what would normally be considered a curriculum objective learning curve was actually due to interface learning and adaptation. That is a striking result for anyone, including ourselves, who would have been satisfied with the assumption that the learning curve results almost entirely from the strengthening of curriculum objective skills.

One limitation in our results from Experiment 1 is that the reading time data rely on the critical assumption that a lack of verbalization means a lack of reading. It is, of course, possible that our subjects sometimes were reading the on-screen text to themselves and not verbalizing it. This only has negative ramifications on the interpretation of our results, however, if the proportion of read-but-not-verbalized text increases over the scenarios. If this were the case, the actual proportion of the learning curve attributable to changes in reading behaviors would be reduced. Although (in the absence of eye movement data) we can not rule out this possibility definitively, we feel this account is unlikely, given the verbalization instructions to the participants and the presence of the experimenter in the room.

Another limitation involves the verbal protocol participants themselves. Despite confidence that our interpretation of the data is a fair one, it must be acknowledged that these conclusions may not necessarily generalize to a different population of learners. The protocol subjects had relatively high prior knowledge and learned what they did not already know so well that we have to admit their "special" status as good learners. Would we actually see the same dramatic impact of interface learning in a different population, or is this just an artifact of the verbal protocol participants' skill level? It could be that less-skilled learners would have to commit so much of their attention to learning the curriculum objectives that they would not have the time or resources for also adapting to the interface. Another possibility is that providing the concurrent verbal protocols changed the nature of their learning in such a way as to exaggerate the

interface learning effect. Perhaps the impact of interface learning would be reduced in students who did not have the additional burden imposed by the talk-aloud protocols. Concerns about the generality of the results motivated a replication experiment, which we describe next.

Experiment 2

Experiment 2 is different from the first experiment in three complementary respects. First, the students come from a more average population of learners who have considerably less prior knowledge, more in keeping with the sort of population one would typically expect to be using a tutor to learn about a new domain. Second, this is a larger sample, thus allowing for greater confidence in the results. Third, participants in this experiment do not give verbal protocols, so there is no chance of this requirement influencing their learning of either the curriculum or the interface.

With no verbal protocols in Experiment 2, we were required to rely on the tutor log files of student actions. Stat Lady was not originally designed to record data at the level of detail necessary for our decomposition analysis, however. For this reason, it was necessary to modify the data-recording code. This alteration to the tutor code is completely invisible to the user. It does not affect the performance of the tutor in any perceptible way, but does allow for a detailed investigation of changes in student behaviors. For example, the new computer log files provide time stamps for the appearance of text on the screen, which allows for more accurate assessments of the time students are attending to that text before they initiate problem solving. Therefore, we can separate out text reading time from curriculum objective time.

The verbal protocol study described in Experiment 1 did more than teach us about the impact of interface learning on performance. That study also led us to consider the implications of our results for redesigning the tutor to make it a more efficient teaching system. We generated

some hypotheses regarding just that issue and decided to test them out in a 2x2 factorial design - resulting in four conditions with different versions of the tutor in each condition. This is not the place for a full description of all of those conditions, as they are not critical for the purpose of this paper.[7] The important thing for our purposes here is that one of the conditions was a Baseline condition, which was the original form of the tutor, the same as that used in Experiment 1. The data for Experiment 2 come strictly from students in that Baseline condition.

Method

Participants

The replication data come from a sample of 31 participants, all of whom participated in this study at the Air Force Research Laboratory's TRAIN Lab at Lackland AFB, Texas. Participants were hired through local temporary employment agencies and were paid $6 per hour. They ranged in age from 17 to 37 years old, with an average age of 23. The sample was 35% female and 65% male. All participants had earned at least a high school degree or an equivalency degree (GED), and 7 of them had earned a college degree.

Materials

Equipment. All participants used a Gateway 2000 Pentium 100 computer with a 15" color monitor, an extended keyboard, and a standard mouse. The TRAIN Lab is a data collection facility with 30 computers arranged next to each other around the perimeter of the room. Each machine occupies its own desk and is separated from the others by dividers. No audio or video data were collected.

Software. Time constraints necessitated a change in the length of the curriculum. We wanted to limit the time necessary to complete the study to one full 8-hour workday. Completing as much of the tutor as our verbal protocol participants did, along with the pretest

and posttest, in-processing time, breaks, lunch, and so forth, requires considerably more time than that. One way to bring down completion time is to decrease the number of scenarios required in each section. Another way is to decrease the number of sections. We would not want to get too carried away with this latter proposal, however, since many of the results from Experiment 1 involve sections 1 and 3, and those are the data we wish to replicate. Thus, it was necessary to leave intact at least the first three sections of the tutor. In the end, the tutor was modified so that all participants would complete three scenarios in each of the first three sections, and then stop. The fourth and fifth sections were simply removed.

With the last two sections of the tutor gone, it would have been a waste of time to pretest and posttest on the full curriculum, so we removed all test items specific to the last two sections. In all other respects, the pretest and posttest were identical to the versions given to participants in Experiment 1.

Procedure

Upon arrival at the laboratory, participants were randomly assigned to one of the four tutor variant conditions mentioned earlier. Again, all of the data under analysis here are from those people assigned to the Baseline tutor condition. Participants completed a demographic survey, then the pretest, the tutor, the posttest, and a subjective experience survey. All work was done individually.

Results

There are two questions to be addressed. The first regards how similar the new sample is to the verbal protocol sample. Recall that it was our goal to find a different population, so as to test the generalizability of the results from Experiment 1. The second question is whether we get a replication of the decomposition results.

Population Differences

Our first global performance measures for assessing population differences are pretest and posttest scores. Participants in Experiment 2 had pretest scores averaging 52% (SD = 12.5) and posttest scores of 84% (SD = 9.7). The pretest scores show that participants in Experiment 2 possessed considerably less prior knowledge than did those in the first experiment. With gain scores averaging 32 percentage points, they also learned quite a bit, but their posttest scores remain well below those of the verbal protocol participants.

Two other global measures of performance are average scenario completion times and error rates. Both variables will be presented as averages over sections 1 and 3. Recall that Section 2 was not included in our analyses of the verbal protocol data, so we exclude it in these analyses as well, for accuracy in the comparison. Summary statistics for these data are displayed in Table 5. The new participants made more errors, and thus were slower, than the verbal protocol participants. Clearly, the new population is sufficiently different from the verbal protocol participants to make replicating the interface adaptation results a worthwhile exercise. Will it be the case that changes in low-level interactions with this tutor also account for a high percentage of the learning curve in this new population?

Learning Curve Decomposition

To answer this question, we will provide the same style of decomposition that was offered for the data from Experiment 1. The primary difference between the two sets of data, aside from their being from different populations, is that the method for measuring time is different in some cases. This is because data analysis in the previous experiment relied largely on the use of videotapes, whereas in this experiment, we are working strictly from computer traces of the participants' behaviors. The measurement methods for each variable will be explained in

turn, as we complete another componential analysis of the learning curve.

Scenario Completion Time

These data include all scenario activities except for filling in the values for the variable and frequency columns of the worksheet and a number sorting task at the beginning of the section 1 scenarios. These activities are omitted due to technical complications in extracting accurate error time data. This amounts to the removal of only 5 of the 28 curriculum objective activities in sections 1 and 3, so the vast majority of the curriculum is included in all subsequent analyses. However, it happens that these five are fairly time consuming activities. As a result, the total completion times are deflated. The only implication of this is that it will appear that participants at the TRAIN Lab were finishing the scenarios somewhat faster than they actually were. This is not significant with respect to the decomposition analysis, because it is the *proportion* of the learning curve that is attributable to different time variables that is important, and not the absolute values of those variables. Scenario completion time averages with those activities omitted are 497 s (154 s), 339 s (86 s), and 280 s (72 s) for scenarios 1, 2, and 3, respectively. These values provide us with starting points for the decomposition, which are plotted at the top of Figure 6.

Error State Time

In this experiment, we have to rely exclusively on patterns of incorrect and correct answers to determine the length of error states. This process is really very simple. The beginning of an error state is signaled by the first mouse click that initiates a response which is incorrect. The end of that error state is signaled by the next mouse click that initiates a response which proves to be a correct answer. The mean time spent in error states by participants in Experiment 2 was 143 s (93 s), 55 s (43 s), and 27 s (33 s), across scenarios 1, 2, and 3, respectively.

Subtracting this from Total Time, we arrive at Error-Free Time in Figure 6.

Peripheral Activity Time

To review, the peripheral activities, which did not involve curriculum objectives, included reading the context at the beginning of the scenario and retrieving data from the Number Factory. Mean peripheral activity times for these participants were 79 s (31 s), 60 s (15 s), and 53 s (14 s) for scenarios 1-3. That is a change of 26 s in peripheral activity time over the course of the three scenarios. It turns out that this population of learners shows a substantial decrease in the time spent on the context page at the beginning of each scenario. This suggests that some people stopped reading the contexts, or at least stopped reading them as carefully. These data also show a slight decrease in the time spent in the Number Factory, which may be indicative of some initial confusion regarding the use of the Number Factory in the first scenario. Despite the trends in these data, it remains important to remove peripheral activity time from the total time, in order to arrive again at the *coarse* measure for Curriculum Objective Time, plotted in Figure 6. There is a change of 76 s from the 1st to the 3rd scenario. How much of this change is attributable to interface learning, rather than curriculum learning?

Reading Time

In the absence of verbalizations, our procedure for computing reading time is considerably different in this experiment. The general flow of activity in the tutor, which is repeated for every curriculum objective tested, is that some instruction text appears on the screen telling the student what problem to work on next, the student does that activity until a correct answer is entered, then some feedback and more instruction text appear - again suggesting a next course of action to the student. At every step along the scenario, it is possible to assess text reading time as the time from the appearance of text on the screen until the next mouse click signaling the initiation of

problem solving.

Unfortunately, it is not feasible to do the text-type analysis that we performed in Experiment 1. This is because it is very often the case that more than one text type appears on the screen at any given point in time. For instance, after a correct response, it always is true that positive feedback and instruction text appear simultaneously. Therefore, it is impossible to determine whether the time between the appearance of that text and the next mouse click should be assigned to the feedback or to the instructions. This is not really a significant loss, however, since the trend in Experiment 1 was towards decreasing reading in almost every case, and we ended up collapsing across them all in the end, anyway.

It also is important to note that "lost" time, as defined for the previous experiment, is subsumed in this measure. In the verbal protocol data, every single example of being lost in the interface occurred just after the appearance of some instruction text and before the subsequent mouse click. Because participants in Experiment 2 are using the same interface, the same is likely to be true. Thus, our "reading time" for this experiment is really a conjunction of reading time and lost time from Experiment 1.

The Reading Time data, which actually measure the development of interface knowledge and adaptation, are 130 s (40 s), 96 s (31 s), and 81 s (28 s), for scenarios 1-3. This is a 49 s drop. There is indeed quite a bit of interface learning going on in this population.

The really important question, of course, is how much of the speedup in this coarse-grained measure of curriculum objective problem solving is actually attributable to changes in low-level interactions with the tutoring environment? Looking back at the coarse measure of curriculum objective performance, we see that there was a 76 s change. Apparently, 49 s of this is attributable to interface learning. Note that the bottom curve in Figure 6, which has interface

learning subtracted out and is labeled "Curr.Obj. Time: Fine," has a considerably more shallow slope. We conclude, therefore, that interface learning accounts for fully 64% of the change in the more coarse-grained assessment of curriculum objective completion time. The impact of interface learning is even more dramatic in this second population of learners.

A Model of Learning about the Learning Environment

The learning processes that we believe provide a reasonable account for the data from both studies are: (1) the acquisition of knowledge regarding the sequence of activities in the scenarios - which allows for less reading of instruction text and more recall from memory, and (2) adaptive changes in reading behaviors, such that redundant and uninformative text comes to be ignored more often. This interpretation would be bolstered if we could *demonstrate* that (a) a detailed implementation of these processes actually does produce these kinds of qualitative and quantitative changes in behaviors, and that (b) these processes do not require any special-purpose learning mechanisms. Such a demonstration requires a running computational model that can reproduce the data and that is built within a computational architecture that posits a basic set of general learning mechanisms. An increasingly common means of creating such models is through the use of cognitive architectures.

Gray, Young, and Kirschenbaum (1997) recently commented on the growing foothold such architectures have in bringing cognitive theory to bear on HCI issues. In their special issue on the topic of "Cognitive Architectures and Human-Computer Interaction" there are articles on four running architectures: ACT-R, EPIC, LICAI, and Soar. We have used one of these, ACT-R (Anderson & Lebiere, 1998), to develop a production system model that not only matches the verbal protocol and latency data from Experiment 1, but does so using the processes postulated above. We describe this model below. It serves as a proof-of-concept for these interface learning

processes, and also as a demonstration of the applicability of the ACT-R architecture for increasing understanding of phenomena occurring at the interface of the human and the computer.

The ACT-R Model[8]

For our purposes, it would be impractical to model the process of completing the entire Stat Lady tutor. Instead, we select a representative activity and model the learning of this activity, as a proof-of-concept. The activity chosen to be modeled is the first step in the process of completing a frequency distribution - filling in the column headings. This particular portion of the tutor was chosen for two reasons. First, it is a fundamental skill taught at the beginning of the Stat Lady tutor. Later sections build on this knowledge in teaching the construction of more complicated distributions. Second, this activity can be characterized generally as part of a "spreadsheet" procedure, and is similar to the sort of actions required of students in the PUMP Algebra Tutor (Koedinger, Anderson, Hadley, & Mark, 1995). Thus, a working model of this activity in Stat Lady may be directly relevant to explaining students' changing behavior in other learning environments.

Students working with Stat Lady proceed through a series of steps in order to accomplish the goal of completing the frequency distribution. A broad characterization of the process, as it is done in Section 1 of the tutor, is that it consists of two main steps: (1) Filling the Column Headings and (2) Filling the Frequency Column.[9] Table 3 contains a synopsis of the procedure for Filling the Column Headings (see Figure 4 for a representation similar to what is on the screen when students begin this activity). In all, there are 17 separate pieces of text that appear during this activity, accompanied by 8 overt actions involving the mouse. The entire "column headings" activity is mouse-driven.

Data to be Fit. The goal is for the model to approximate the data from the verbal protocol

participants in Experiment 1. Generally speaking, we would want the model to (a) capture the sort of verbalization trends we see at the scenario level (i.e., decreasing attention to most of the text with practice, and generally low attention to text in the feedback window), and also (b) to decrease its performance times in a way comparable to that of our protocol participants. Table 4 contains the verbalization counts and latencies the model is intended to replicate for the activity of filling in the column headings. The verbalizations are frequency counts of verbal protocol segments coded as the reading of on-screen text during this activity, and latencies are in seconds. Note that the same trends hold at the level of this single tutor activity as at the more gross scenario level of analysis: both verbalizations of text and solution times decrease with practice.

Model Overview. The model actually consists of two parts. First, there are the productions and declarative knowledge chunks which constitute the cognitive skill of performing this task. In ACT-R, *productions* and *chunks* are the atomic components of cognition (Anderson & Lebiere, 1998). A production is an IF-THEN condition-action statement which serves as the procedural basis of cognitive skill. Productions produce action. A chunk is a proposition containing factual knowledge, such as the fact that Pittsburgh is in Pennsylvania.

The second part of the model is a computer simulation of the Stat Lady tutor that provides information to the cognitive model (text to read, buttons to click, etc.) just as the real Stat Lady tutor provides information for students to process. The Stat Lady simulation is written in Lisp as a list of chunks representing text, buttons, windows … everything available on the tutor interface. This presents information in a simplified format that the cognitive model can attend to and parse. When the model attends to something in the simulation, it is akin to a student attending to some on-screen information. The information is then taken to be represented internally as a declarative chunk by the model.

Knowledge Representation. The model makes the assumption that, at each step along the way, learners have to either retrieve the next action from declarative memory or read on-screen text to find out what to do next. The following production notes that the model does not know what to do next and then sets a subgoal to find out what that step should be:

Figure Out What To Do Next
IF the goal is to complete the tutor,
and the previous action is known
and the appropriate current action is not known,
THEN push a subgoal to get the current action that follows the previous action

With that subgoal set, the model then chooses among productions for (a) scanning a region of the screen for informative text or (b) retrieving the next step from declarative memory. There are two screen scanning productions, one for the instruction window and one for the feedback window. These are the two locations where most of the text occurs. The production for scanning the instruction window looks like this:

Scan Instruction Window
IF the goal is to get the next action
and the previous action is known
and the appropriate next action is not known,
and I am not currently attending to information on the screen,
THEN push a subgoal to scan the
instruction window

The alternative to reading the next step off the screen is to retrieve it from memory. Here is the retrieval production:

Retrieve Next Action
IF the goal is to get the current action
and the previous action is known
and the appropriate current action is not known,
and I am not currently attending to information on the screen,
and there is a chunk in memory for the
appropriate current action,
THEN pass that memory to the current goal
as the appropriate current action

The first time through a scenario, the model does not have any chunks in memory for the appropriate next actions, just as a person who had never completed the tutor before would have no knowledge of the sequence of activities. The model is required, therefore, to read a good deal of on-screen text during the first scenario.[10] Reading instruction text achieves the subgoal of finding the next action to be performed, and this subsequently results in the setting of a new subgoal to perform that action. After the action is completed and that subgoal is popped, ACT-R automatically creates a chunk in declarative memory that pairs the current action with the previous action, because they were both present in the goal chunk of the subgoal that was popped. This is how the model acquires knowledge of the sequence of activities involved in completing a scenario. To get an idea of what the representation for these "episodic" memories is like, here are a couple of examples:

click-red-?

isa	action
previous	sort-descending
current	click-red-?

enter-x

isa	action
previous	click-red-?
current	enter-x

Once the first scenario is completed, this knowledge is then available for retrieval on subsequent scenarios, which allows for decreases in reading behavior. The next action to be taken can be retrieved from memory instead of deduced from instruction text.

While all of this declarative knowledge acquisition is taking place at the symbolic level, there are also some sub-symbolic changes going on. Specifically, this is the tuning of parameters

which control the selection of productions. In ACT-R, choice is determined by comparing the expected gain of all productions that match the current goal and selecting the one with the highest expected gain. The expected gain of a production is determined by a number of factors. The most important ones in this model are the prior history of success and the prior history of failure when that production has fired in the past. Success and failure are assigned with respect to whether the current subgoal has been successfully achieved. Any productions that fire during the successful completion of a subgoal acquire "success" and those that fire during unsuccessful subgoal completion acquire "failure." As the ratio of a production's successes to its failures increases (more successes), the expected gain of that production increases. Similarly, as that ratio decreases (more failures), the expected gain decreases.

There are a number of different productions for reading different types of text on the screen: descriptive instruction text, directive instruction text, feedback text, and so on. The decision to use different productions for reading each type of text is based on the observation that the relative locations of these different text types is very consistent throughout the tutor. For instance, descriptive instruction text (e.g., "You finished the first column.") precedes the directive text (e.g., "Now click on the red question mark in the second column and enter the correct column heading."). Although we have no direct evidence for this, it is consistent with the pattern of results to assume that students are sensitive to the general location of text that is typically useful and also to the general location of text that is typically not useful. By using this particular representation, it is possible to capitalize on ACT-R's expected gain computation as a mechanism for directing visual attention. Only directive instruction text contains directions for the next step, so productions for reading text of this type are marked as "success" productions. This is how the model knows the subgoal of getting the next action off the screen has been

achieved successfully. Productions for reading other types of text are considered “failures” of the subgoal to find out what to do next. These successes and failures in attempting to read information about what to do next are automatically taken into account each time ACT-R makes a decision about which production to fire next. It is in this manner that the model quickly develops a bias against reading anything but instruction text, much as the participants did.

The model does not include an explicit representation for eye movements or motor movements. The time required for such actions is built into the effort parameter in the productions. This design is consistent with the fact that this is intended primarily as a cognitive model, and not as a model of perceptual or motor processes.

Model's Performance. Having established the basic structure of the model, the critical question now becomes whether it behaves as the human participants behaved. In order to allow for changes in performance with practice, we ran the model through the same scenario representation four times. Note that this is not exactly identical to the situation the human learners were in, since they did a different scenario each time, but the scenarios are all similar enough that this serves as a fine approximation. Looping through the scenario four times provides an equal amount of data as would be provided by one of our protocol participants. To allow for some stochasticity, noise is added into the model’s production selection procedure. Therefore, in order to get a fair sense for the central tendency of the model’s behavior, we simulated 100 subjects and averaged over their results to get the model’s data.

The model's verbalization and latency data are presented in Table 4, and the latencies are graphed in Figure 7. As is evident there, it performs in a manner similar to what we see in the verbal protocol participants. As a first comparison between the model and the data, note that the model does produce the desired verbalization trend with regard to text reading. For all three text

types there is less reading across scenarios. The model produces this behavior, a decreasing overall reading trend, through the acquisition of declarative knowledge for the order of events in the scenarios, which allows it to attend less to on-screen text over time.

As a second comparison between the model data and the human data in Table 4, note that the relative text reading rates produced by the model are comparable to those seen in the protocol data (e.g., less reading of feedback text than instruction text). The model produces this second result, the relative reading rates of different text types, through sub-symbolic tuning of the parameters which control conflict resolution.[11]

A final comparison between the model's behavior and the data focuses on the latency data in Figure 7. Note that the latencies produced by the model are similar to those produced by the protocol participants. This learning curve falls naturally out of the two learning processes just described.[12]

Both qualitatively and quantitatively, the match of the ACT-R model's behavior to the human data is quite satisfying. The model demonstrates explicitly that the hypothesized interface learning mechanisms can account for the data in Experiment 1. The model uses standard ACT-R learning and selection mechanisms to store information on the sequence of activities as declarative knowledge, then adapt its reading preferences once that declarative knowledge is available to direct action. These are not special-case mechanisms designed to achieve a match just to these data. They are very general learning processes that were already assumed to exist in the ACT-R cognitive architecture, and they produce the same quantitative and qualitative changes in behavior we saw in the verbal protocol participants. The characteristic that distinguishes this model of learning to interact with a computer-based tutor from most other production-system-based accounts is the important role of declarative knowledge acquisition and the effect that has on

problem solving time.

Summary and Conclusions

Newell and Simon (1972) wrote:

> Just as a scissors cannot cut paper without two blades, a theory of thinking and problem solving cannot predict behavior unless it encompasses both an analysis of the structure of task environments and an analysis of the limits of rational adaptation to task requirements. (p. 55)

We think this admonition applies to the understanding of learning from computer tutors. To really understand what is happening, we need to attend carefully to the structure of the tutor interface, and the way learner behaviors change in that interface, and not just content ourselves with measures of posttest gain on curriculum objectives and improvements in global completion times. We have found that many of the subtleties in the learning that is taking place are not reflected in these coarse measures.

Any learning experience must take place in some learning environment. The student's facility in navigating through that environment is going to determine the amount of overhead they encounter in achieving curriculum objectives. So even if we place no value on learning the environment *per se* it behooves us to understand the nature of how students learn about that environment. We have been looking at a particular computer interface, and have shown that our results generalize across different populations of students.

In the two experiments, 56% (Experiment 1) and 64% (Experiment 2) of students' error-free speed-up in problem solving was attributable to interface learning, rather than curriculum learning. Two processes appear to be responsible for this interface learning: the acquisition of

knowledge of the sequence of actions, which makes reading the instructions unnecessary, and adaptation to more useful portions of the tutor interface so redundant information can be ignored.

Generalizability and Interpreting Improvement

We have not yet addressed the issue of whether our results are specific to the Stat Lady tutor. Although it is a fine tutor in its own right, other instructional designers are likely to be interested in the extent to which our results have implications for interpreting performance improvements in their own learning environments. We believe that it is not the case that the conclusions we have reached here are relevant only to learning from Stat Lady. There is evidence that significant interface learning is going on in other tutors, as well. Earlier we cited the data from Anderson et al. (1989), where a similar conclusion was reached regarding learning from the LISP Tutor, and Bishay (1996) provided evidence that interface learning impacts performance in the PAT Algebra tutor (Koedinger et al., 1995). Others have reported changes in reading behaviors that, as we conclude here, result from increased familiarity with the domain and with the demands of the learning environment (Harvey & Anderson, 1996; Kieras & Bovair, 1986).

The extent to which interface learning, as a component of the overall learning curve, should be important to any given instructional designer depends entirely on the overhead that the interface imposes on anyone trying to carry out the curriculum objectives. It should be the case that the less burden the interface imposes, and the more transparent it is in the problem solving, the less critical it is to take into consideration when interpreting improvements in performance time.

A related issue is how one goes about distinguishing the interface from the curriculum. One way to think about this is in terms of transfer. Designers of instructional technology are generally interested in whether students will transfer what they have learned in their particular

computer-based tutor to some alternative environment, like a job site, for instance. The knowledge and skill those students are supposed to apply in the new environment is the curriculum. Everything else is interface. For instance, in the Stat Lady tutor, one piece of curricular knowledge that is useful for constructing a frequency distribution table is that the symbol 'X' is a standard symbol for the variable column. We would want a student who has completed the Stat Lady tutor to know that and be able to use it in a different environment, like making a frequency distribution with paper and pencil. Knowledge of the location of the Fill Column button, which is interface knowledge, is irrelevant in the paper and pencil environment.

Implications for Those Interested in Curriculum Learning

Primarily implicitly, but sometimes explicitly, we have made a distinction between those researchers interested in issues surrounding curriculum learning and those researchers who are more interested in studying interface learning. Obviously, these interests are not mutually exclusive. It occurs to us that the results presented here have implications for scientists with either or both of those interests. This section discusses issues relevant to curriculum learning, and the next section discusses issues relevant to interface learning.

The critical component which distinguishes intelligent tutoring systems (ITS) from other computer-based instructional systems is the presence of a *student model* (Greer & McCalla, 1994). Student models are a special case of user models, in that they are specifically developed for monitoring knowledge and skill development in instructional domains. They are used for making real-time decisions regarding promotion through the curriculum, problem selection, hint generation, and feedback generation. There are many different approaches to student modeling. One specific approach, which has proven successful in a number of ITS, is to create the student model from a production system cognitive model of the task similar to the one we developed

earlier in this paper. These are process models that run in real time along with the student, and which make it possible to match the step-by-step actions of the student with production firings in the model - a process called *model tracing*.

This research has implications for the use of student protocols in model tracing to infer what students know. It shows that student performance can be speeding up because of interface learning and this may or may not mean that they are gaining further mastery of the curriculum objectives. To separate out interface learning from curriculum learning requires that we do more fine-grained analysis than simply measuring total time and total errors for each curriculum objective activity.

This addresses an issue mentioned by VanLehn (1988), regarding the appropriate grain size for student models:

> There is a tacit assumption that tutoring based on fine-grained student models will be more effective than tutoring based on coarse-grained models. No one has attempted to check this assumption. (p. 75)

Our results clearly suggest that a fine-grained student model, say one that distinguishes between curriculum objective activities and interface activities, would be a more accurate student model. Such a fine-grained student model could use latency information (chronometric data) to estimate learning rate. VanLehn goes on to comment on the use of chronometric data:

> The amount of time between the student's actions is one type of information that is available for free but that so far has been ignored by every ITS I know of. Chronometric data has been used in psychology for years as a basis for deciding between potential models of human cognition. It would be interesting to see whether chronometric data would favor fine-grained student modeling. (p. 75)

We would like to suggest that a fine-grained student model, which used chronometric data and that could distinguish between curriculum and interface learning, would be a more accurate model of student learning of the curriculum.

The difference between fine- and coarse-grained models has implications for predictions of retention and transfer. In two experiments we found that a substantial portion of the speedup in performance was due to interface learning. Coarse-grained measures of performance, which conflate this interface learning with curriculum learning, show a faster learning rate (steep curve). Fine-grained measures, which more accurately separate these two kinds of learning, show a slower learning rate for the curriculum objectives (shallow curve). A faster learning rate suggests students have learned the material better and that they will exhibit better retention and more likely transfer to a new situation. Since a large part of this speedup is not due to curriculum objectives, a coarse-grained analysis could lead to over-prediction of student knowledge and skill. The slower learning rate of the fine-grained analysis, however, suggests relatively less retention and transfer. This learning rate is based on a more accurate analysis of actual curriculum objective learning; hence, is not exaggerated by additional interface learning. This highlights, once again, the likely advantage of finer-grained modeling - more accurate predictions of future performance.

Implications for Those Interested in Adaptation to the Learning Environment

The focus of this special issue is on using cognitive models to improve HCI. The focus of our project has been on describing the relative contributions to the learning curve of learning about the learning environment and learning the curriculum, and also on accounting for the data with an ACT-R model. For those interested in improving HCI, and for those interested in interface learning *per se*, or even in conjunction with curriculum learning, the results presented in this paper are potentially interesting in at least three ways. First, the fact that a substantial

portion of the learning curve was attributable to increasing familiarity with the interface affirms the importance of the human-computer interface as a research platform. The field needs to continue to expand the understanding of how people use, learn about, and adapt to computer learning environments.

Second, we have provided a set of measurable behaviors which serve as indicators that something is being learned about the learning environment. These behaviors include the following:

- A reduction in the reading of instructional text for what to do next.
- A reduction in the reading of redundant/uninformative text.
- A reduction in uncertainty regarding how to implement the next action in the interface.

We have interpreted these behaviors as stemming from a corresponding set of learning processes that can be easily captured by the ACT-R cognitive architecture:

- Acquisition of declarative knowledge for a required sequence of actions.
- Adaptive production tuning to facilitate efficient goal completion.
- Acquisition of declarative knowledge for tools available in the interface.

At an abstract level, these learning mechanisms are applicable in any learning context. That is, the adaptive nature of human cognition makes it a certainty that one or more of these types of learning will take place whatever the environment, whatever the interface.

Finally, these results and the accompanying ACT-R model should be of interest to those who stand for the possibility of using executable simulations as surrogate users for design evaluation. In the context of tutoring system design, a clear implication of these results is that surrogate user simulations should be able not only to learn the curriculum, but also to learn about the learning environment.

References

Anderson, J. R. (1993). Rules of the mind. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. Artificial Intelligence, 42, 7-49.

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. Cognitive Science, 13, 467-506.

Anderson, J. R., Corbett, A. T., Fincham, J., Hoffman, D., & Pelletier, R. (1992). General principles for an intelligent tutoring architecture. In Regian, J. W. & Shute, V. J. (Eds.), Cognitive Approaches to Automated Instruction. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. The Journal of the Learning Sciences, 4(2), 167-207.

Anderson, J. R., & Lebiere, C. (1998). The atomic components of thought. Hillsdale, NJ: Erlbaum.

Bishay, M. (1996). Intelligent tutoring systems: The effects of software structure and support on student learning of equation solving algebra. Unpublished master's thesis, Carnegie Mellon, Pittsburgh, PA.

Card, S. K., Moran, T.P., & Newell, A. (1983). The psychology of human-computer interaction. Hillsdale, NJ: Erlbaum.

Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction, 4, 253-278.

Ericsson, K. A., & Simon, H. A. (1993). Protocol analysis: Verbal reports as data (revised edition). Cambridge, MA: MIT Press.

Gluck, K. A., Shute, V. J., Anderson, J. R., & Lovett, M. C. (1998). The Stat Lady tutor

design project: Exploring feature and context manipulations for better learning efficiency. Manuscript submitted for publication.

Gray, W. D., Young, R. M., & Kirschenbaum, S. S. (1997). Introduction to this special issue on cognitive architectures and human-computer interaction. Human-Computer Interaction, 12(4), 301-309.

Greer, J. E., & McCalla, G. I. (1994). Student modelling: The key to individualized knowledge-based instruction. New York: Springer-Verlag.

Harvey, L., & Anderson, J. R. (1996). Transfer of declarative knowledge in complex information-processing domains. Human-Computer Interaction, 11, 69-96.

Kieras, D. E., & Bovair, S. (1986). The acquisition of procedures from text: A production-system analysis of transfer of training. Journal of Memory and Language, 25, 507-524.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1995). Intelligent tutoring goes to school in the big city. In J. Greer (Ed.), Proceedings of the World Conference on Artificial Intelligence in Education (pp. 421-428). Charlottesville, VA: AACE.

Lajoie, S. P., & Derry, S. J. (1993). Computers as cognitive tools. Hillsdale, NJ: Erlbaum.

Lesgold, A., Eggan, G., Katz, S., & Rao, G. (1992). Possibilities for assessment using computer-based apprenticeship environments. In Regian, J. W. & Shute, V. J. (Eds.), Cognitive Approaches to Automated Instruction. Hillsdale, NJ: Erlbaum.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), Cognitive skills and their acquisition (pp. 1-56). Hillsdale, NJ: Erlbaum.

Newell, A., & Simon, H. A. (1972). Human problem solving. Englewood Cliffs, NJ:

Prentice-Hall.

Shute, V. J. (1995). SMART: Student Modeling Approach for Responsive Tutoring. User Modeling and User-Adapted Interaction, 5, 1-44.

Shute, V. J., Gawlick, L. A., & Gluck, K. A. (1998). The effects of practice and learner control on short- and long-term gain and efficiency. Human Factors, 40(2), 296-310.

Shute, V. J., & Gluck, K. A. (1994). Stat Lady Descriptive Statistics Tutor: Data Organization and Plotting Module. [Unpublished computer program]. Brooks AFB, TX: Armstrong Laboratory.

VanLehn, K. (1988). Student modeling. In M. C. Polson & J. J. Richardson (Eds.), Foundations of intelligent tutoring systems (pp. 55-78). Hillsdale, NJ: Erlbaum.

Author Note

Kevin A. Gluck, Department of Psychology, Carnegie Mellon; Marsha C. Lovett, Center for Innovation in Learning, Carnegie Mellon; John R. Anderson, Department of Psychology, Carnegie Mellon; Valerie J. Shute, Air Force Research Laboratory.

This research was supported by AFOSR grants F49620-95-1-0284 and F49620-96-1-0361, and by the Air Force PALACE Knight Program. We thank Joon Park for his contributions to the verbal protocol coding, Nancy Lefort for her heroic programming efforts, and Lisa Gawlick and the other folks from Galaxy Scientific for their data collection efforts at the TRAIN Lab. Thank you also to Frank Lee for his comments on an earlier draft of this paper.

Correspondence concerning this article should be addressed to Kevin A. Gluck, Department of Psychology, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213. Electronic mail may be sent to kgluck@cmu.edu.

Footnotes

[1] We will use the terms “interface” and “learning environment” interchangeably throughout this paper. In both cases, we are referring to the material that is visible on the computer screen.

[2]Ericsson and Simon actually refer to short-term memory, rather than working memory.

[3] The tutor provides the correct answer after the third error.

[4]A caveat to this statement is that the Number Factory is meant as an analogue to the process of collecting data - which, of course, generally precedes statistical analysis.

[5]Pilot assessments with an earlier version of the tutor showed that subjects often did not note the appearance of a button when they were immersed in the column filling activity described in this example. To facilitate their attending to it, the button was designed to flash in alternating red and gray when it first appears. It’s striking that subjects often still do not see it immediately.

[6] We do not include a separate measure of negative feedback because the verbal protocol participants committed so few errors that it is impossible to pick up any trends in their attention to the negative feedback text.

[7] A complete description of the tutor variants and the results is available in Gluck, Shute, Anderson, and Lovett (1998).

[8] The reader may supplement the following description by examining, modifying, and/or running this model directly: http://act.psy.cmu.edu/ACT/ftp/models/freq-dist/freq-dist.html

[9]Filling the variable column is accomplished with the click of a button, and so we don’t include it as a primary subgoal in the process of completing the frequency distribution, at least not in this section of the tutor. In other sections, it is a more critical part of the process.

[10]This model does not do natural language parsing or comprehension. It “reads” text in an abstracted sense. Each piece of text is represented as a chunk with a slot for its semantic content. The model reads by extracting this slot value (the semantic content) from the text chunk.

[11] Conflict resolution is ACT-R's mechanism for selecting one production to fire when multiple productions match the current goal.

[12] Increasing chunk activations and production strengths are also typically assumed to contribute to decreasing latency curves. Although these could easily be added, the model does not bother with these learning mechanisms, on the basis that they are not necessary (in this case) to achieve the desired behaviors and would simply complicate the model.

Table 1

Coding Categories Used in the Verbal Protocol Analysis of On-Screen Attention

Text Type	Verbalization	Description	Example
Instruction in Instr. Window	Read	First-pass read of Stat Lady instructional text	“Now follow the directions at the top of the exercise.”
	Re-Read	Subsequent reads of previously-read instr. text	“... follow the directions ...”
	Paraphrase	Reads some smaller segment of instruction	“… follow directions at the top ...”
Instruction in Fdbk Window	Read	Reading instructions in the feedback window	"Click on the Fill Column button."
	Re-Read	Subsequent reading of instructions in fdbk. window	
	Paraphrase	Reading a reduced portion of instructions in fdbk. window	
Question	Read	Reading the entire question for the first time	“What age did Dr. Young not see?”
	Re-Read	Subsequent reads of previously read question	
	Paraphrase	Reads a reduced section of the question	
Feedback	Read	Reading feedback	“Isn’t that special?”; "You are correct!"
	Re-Read	Subsequent reads of previously read feedback	
	Paraphrase	Reads a reduced portion of the positive fdbk.	

Table 2

Performance Data from Verbal Protocol Participants- Averaged over Sections 1 and 3.

	Scenario							
	1		2		3		4	
Dependent Measure	M	(SD)	M	(SD)	M	(SD)	M	(SD)
Completion Time	543	(202)	435	(166)	416	(170)	422	(170)
# Errors	2.4	(3.5)	1.8	(2.9)	1.1	(2.8)	1.3	(1.6)

Note: Values in parentheses are standard deviations. Completion time is measured in seconds.

Table 3

Synopsis of the Procedure for Filling the Column Headings

(*Three instruction sentences appear in the instruction window. Simultaneously, data and an empty frequency distribution appear on the right of the screen, in the activity window. There is a red question mark in the header for the variable column, and a gray one in the header for the frequency column.*)

1. Participant clicks on the cell containing the red ?, which brings up a Symbol Pad.
2. Participant uses the Symbol Pad to enter an X.
3. Participant uses the Symbol Pad to click Done.

(*After auditory positive feedback, text feedback appears, accompanied by instructions to click on the Fill Column button. Simultaneous with the appearance of the feedback, instructions come up in the instruction window to click on the Fill Column button. Note that there are redundant instructions on the screen at this point.*)

4. Participant clicks on the Fill Column button.

(*No positive feedback appears, but redundant text appears in both the instruction window and the feedback location, telling the participant to click on the red ? (now in the frequency column) and enter the correct symbol.*)

5. Participant clicks on the red ?, which brings up the same Symbol Pad again.
6. Participant uses the Symbol Pad to enter an f.
7. Participant uses the Symbol Pad to click Done.

(*Feedback appears, accompanied by two instruction sentences. One says that is the correct symbol for the frequency column, and the other says to fill the table by clicking on each cell and enter the answer. Simultaneously, redundant text appears in the instruction window. The text says to click on each cell and enter the answer.*)

8. Participant clicks on a cell in the frequency column to start entering values.

(*This is the point at which we have a time-stamp from the tutor indicating that the column headings are complete. It is also at this point that the model forces goal popping to end its run.*)

Table 4

ACT-R Model (in parentheses) Match to Participant Data for Filling in the Column Headings

	Scenario							
	1		2		3		4	
	VP's	(Model)	VP's	(Model)	VP's	(Model)	VP's	(Model)
Verbalizations								
Instruction	4.00	(4.88)	2.00	(2.11)	2.00	(1.55)	1.75	(1.37)
Inst_Pos	1.75	(1.36)	1.00	(.69)	.25	(.53)	.50	(.48)
Posfeed	1.50	(1.12)	.75	(..36)	.25	(..35)	.25	(..28)
Latencies	53	(58)	38	(35)	36	(31)	30	(30)

Note. "VP's" - verbal protocol participant data. "Model" = model performance data. "Instruction" = instruction text in either the Instructional Window or Activity Window. "Inst_Pos" = instruction text in the Feedback Window. "Posfeed" = positive feedback text anywhere on the screen. Latencies are in seconds.

Table 5

Performance Data from Experiment 2- Averaged over Sections 1 and 3.

	Scenario		
Dependent Measure	1	2	3
Completion Time	754 (248)	537 (146)	439 (122)
# Errors	7.9 (5.1)	4.9 (3.6)	3.0 (2.7)

Note: Values in parentheses are standard deviations. Completion time is measured in seconds.

Figure Captions

Figure 1. Structural representation of the Stat Lady Descriptive Statistics Tutor: Data Organization and Plotting Module (Shute & Gluck, 1994).

Figure 2. Screenshot of the Number Factory.

Figure 3. Scenario completion time data from verbal protocol participants. Data are averaged over Sections 1 and 3. The graph is a cumulative subtraction of dependent measures as one moves from top to bottom. “Total Time” = Total scenario completion time; “Error-Free Time” = completion time computed after Error Time is subtracted from Total Time; “Curr.Obj. Time-Coarse” = completion time computed after time in peripheral tutor activities (Context and Number Factory) is subtracted from Error-Free Time; “Curr.Obj. Time-Coarse” = completion time after Lost Time and Reading Time are subtracted from "Coarse" time.

Figure 4. Skeletal representation of the Stat Lady interface and sample text.

Figure 5. Verbalization rates for all four text types. Data are averaged over Sections 1 and 3.

Figure 6. Scenario completion time data from TRAIN participants. Data are averaged over Sections 1 and 3. The graph is a cumulative subtraction of dependent measures as one moves from top to bottom. “Total Time” = Total scenario completion time; “Error-Free Time” = completion time computed after Error Time is subtracted from Total Time; “Curr.Obj. Time-Coarse” = completion time computed after time in peripheral tutor activities (Context and Number Factory) is subtracted from Error-Free Time; “Curr.Obj. Time-Coarse” = completion time after Lost Time and Reading Time are subtracted from "Coarse" time.

Figure 7. Comparison of model latencies with verbal protocol participant latencies.

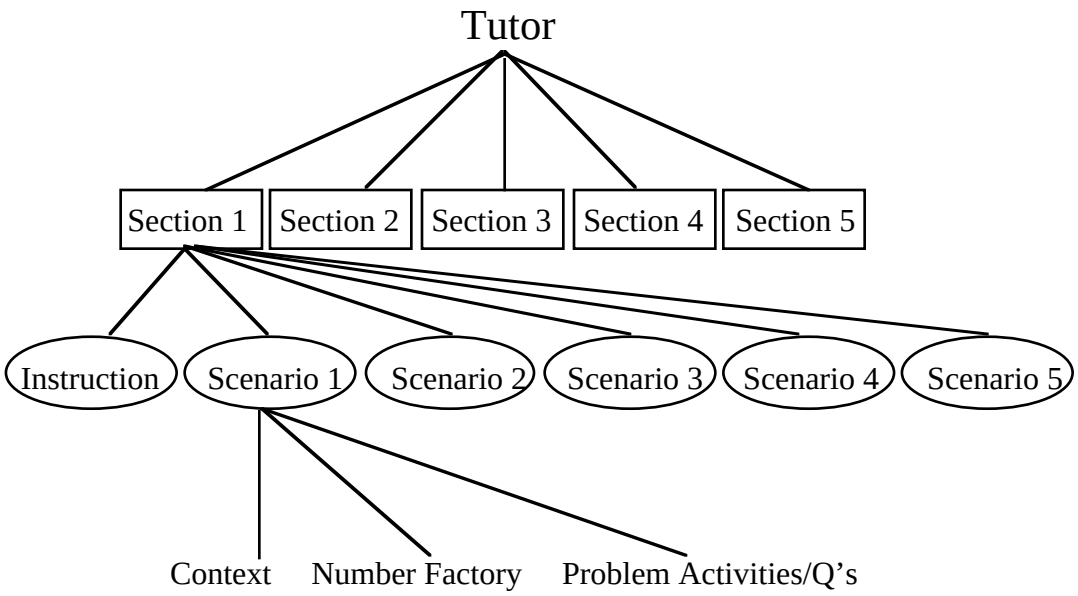
Tutor
Section 1
Section 2
Section 3
Section 4
Section 5
Instruction
Scenario 1
Scenario 2
Scenario 3
Scenario 4
Scenario 5
Context
Number Factory
Problem Activities/Q's

Figure 1.

Stat Chat Window
Let's say you were recently promoted to Chief of Operations at a huge amusement park. You're concerned about the park's liability because of a new roller coaster named Screaming Death Rocket.
In order to determine whether the new ride is too dangerous for continued use, you choose to make a frequency distribution of the number of injury complaints that are related to the "Death Rocket" every day for two weeks (14 days).
Step 1: Go to the Number Factory and get data with the following parameters:
N = 14 days
MIN = 1 complaint
MAX = 20 complaints
Step 2: Ship the Data to the Worksheet.
Quit
Prev. Page
Next Page
NP
Number Factory
Grab a Graph
Dictionary
Formulas
Calculator
Number Factory
Distribution : Uniform Normal Poisson
Number Type : Integer Real
N : 14
MIN : 1
MAX : 20
Number Sorting: Random Ascending Descending
Get Data
Destination : None Worksheet Problem Graph
Ship Data

Figure 2.

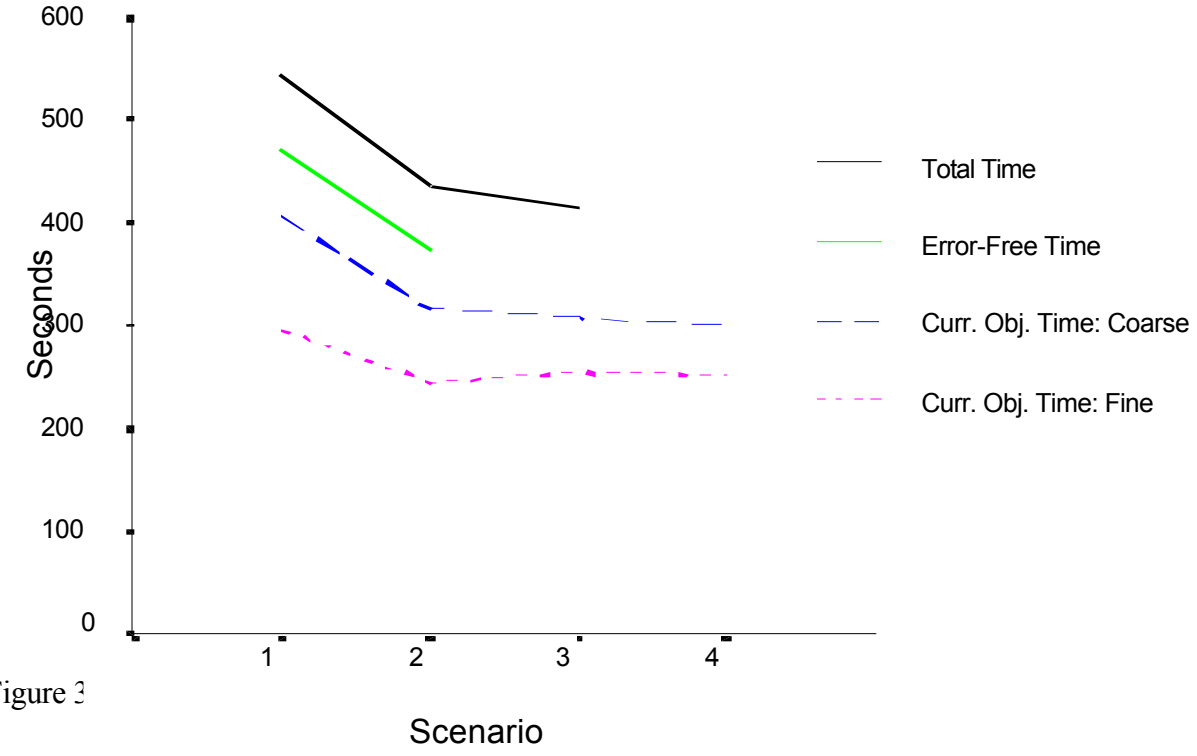
600
500
400
300
200
100
0
Seconds
1
2
3
4
Scenario
Total Time
Error-Free Time
Curr. Obj. Time: Coarse
Curr. Obj. Time: Fine

Figure 3

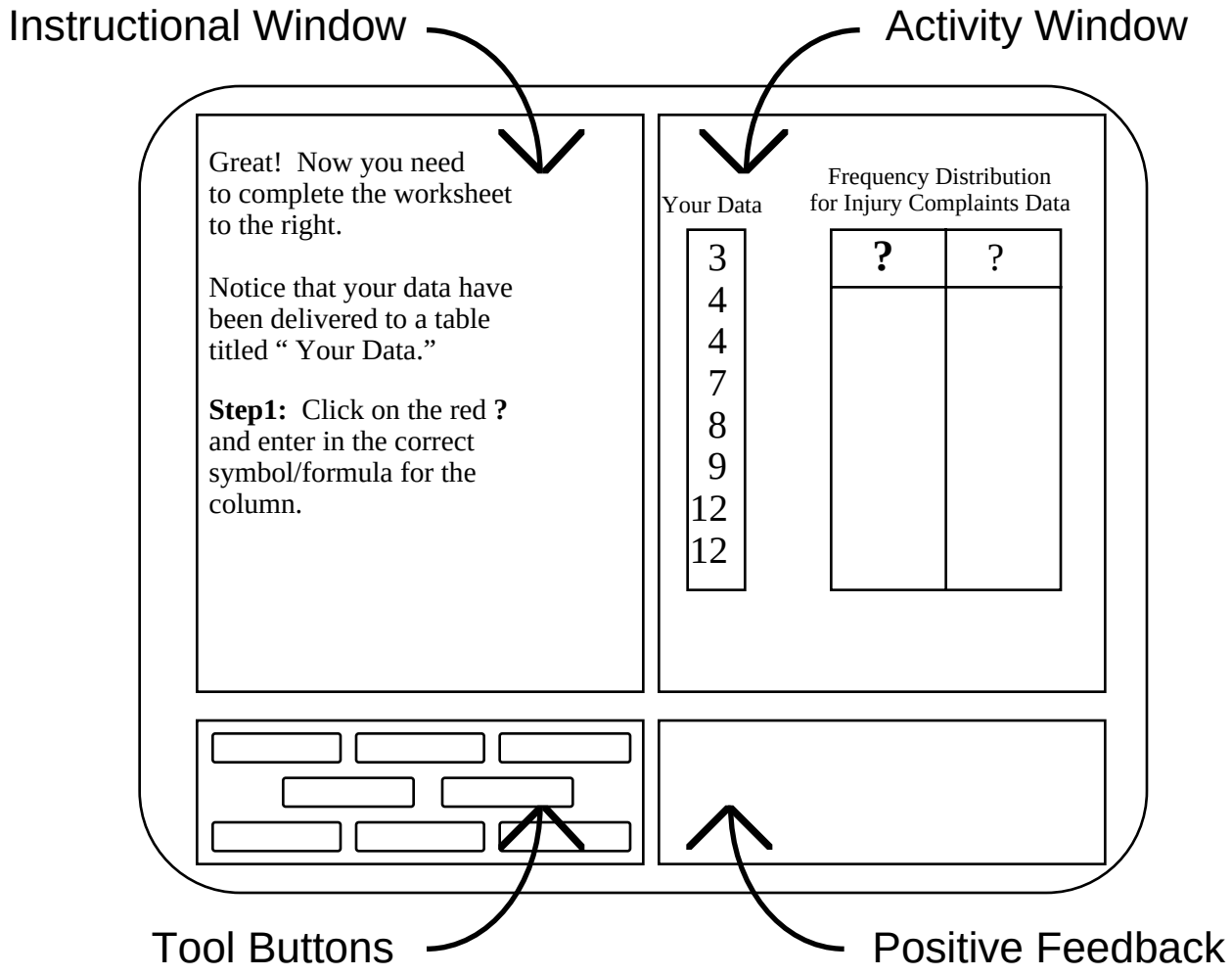
Instructional Window
Activity Window
Great! Now you need to complete the worksheet to the right.
Notice that your data have been delivered to a table titled " Your Data."
Step1: Click on the red ? and enter in the correct symbol/formula for the column.
Your Data
3
4
4
7
8
9
12
12
Frequency Distribution for Injury Complaints Data
?
?
Tool Buttons
Positive Feedback

Figure 4.

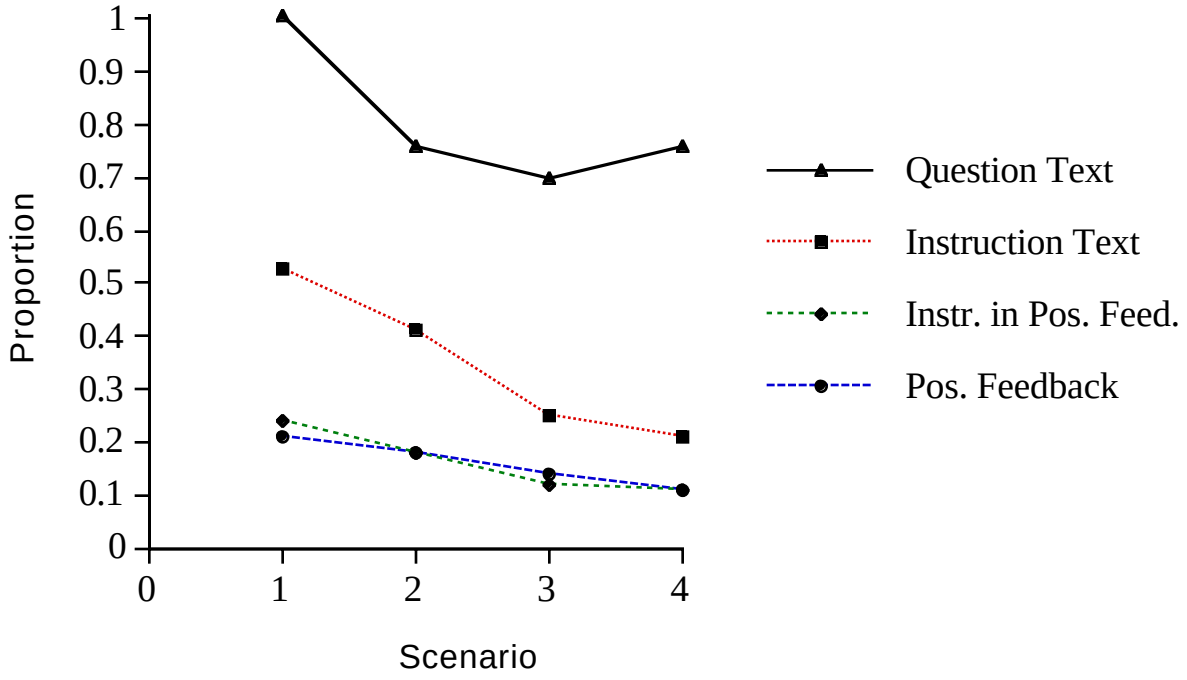
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Proportion
0
1
2
3
4
Scenario
Question Text
Instruction Text
Instr. in Pos. Feed.
Pos. Feedback

Figure 5.

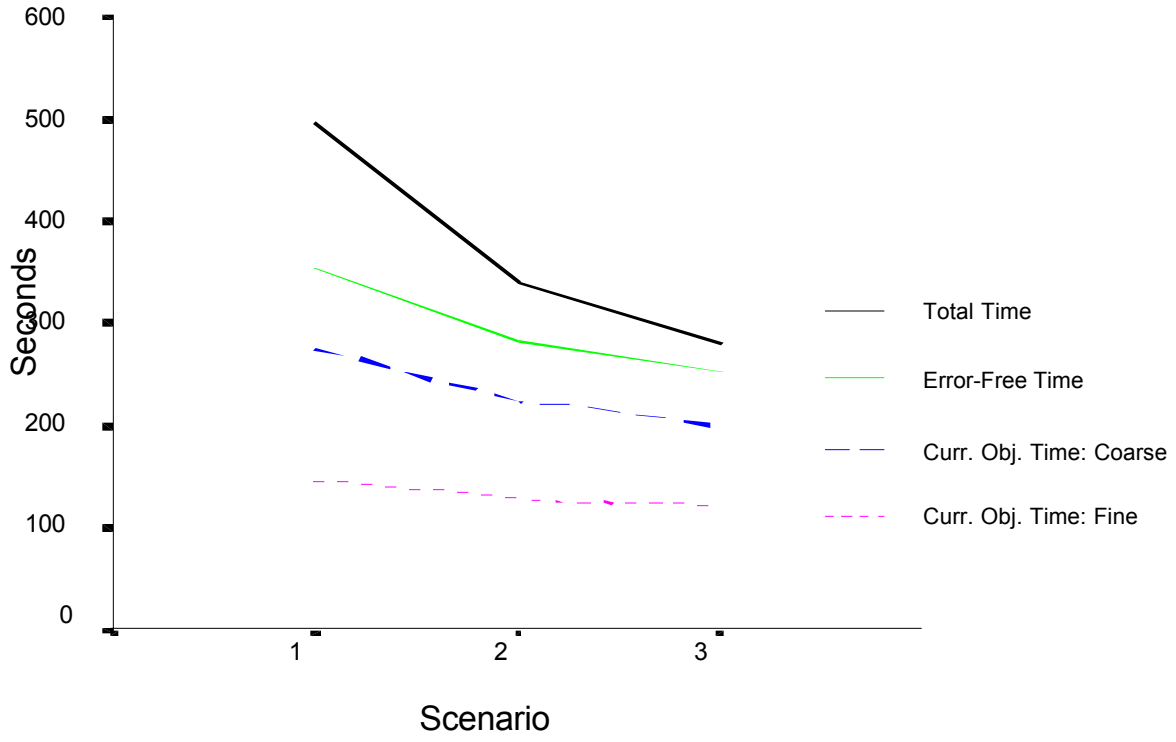
600
500
400
300
200
100
0
Seconds
1
2
3
Scenario
Total Time
Error-Free Time
Curr. Obj. Time: Coarse
Curr. Obj. Time: Fine

Figure 6.

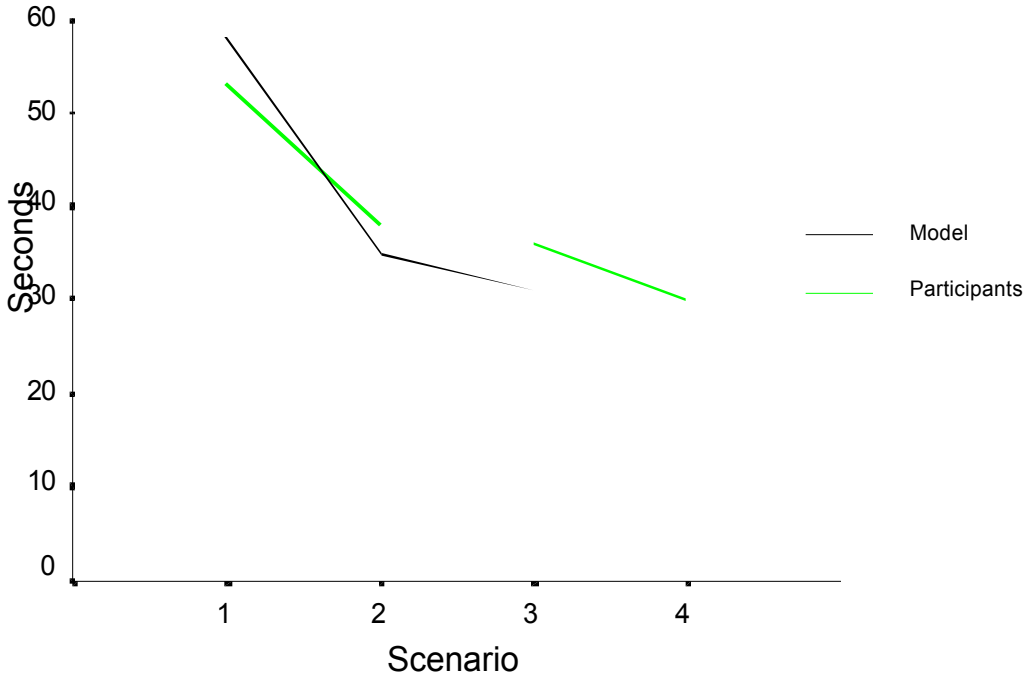

Figure 7.