THE FATHOM EXPERIENCE—IS RESEARCH-BASED DEVELOPMENT OF A COMMERCIAL STATISTICS LEARNING ENVIRONMENT POSSIBLE?

William Finzer
KCP Technologies
USA

Research on learning and commercial software development competes strongly for a project's scarce resources, and yet they have widely overlapping goals. If they could be made to coexist, their synergy could improve both processes. On the research side, to use software to help understand how students perceive and learn statistical concepts requires a software platform that is stable, easy for students to use, and flexible enough to allow different models to be tried; that is, the research benefits from a smoothly functioning development process. On the development side, there is great need for insight into the learning process to inform the software design, and need for research methods to test whether any given design works with students and improves their statistical understanding.

INTRODUCTION

In some ideal world the boundary between development of software for use in research on learning and the development of software for classroom use would be truly porous so that researchers could easily adapt classroom software for research purposes and classrooms would reap the benefits of educational research in the form of software based on research principles and results. In the real world this boundary is barely permeable: little research-based software development has influence beyond the particular research project, and software used in classrooms, such as spread sheets and word processors, has been designed with other uses in mind. Why does this situation exist and what can be done to change it? We illustrate the problems and possibilities of research-based software development as experienced in the development of Fathom Dynamic Statistics™ Software (Finzer, 2001).

WHY IS THIS IMPORTANT?

Let us make an artificial distinction between educational software developers and education researchers—artificial because many educational software developers do research and many researchers are also engaged in software development. Even when the people engaged in each enterprise are the same, the tasks themselves are different.

FOR SOFTWARE DEVELOPERS

Educational software development labors under constraints that make most commercial software development seem trivially easy. Users must be able to begin using the software to do something productive in a very short time, on the order of a class period. Unlike users of spreadsheets and word processors, users of educational software may spend a total of only 10–20 hours using it. The content domain in which they are working is, by the very nature of the learning experience, something they are unfamiliar with, and the end result of use must be more than mere accomplishment of a task, namely increased understanding of concepts that the user may or may not be motivated to learn.

Combine inherent difficulty of task with miniscule budgets and low probability of commercial success, and you have a recipe for underachievement that goes a long way toward explaining the relatively low classroom utilization of software even in statistics, a discipline built on the assumption of omnipresent computing power to carry out even the simplest of tasks.

Faced with these difficulties, educational software developers, though they may not realize it or be willing to admit it, desperately need the assistance of education researchers to help them understand what works and what doesn't work, what conceptual difficulties can benefit from software assistance and how to provide it, and how to influence the educational practice of a community singularly unaccustomed to change.

FOR EDUCATION RESEARCHERS

Software learning environments can serve as powerful tools for researchers to help them gain insight into students' learning processes, especially if researchers can modify the software to test conjectures (Cobb, 1999). But building learning environments from the ground up is an expensive, time consuming process that will drain the resources of most research projects. Those projects successful in building malleable software research tools have to decide at the end of the project what to do with the tools in which so much effort has been invested—throw them away, or invest more time and money in a risky attempt to create classroom-usable software?

Research stands to benefit from the ability to tap stable development platforms for rapid prototyping of alternative software designs to use in testing hypotheses. If the most effective of these designs can be incorporated into software destined for classroom use, the results of the research have a chance to influence students' learning.

EXAMPLE OF RESEARCH AND DEVELOPMENT INTERACTING BENEFICIALLY

What does a beneficial interaction of research with software development look like when it works? Here we describe a thorny problem in the development of Fathom and the process that eventually yielded a solution.

We start with an activity (Erickson, 2001) well-suited to teaching about inference at the introductory statistics level. The scenario involves small groups of students working as the statistics department for Orbital Express (OE), the delivery company that drops its packages on their destinations from orbit. The engineering group for OE has developed two new designs for re-entry vehicles. The students-as-statisticians' task is to decide which of the two designs performs better.

The first design consists of a crumpled up paper towel and the second of crumpled up copier paper. A test consists of dropping the vehicle from some height on a coin (representing the package destination) lying on the floor. The outcome of a test consists of a measurement of the distance from the paper to the coin. Each test is expensive and OE has only allocated enough money for a total of 14 tests.

Students must decide on, among other things, from what height the drop will be done, the order of the tests, which group member or members will do the dropping, and whether distance will be measured to the vehicle's initial strike position or to its final resting spot. The data for a complete experiment usually consists of 7 distances for the paper towel and 7 for the copier paper. Students write each measurement on a piece of paper and arrange these along two sides of a measuring tape as a kind of concrete dot plot as shown in Figure 1. For most groups the spread and overlap of the measurements will be such that students have reasonable doubt about whether one design is better than the other.

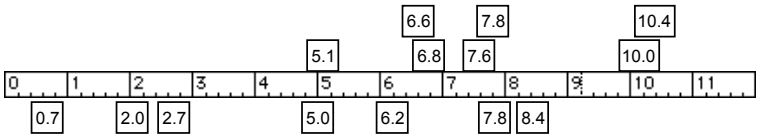

Figure 1. Concrete Dot Plot.

Students are asked to come up with some "measure" of how different one vehicle design is from the other. This measure must be a single number that (typically) is larger when the difference between the two sets of results is great and smaller when the difference is not so great. Measures students have used include the absolute value of the difference of the median (or mean) distances, the maximum distance of the copier paper minus the maximum distance of the paper towel, and the difference of the standard deviations of the distances.

The difference of medians for the data shown in Figure 1 is 2.6, in favor of the paper towels. Is this a number that can be explained by chance alone? How likely would it be, if there were really *no* difference between the two vehicle designs, that we would have obtained 2.6 as the difference of medians, or even a value greater than that? To answer this question (for their measure), students are asked to shuffle all 14 of the cards and then deal them out randomly on the two sides of the measuring tape. They compute a new measure, one whose value cannot be due to a difference in the designs because only chance determined on which side each measurement fell.

Repeating this "scrambling" of the cards and recomputation of the measure of difference gives students the beginnings of a distribution and a sense of where the measure for the experiment lies in that distribution. They soon realize, however, that a computer would be helpful in building up the rest of the distribution. Providing an intuitive tool for students to use in this and similar situations was a central design goal for Fathom, one whose solution posed major problems during the entire course of development.

DESIGN PROBLEM OR RESEARCH PROBLEM—TWO PERSPECTIVES

A software developer is likely to view this problem as one of design—what software capabilities and interface for them will allow the user to map the situation to what they see on the screen? The solution will come about through repeated cycles of interface design, prototyping, and trials. The software developer has a theory, a *user model,* that will be used to predict what modifications to the current prototype of an interface is most likely to increase its *usability.* Observations of users, who range from members of the software development team to students in classrooms, are used to modify the theory, and the theory is validated when predictions about user behavior are confirmed. Understanding of how students view the situation and how they learn may be viewed as tools for development of software that will facilitate learning.

An education researcher will be more likely to center investigation on the learner—how do learners approach the situation *without* software tools, and how can we make their approach concrete in the software? A researcher may seek to understand the learning process as it takes place in students in order to discern obstacles to learning that may be lowered through use of the software. This *theory of learning* will suggest ways that information should be displayed and metaphors to embody in the user interface of the software. The software may be viewed by the researcher primarily as a tool for increasing understanding of the learning process.

These two perspectives are complementary, and, of course, a single person or team may easily hold both of these perspectives at different times. The shift between thinking of a user model and thinking of a theory of learning can be subtle. At one moment we ask, *Should the user begin the scrambling process by choosing a menu item or pushing a button?* The next moment we ask, *Does the student think of the experiment as one of an infinite set of possible experiments?* We go back and forth between the two as the need arises.

But even if the two perspectives are complementary and can appear in one and the same person or one take the place of another in a single meeting, a software development team without a research component will have a *vastly* different emphasis than a team focused on research and lacking a software development goal. The software development team will, if successful, contribute to the tools available for learning, but will have missed the opportunity to contribute to our understanding of how learning takes place, and its product will have likely failed to fulfill its full potential. The research team, if successful, will add to our knowledge, but the software tools developed along the way will likely be discarded and the possibility of embodying the new knowledge in concrete form widely accessible to students and teachers greatly diminished.

SOLVING THE DESIGN PROBLEM

We return to the students engaged with the Orbital Express data attempting to decide whether chance alone can account for their observations. They would like the computer to repeatedly scramble their experimental values and compute a measure of difference for each scramble. As a software development team we grappled over a five year period with how best to provide an interface for scrambling and other computer intensive methods for doing inference.

Resampling—content confusion and clarification. We were amazed to discover just how confused we ourselves could become as we attempted to work through statistical inference problems using computer intensive methods. This confusion taught us much about students' confusions, introspection being an important source of research ideas. Our introduction to resampling techniques occurred in 1994 during an advisory meeting in which Cliff Konold demonstrated the workings of DataScope (Konold, 1995), a tool he had developed from a research perspective and made commercially available for classroom use. As we delved into the literature of resampling and other computer intensive methods (Efron & Tibshirani, 1993), we

became completely enamored of the ideas, seeing in them ways to cut through the morass of statistical recipes to reach the essence of statistical inference.

Had we been a research team perhaps we would have turned to the students at this stage to study their understanding of inference and the effect of introducing a resampling perspective on their understanding. But, as software developers, we had a different job.

The era of the dialog box. Many problems in software design can be solved in a very straightforward manner by introducing a modal dialog box. The trouble is that the "solution" may literally cover up the problem. When a dialog box comes up on your screen, the thing you were looking at, and hoping to change, is likely to be underneath. Suddenly new choices appear, vying for space in your mind, choices that must be made before you can go on. We reasoned that such an intrusion could hurt the learning process, and, over time, we became committed to banishing modal dialogs from Fathom.

This stage serves to illustrate that there are interesting areas of research *outside* the scope of research appropriate to educational software development. Our decision to eschew modal dialogs was based on research in human computer interface design (Laurel, 1990; Smith, 1982).

Wiring diagrams. Early designs for Fathom frequently relied on "wires" that would show the user how data flowed from one place to another. Without a wire, how would the user keep track of the source of data such as the scrambled data? Flow diagrams ran rampant in our design documents. It seemed inevitable that we would end up with wires showing how one thing is "connected" to another.

It is interesting to speculate on how the development process would have gone at this juncture if we had been collaborating with a research team. Investigations into the research question, "What is the learner's conception of data?" might have shed considerable light on the question of whether wires were a necessary or helpful piece of the interface.

In the end we never incorporated wires, and we were surprised that users could make sense of things anyway. As will be explained in more detail, it has turned out that careful naming plus animation has supplied representation of data flow users require. But there is room for study here—how do users perceive the connection of one object to another in Fathom, and, especially, how do they conceive of the *process* inherent in statistical inference?

Confusing process with object—moving control out of the document. Should there be an object in the document whose function is to scramble the collection of experimental measurements and produce measures? It seemed at first that there should. We called them Samplers, Scramblers, and Collectors. Figure 2 shows a design for a Sampler similar to one Tim Erickson field tested in a statistics class engaged in the Orbital Express activity. The afternoon he returned from the classroom was memorable for despair. *They can't use it. It's too hard*, he stated flatly, leaving us perilously close to Fathom's release without a working design for one of its fundamental capabilities.

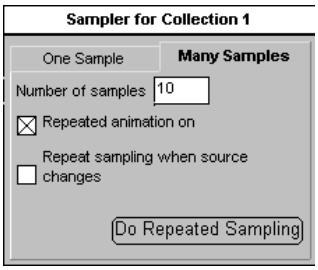

Figure 2. A Design for a Sampler.

A field test is a tool for research and a tool for software development. The software developer uses a field test to discover to what extent the software is usable by real users and how well it will stand up to prolonged use. A researcher goes into the classroom to gather information and test hypotheses. The two uses have a useful area of overlap so that a single field test may serve both purposes even better than it could serve one. As software designers we went back to the drawing board in a situation in which researchers might have delved more deeply into *why* students were confused by the interface. Working together we might have learned more and still come up with something that worked.

A basic principle of software design is KISS, "Keep It Simple, Stupid." The documents with Scramblers and Collectors were anything but simple. We decided to pull these controller objects out of the document and associate them more closely with the scrambled collections and the measures collections. A design principle emerged: "Controls for process don't belong in the document. They should float above it." Fortunately we already had a place to put controls—in a pane of a collection's inspector, an object with an already established place in the user model.

Inspectors. Each collection has an inspector that can be shown in a floating window. As shown in Figure 3, there are four *panes* in the inspector of an ordinary collection. One of those panes contains *measures,* a value, usually computed with a formula, that describes the collection as a whole rather than individual cases in it. A measure is perfect for defining the difference of medians statistic for the orbital express experiment.

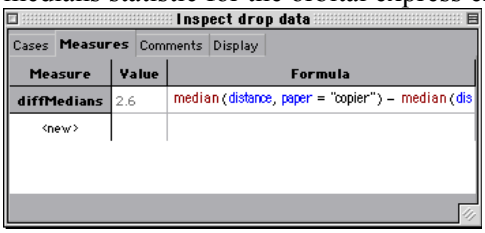

Figure 3. An Inspector.

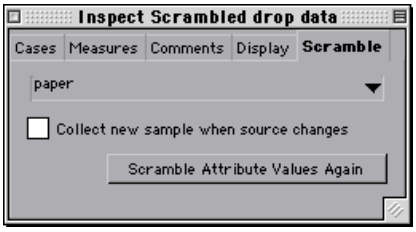

Figure 4. Pane for the Purpose of Controlling the Scrambling Process.

In a breakthrough for which Tim Erickson was largely responsible, we decided that in place of the metaphor of an object responsible for scrambling, the user would create a *scrambled collection* and its inspector would have a pane (shown in Figure 4) for the purpose of *controlling* the scrambling process.

From a design perspective, this idea was a breakthrough. From a research perspective, it might have represented the formulation of a new hypothesis of the form "Learners can easily associate the *origin* of a collection with the collection itself." Had we been collaborating with researchers, we might have begun an investigation of learners' perception of the origin of simulated data. Suddenly we could create an interface for the Orbital Express activity that held real promise. The final piece was a *measures collection* whose attributes were drawn from the measures of the source collection and which could trigger re-randomizations of its source each time it collected a new measure.

As a design team we were extremely pleased, but researchers collaborating with us would have been skeptical. Aside from our own introspective use, how were we to know that the new design was even better than the previous one, let alone a "good" design? As of the time of this writing, all we know is that comprehension of what is going on remains difficult for students.

Animation to Illuminate Process. Figure 5 shows a Fathom document in which the three levels of abstraction are represented as three collections: the experimental data, the scrambled data, and the sampling distribution of measures. The user can choose to have animation turned on or off while measures are being collected. With it turned on, the user sees the graph for the scrambled collection update each time the scrambling is done and a gold ball with an "m" on it moves from the scrambled collection to the measures collection to indicate that a new measure or set of measures has been collected. After the ball reaches its destination, the graph of measures updates.

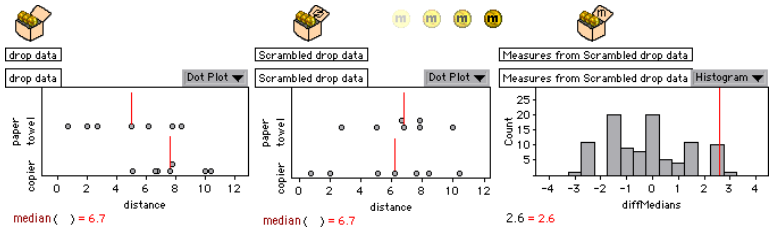

Figure 5. A Fathom Document (the three levels of abstraction are represented as three collections).

This design for animation emerged naturally from the case-centric user model of data in which, by default, each case in a collection looks like a small gold ball. It seemed important to us that the user be able to slow down any repeated process and that there by a visual representation

of the process itself, something in addition to the updating graphs and tables. We were not able, however, to conduct any user tests outside the immediate development team to compare the proposed design with any others.

What are the stages that learners go through as they attempt to understand the process of statistical inference? What are the sources of confusion and in what ways can software ameliorate them? To what extent does the simple animation in Fathom provide a mnemonic framework on which learners can hang their understanding of the statistical inference process? In a collaboration between software developers and researchers, there would be time and motivation to explore these questions, and the software itself would be a tool for investigation and be improved by the process.

BENEFITS OF SYNERGISTIC COLLABORATION

The thesis of this paper is that collaboration between education researchers and software developers holds great potential for both groups as shown in Table 1.

Table 1
Benefits to Education Researchers and Software Developers

Benefits to the education researcher	*Benefits to the software developer*
• Access to a programming team that can rapidly prototype alternative interfaces to be used in testing of research hypotheses. • An increased ability to channel research results into classroom practice. • By including members of the design team in the research process, to draw on their experience with user models and their ability to propose modifications to the user interface that may be effective in testing research hypotheses.	• The ability to obtain insights into students' learning process that can inform the software design process. • A design process more informed by evidence from the classroom than usual. • Ability to draw on researchers' understanding of learning to increase the pool of innovative design possibilities. • An opportunity to draw from evaluative research designs to determine the effectiveness of the software.

Why is collaboration between researchers and software developers rare? Both research and software development are already complex processes and collaboration inevitably makes them more complex. The more complex the process, the harder it is for either group to accomplish its goals. A newly funded project (Rubin, 2001) is designed in part to test the hypothesis that such collaboration can be fruitful for both software development and research, yielding research results and a mechanism for funneling these results into the classroom.

ACKNOWLEDGEMENTS

The work reported here was partially supported by the U.S. National Science Foundation SBIR grants III-9400091 and 0060710.

REFERENCES

Cobb, P. (1999). Individual and collective mathematical learning: The case of statistical data analysis. *Mathematical Thinking and Learning*, *1*, 5-44.

Efron, B., & Tibshirani, R. (1993). *An Introduction to the Bootstrap.* Chapman & Hall/CRC.

Erickson, T. (2001). *Data in depth: Exploring mathematics with fathom*. Key Curriculum Press.

Finzer, W. (2001). *Fathom dynamic statistics™ software*. Key Curriculum Press.

Konold, C. (1995). Issues in assessing conceptual understanding in probability and statistics. *Journal of Statistics Education*, *3(1)*.

Laurel, B. (Ed.) (1990). *The art of human-computer interface design.* Reading, Mass.: Addison-Wesley.

Rubin, A. (2001). Visualizing Statistical Relationships. Project funded by Research on Learning and Education, National Science Foundation, Award Number 0106654.

Smith, D.C., C. Irby, R. Kimball, & B. Verplank (1982). Designing the Star user interface. *Byte 7 (4),* 242-282.