A model for assessing student interaction with educational software

STEVE COHEN, FRANK TSAI, and RICHARD CHECHILE
Tufts University, Medford, Massachusetts

We present a simple model of generative learning that permits us to define four kinds of interactions and a system for tracing and recording how students use educational technology. We believe that this model will maintain a link between interaction and learning, thus providing one method for the assessment of a wide range of educational technology environments. Two results are presented from an evaluation of ConStatS, a program for teaching conceptual understanding of probability and statistics. The results illustrate the kinds of insight into generative learning that a detailed trace method can provide.

One of the most widely cited reasons for using computer technology in education is to help permit students to interact meaningfully with ideas and learn generatively (Cohen, Smith, Chechile, & Cook, 1994). The focus on education through interaction has taken on many forms, not all of which address instructional technology. Research on interaction has often been cast in process–outcome models. For instance, process–outcome models of teacher and classroom interaction have yielded insights into the effectiveness of instructional pace and the influence of teacher expectations (Brophy, 1986). Many of the results depend on the profile of the class and are not universally effective. Process–outcome models have also been used to investigate reasoning skills through performance on verbal analogy and classification problems (Alderton, Goldman, & Pelligrino, 1985). The models have been effective at isolating process differences between the most and least successful subjects, as well as providing evidence that common or similar processes are responsible for skills across domains.

Much of the research on student interaction with instructional technology has made use of specific learning models that conditionalize responses on student input. Dede (1985) describes seminal examples of such programs (e.g., Buggy and Debuggy) which are used for teaching subtraction and for diagnosing execution problems. Park and Tennyson (1983) discuss several models, including a Bayesian model of concept generalization that selects subsequent problems and examples on the basis of student interactive histories with the programs. In each case, the model made specific use of the interactions and generated a systematic but limited set of responses.

However, a good deal of the technology used in instructional settings today does not make use of highly specific learning models, but offers students environments for exploring ideas through experiments and open inquiry. This is true both in the areas of mathematics and science teaching, especially in statistics and physics (Cohen, Smith, et al., 1994; Laws, 1991) and in reading instruction (Horney & Anderson-Inman, 1994). Typically these environments permit students to construct experiments that either confront or develop their existing understanding. Despite the consensus that these kinds of interactions help students investigate ideas by focusing their attention and permitting them to learn generatively, it is not clear what kinds of interactions are most useful in a variety of educational contexts. Even when instructional technology has proven to be effective by global measures, individual differences in the way students use the program can help explain why some students gain more from technology than others. Subtle differences revealed by tracing student use may be useful for identifying “hands on, minds on” students from those who are “hands on, minds off.”¹

Tracing as Part of an Evaluation

To help address these problems, we have devised a simple model of generative learning that permits us to define four kinds of interactions, along with a system for tracing and recording how students use educational technology. The traces were intended in order to accomplish three goals. The first goal was to help us execute a large-scale evaluation of ConStatS, a package used for teaching probability and statistics.² (See Cohen, Smith, et al., 1994, for a discussion of the evaluation of the students’ understanding.) All concepts tested were taught by the software. Since each class and student in the experimental group used the software differently, it was crucial to keep track of which parts of the program were used. Thus, tracing is a tool for understanding how students used the software. The second goal of the evaluation project was to assess the effectiveness of the software in fulfilling its aim and to reveal ways in which the software could be

The research supported in this article was supported in part by the Fund for the Improvement of Postsecondary Education (FIPSE), Grant 116AH70624. The authors would like to thank Durwood Marshall for statistical consultation and both Denise Aronson and Kirk Israel for documentation preparation. Correspondence concerning this article may be sent to S. Cohen, Curricular Software Studio, Tufts University, Medford, MA 02155 (e-mail: scohen@emerald.tufts.edu).

made more effective. The long-term aim was to develop a more effective product; and one cannot always tell beforehand which factors might be limiting effectiveness, or what it is best to do to counteract them. Consequently, nearly all of the interactions were recorded. The third goal was to determine whether the students were engaged in the types of activity that lead to increased understanding. Since the questions designed for the evaluation were mapped to specific parts of the software, we could determine whether certain interactions and patterns of behavior led to increased learning.

Despite the fact that the tracing method came out of a detailed evaluation of a single application, ConStatS, we believe it to be generalizable. ConStatS uses several different kinds of pedagogical environments. Tracing has been done successfully in more limited environments (Horney & Anderson-Inman, 1994).

A Description of the Trace Method

The method is based on an analysis of the kinds of cognitive and behavioral events that take place while students interact with different educational programs. This analysis led to a simple model of the way in which students interact with educational software. The model includes four basic kinds of interactions. The first kind consists of *scaffolding* interactions. These allow the student to build up an understanding of the context surrounding a specific concept so as to pose meaningful questions about the concept itself. For example, the student engages in this kind of interaction when reading or retrieving text, or when selecting a part of the context used to illustrate the concept. Such interactions serve as cognitive preparation for when the student manipulates the concept directly.

The second kind of interaction, an *investigation*, consists of interactions that permit the student to manipulate the concept. Investigations include experiments and simulations, since they typically result in an outcome that puts the student in a position to clarify her/his understanding of the concept. Investigations also include interactions such as animations and processes, which do not necessarily involve an active, behavioral interaction (i.e., clicking a button), but which put the student in a position to construct a new understanding. Investigations provide students with the direct means of understanding the intended concepts.

The third basic kind of interaction is called a *reformulation*. This kind of interaction represents the cognitive change or reconciliation that begins to take place immediately after an investigation and continues until the next interaction. It represents the crucial moment during which the student either understands or fails to understand a concept.

The final interaction category is *navigation*. Interactions that take a student from one part of a program to another are assigned to this category. We call these separate sections *screens*. Although this interaction may seem simple, it requires that the program be categorized into screens, with each one in some way separate and distinct from the other. Each program in ConStatS was divided

into a set of screens and was labeled by number. Subtle changes or updates to screens—either user or program driven—may require a new entry into the taxonomy.

Depending on the pedagogical style of the technology, identical interactions might be categorized differently. For instance, in a hypertext document, an investigation might involve moving to and reading a passage of text linked to a previous passage (or image). Here, reading the passage might be the agent of reformulation. In a more generative environment, accessing and reading the same passage may instead constitute cognitive preparation or scaffolding. The pedagogical design of the technology determines just how interactions are mapped to the separate categories.

Implementing Traces

Each of the interactions—scaffolding, investigation, reformulation, and navigation—maps to a set of specific traces (Cohen, Chechile, Smith, Tsai, & Burns, 1994) that may be embedded in a program or set of programs. ConStatS was written in Microsoft C with the Microsoft Windows Software Development Toolkit, and the traces were embedded in each program in C language. To illustrate how traces are assigned, three examples from ConStatS are included.

Before beginning to work with an instructional program, each student enters his/her name and identification number. This information, along with the time and date, is placed at the start of an ASCII file that stores the traces for a session with ConStatS. Up to three students may be assigned to a set of traces. Each session is stored in a separate trace file.

Figure 1 shows a screen from the Probability Distributions program designed to teach two concepts: (1) how probability distributions represent probability and (2) the relationship between a given probability density function (PDF) and the corresponding cumulative density function (CDF). When a student arrives at the screen, a specific trace called a screen survey (an instance of a scaffolding interaction) is recorded. This trace records the program used (No. 6), the specific screen number (No. 5), the amount of time since the previous interaction, the number of options, and the length of the introductory text displayed at the top. The number of options and length of text serve as a simple measure of screen complexity. Recording the program number is important, since Microsoft Windows permits more than one program to execute at a time, and all other ConStatS programs have a screen No. 5. The trace may also include specific information, unique to the trace. For instance, the kind of probability distribution might also be included in the screen survey trace. Each trace in ConStatS can accommodate six unique units of information. The ability to tailor a trace to the unique interaction context permits us to capture a detailed record of student behavior. The necessary information to address domain and experiment specific hypotheses is likely to be different in each context.

Selecting a Why or Help (lower left in Figure 1) at this point would constitute a second scaffolding interaction,

Figure 1. This screen, from the Probability Distributions program, gives an example of an instructional environment designed for generative learning. Students can investigate how probability distributions represent probabilities by clicking on either the PDF or the CDF, defining an interval, and learning the resulting probability.

an information retrieval. Selecting either of these causes a text window to appear on the screen, though it does not cause a change in screen and new screen number. When the student closes the text window, a navigation (close) trace is executed, followed by a new screen survey.

Tracing experiments may get intricate, as in the case of this screen. Students may perform two basic experiments on this screen, the most prominent being defining an interval on the distribution by clicking below the abscissa on either the PDF or the CDF. Each time a student defines a new interval, it constitutes a new experiment and is entered into the trace record. For each of these experiments, six unique parameters are recorded: the kind of distribution, the two x values, the y values on the CDF, and the probability defined. After each experiment, a reformulation trace is recorded—specifically, a result survey. Assessing reformulation with the traces involves considering the time spent in studying the result and the kinds of subsequent interactions performed by the student. This kind of reformulation assessment can provide insight into the temporal progress of the student's understanding.

Every trace is not immediately written to a file on a hard disk or server. In many instances, the traces may be generated very quickly, and the traffic created by the tracing might require excessive input/output processing. Instead, traces are stored in a memory buffer large enough to accommodate 10 individual traces. Every 10th trace requires the contents of the buffer to be stored in a trace file. To help reduce both the processing burden on the instructional modules and the risk of losing trace data stored only in a memory buffer, all traces are sent to an

independent Windows program designed to temporarily store and output the traces.

A second environment is the graphical user interface program in Figure 2. In this environment, there are few options for building mental scaffolding. The menu options (and all suboptions) are investigations, and include Displays, Statistics, and so forth. Everything is traced in much the same way, with specific scaffolding, investigation, and reformulation traces capturing the use of the software. But unlike the preceding example of the Probability Distributions exercise, in this kind of an environment any mapping between on-screen interaction and cognitive events is not nearly as clear. Students can at any time be processing information relating to any number of experimental displays, irrespective of the order in which they were created, and inferences about how one interaction may have set the conceptual stage for another cannot be made with precision.

Finally, the trace method may be used in a simulated experiment, such as the Shepard-Metzler mental rotation experiment. In this instance, the traces may be used to record how students design experiments, yielding insight into untutored intuitions about experimental design.

Collecting Trace Data

The trace data were collected during a 3-year, multi-site study of ConStatS. The evaluation occurred during the 2nd and 3rd years, with improvements to the software implemented after the 2nd year. Overall, 20 different introductory statistics and research methods courses, with 739 students, participated in the evalua-

Figure 2. This is a basic screen with pull-down menus. Students can interact with objects directly. It becomes difficult to determine what interactions lead to successful experimentation when the screen becomes filled with a variety of objects.

tion. Six hundred and fifty-nine students from 16 classes participated as experimental subjects who used the software. The courses were taught in seven separate disciplines: psychology, economics, child study, biology, sociology, engineering, and education. Sixteen of the classes (621 students) were taught at Tufts University, and 4 courses (118 students) were at outside colleges and universities.

Results

An extensive and detailed analysis of the trace data is beyond the scope of this paper. The two results presented are intended to provide an example of the kinds of insights that the trace method might generate. Both results focus on a specific interaction that correlates with increased learning, and on the general patterns of interaction that characterize students who execute the interactions. The results were predicated on hypotheses formulated before the evaluation was executed and based on the following three assumptions. First, certain interactions are likely if a student is learning by active experimentation. Second, these interactions indicate that the student has a relatively well-formulated question in mind. Third, the goal of software designed for generative learning is to put students in a position to pose these questions and perform the interactions.

The first result involves student investigations of discrete probability distributions occurring on the screen displayed in Figure 1. ConStatS offers students a choice of four discrete distributions: Uniform, Binomial, Geometric, and Poisson. We examined the trace data for all instances of a student investigating one of these distributions. The initial hypothesis predicted that students performing an experiment (defining a probability interval) that yielded a zero probability would do better on the question in Figure 3.

This is a difficult question for most students. Merely to execute an experiment that yielded a zero probability was not adequate for them to answer this question correctly. However, students performing three consecutive experiments with successive nonzero, zero, nonzero results gave correct answers 46% of the time, whereas other students gave correct answers only 10% of the time. Students performing this sequence also distinguished themselves in two additional ways. They averaged about twice as many total interactions as did other students over the same period of time, and used the Why option twice as often as did other students. Finally, the median time for performing the nonzero, zero, nonzero experiment was under 2 min. Hence, students typically executed the experiment at an early opportunity, or they did not execute it at all.

A similar result emerged through examination of the trace data for the screen shown in Figure 4. On this screen, students can investigate the results of a sampling distribution.

The average, or expected value, of the probability distribution below is 13.5. What is the probability that a variable distributed this way will assume the average value?

Figure 3. This question was designed to test students' understanding after they had worked with the screen shown in Figure 1.

Figure 4. This screen, like the one shown in Figure 1, contains a specific subset of interactions that correlate well with learning a certain concept.

butions experiment. The histogram at the bottom right shows a distribution of 50 means from samples of size 10. The samples were taken from the population of Baby Weights shown center right. In order to help get a sense of how the variability of samples influences the location of a mean in a sampling distribution, students may click on any mean and see the sample from which it came. Figure 4 shows a sample mean in the first interval that was generated by Sample 13 in the upper left.

Student understanding of the concept was tested with the following question: "What, if anything, is misleading about the following claim? An increase in sample size will always cause a sample mean to move closer to a true mean." We hypothesized that students who investigated the distribution of sample means by clicking first on a high end mean and then on a low end mean—or vice versa—would perform better on this question. High end was defined as interval 9 or 10, and low was defined as interval 1 or 2. Students performing this interaction averaged 80% on the question, whereas other students averaged only 60%. Again, students performing the interaction had about twice as many interactions, and the median time for performing the sequence of experiments was under 50 sec. There was no difference in the number of times that the two groups accessed the Why option.

Discussion

A detailed assessment of instructional software should reveal what aspects of the software are effective and which are not. Instructional software tracing creates a link between a student's understanding, as measured by the

comprehension assessment, and their behavior in using the software. By indicating which interactions are most useful, tracing can help inform ancillary educational environments, such as workbooks and assignments, as well as guide the redesign of the software itself.

The detailed, context-dependent component of the traces offered the most useful insight into student interactions. Furthermore, the specific interactions described, along with the pace at which the students moved through the program, seem to offer a good indication of which students are actively engaged by the software. Analysis of the pace and specific interactions may permit identification of students who are in some way "hands on, minds off." The two measures together can be part of a nonintrusive, formative evaluation model for generative learning environments. They can be used to guide feedback, either by informing a system within the program, or by notifying an instructor at a remote site about a student's interactive progress. Through this way of thinking, the ability to encourage specific interactions then becomes the chief measure of a program's effectiveness. It is the responsibility of the program to put students in a position to pose the question and execute the interaction. Programs for generative learning need to be improved with respect to this goal, and the traces offer a means to begin doing just that.

REFERENCES

- ALDERTON, D., GOLDMAN, S., & PELLIGRINO, J. (1985). Individual differences in process outcomes for verbal analogy and classification solutions. *Intelligence*, 9, 69-85.

- BROPHY, J. (1986). Teaching and learning mathematics: Where research should be going. *Journal for Research in Mathematics Education*, **17**, 323-346.
- COHEN, S., CHECHILE, R., SMITH, G., TSAI, F., & BURNS, G. (1994). A method for evaluating the effectiveness of educational software. *Behavior Research Methods, Instruments, & Computers*, **26**, 236-241.
- COHEN, S., SMITH, G., CHECHILE, R., & COOK, R. (1994). Designing software for conceptualizing statistics. In *Proceedings of the First Scientific Meeting of the International Association for Statistics Education* (pp. 237-245). University of Perugia.
- DEDE, C. (1985). *Intelligent computer assisted instruction: A review and assessment of ICAI research and its potential for education*. Cambridge, MA: Educational Technology Center.
- HORNEY, M., & ANDERSON-INMAN, L. (1994). The ElectroText project: Hypertext reading patterns of middle school students. *Journal of Educational Multimedia & Hypermedia*, **3**, 71-91.
- LAWS, P. (1991). Learning physics by doing it. *Change*, **23**, 20-27.
- PARK, O., & TENNYSON, R. (1983). Computer based instructional systems for adaptive education: Review. *Contemporary Education Review*, **2**, 121-135.

NOTES

1. The phrase "hands on, minds off" was first introduced to the lead author by Candace Schau at the 1994 American Educational Research Association meeting in New Orleans.

2. ConStatS consists of three major parts, each of which contains several distinct programs that cover different topic areas:

Data Representation: Displaying Data, Summary Statistics, Transformations, Describing Bivariate Data.

Probability: Probability Measurement and Probability Distributions.
Sampling: Sampling Distributions, Sampling Errors, A Sampling Problem.

The nine programs together provide 15-20 h of curricular material. Three additional programs are under development.

(Manuscript received November 18, 1994;
 revision accepted for publication January 27, 1995.)