Experiments in Learning Chemical Engineering Modeling Skills[1]

Noel Rappin, Mark Guzdial, Matthew Realff, Pete Ludovice
College of Computing/School of Chemical Engineering
Georgia Institute of Technology
Atlanta, GA 33039-0280/Atlanta GA 30032-0100
770 894-4650
{noel,guzdial}@cc.gatech.edu /{matthew.realff,pete.ludovice}@che.gatech.edu

ABSTRACT
Creating educational software forces a difficult tradeoff. The software must be easy for the students to use, but the tasks from which the students will learn the concepts must not be automated. DEVICE (Dynamic Environment for Visualization of Chemical Engineering) is a learning environment aimed at allowing chemical engineering students to model chemical engineering problems, then execute those problems as simulations. In the design of DEVICE, we have attempted to use student tasks to focus attention on the most important parts of the problem without overwhelming students with extraneous detail.

Keywords
chemical engineering, modeling, simulation, interface design.

INTRODUCTION

Creating educational software forces a difficult tradeoff. The challenge is deciding what parts of the problem are not necessary for students to learn and should be made easy, and what parts are absolutely critical such that the students should be allowed to struggle to gain mastery of the material.

Description of Problem

Computer simulation is an area in which the aforementioned tradeoff is particularly acute. Simulation provides students with realistic experience, even in domains where realistic activities are too complex to be performed by novices, too expensive to be offered in an undergraduate lab, or too dangerous to allow students to make mistakes. Our hypothesis is that merely watching a simulation is not enough to trigger learning: the student must have some hand in creating the model that drives the simulation. However, learning to build models in a fully functional simulation environment is equivalent to learning an entire programming language. It is too difficult to expect students to be able to do this in addition to learning the domain knowledge.

The problem, then, is to create a simulation environment which allows students to construct and execute complex simulations in a manner that allows them to connect their theoretical knowledge to the real world and which does not force them to learn formalisms that will not be of later use. A key component of this environment is support for modeling – creating a conceptual representation of reality.

The objective of our project is to create a computer environment, called DEVICE (Dynamic Environment for Visualization in Chemical Engineering) which will facilitate student learning through construction of equation-based models and evaluation of those models executed as simulations. We have several specific objectives that we hope to achieve with DEVICE.

- To teach chemical engineering modeling skills. Modeling is a skill that engineers use frequently in practice, but is rarely explicitly taught [1]. DEVICE is prefaced on the hypothesis that to support the learning of

[1] A version of this paper will appear in the ACM CHI97 Conference Proceedings.

modeling may require a different software architecture than that to support the professional practice of modeling.

- To familiarize students with the concept and role of degrees of freedom. Effective engineering design requires the introduction and manipulation of degrees of freedom. It is our hypothesis that students need help in transferring skills in modeling a fixed system to manipulating that model for design purposes.
- To impress upon students the role of a model and the potential gap between the real system behavior and the model. With increased enrollments in Chemical Engineering degrees the relative numbers of students that have any experience with chemical processing facilities is decreasing. Furthermore computers are all pervasive in engineering activities and a majority of this generation of students is "too familiar" with significant computational resources. Our hypothesis is that many students fail to perceive the difference between a model and the real system and "blindly" trust the results of a computer model.

Description of Students

The primary users of DEVICE are undergraduate engineering students. For the purposes of our initial studies, we have used sophomore students in the Chemical Engineering department at the Georgia Institute of Technology. These students are near the beginning of the Chemical Engineering course sequence. They are expected, however, to already have taken courses in physics, calculus and an introductory course in mass balances.

The students being used are, based on their average SAT scores, considered to be somewhat above the national average for engineering students. Not surprisingly, they tend to have much stronger math skills than language skills. They are expected to have used a Macintosh computer (the undergraduate facilities at Georgia Tech make Macintosh computers available), and to have had at least an introduction to the chemical engineering concepts being presented. Our analysis of student problem solving supports the perception that the students have difficulty with adapting theory to fit the real world.

Our initial problem domain is simple chemical engineering pumping systems. This domain was chosen because:

- It is encountered early in the course sequence.
- The equations that describe the system are simple compared to other pieces of equipment.
- There is a simple type of representative design problem - the selection of an appropriate pump.
- There are simple degree of freedom problems - such as the introduction of a valve to adjust pressure drop.

The primary equation in this system is called the mechanical energy balance. This equation balances the energy coming into the system consisting of kinetic, potential, and flow components and work done by a pump with the same components leaving the system and energy dissipated by friction. In order to correctly use this equation, the student must define a consistent system such that input and output values can be defined, and then specify variables in the equation such that it can be solved in conjunction with a number of other equations that define the system behavior (the characteristic curve of the pump). They may also be called upon to calculate auxilary quantities, such as the cross-sectional area of the pipe.

There are two basic problems in this domain. The first is determining the flowrate of a system that contains a given pump. The second is to choose an appropriate pump for a system that will result in a desired flowrate. Both problems use concepts of energy balance that the student will use frequently throughout his or her career.

What Students Don't Know

In order to measure student needs, we analyzed the performance of students solving modeling problems off-line. We gave students, in two separate courses, test problems that involved pumps. Both classes had learned about the behavior of such systems at differing level of detail, and the problem was tailored to expected student knowledge in each course. The two courses were both sophomore level chemical engineering classes, one was the second process analysis class taught at Georgia Tech that covers basic energy balance concepts, the other was the fluid mechanics class, which is the next class in the sequence. In the first they have exposure to the mechanical energy balance but not to fluid flow and friction, which is featured heavily in the fluid mechanics class.

Examination of student performance revealed that fewer than half of the students were able to apply the correct set of equations to the problem, while at least two-thirds made some mistake in their problem solving that indicated some misunderstanding of the connection between the real world problem and the equations. Student mistakes

tended to concentrate in three areas. First, students had difficulty defining a mathematical model of the system with consistent boundaries. Second, they frequently made assumptions in the mathematical model that reflected contradictory choices of boundary points in the application of the mechanical energy balance. Finally, students had trouble at places in the problem where it was necessary to convert between units, particularly dynamic units like flowrate and velocity.

The results of this evaluation suggested to us that students needed a system that would encourage them to examine their assumptions as they worked through problems in their domain.

Related work

There are two related domains of research. First, there are the generic attempts to support modeling developed in the educational field. These support environments are often not applicable to the needs of engineering students. Some environments, such as STELLA [9], use a notational constructs that are difficult for novice users to understand. This requires an additional overhead in learning concepts and syntax that are not connected to the theoretical understanding of the domain itself that students are building [12] [13]. Engineering students require environments with quantitative precision, unlike Model-It [8] [7], which uses a graphical environment that allows students to model an ecosystem. More generalized simulation or programming environments are too complex to expect students to learn along with domain content, and have no direct tie to that content. Examples of such systems that do support simulation and modeling include Emile [5], an environment that allows students to build procedural simulations of basic physics, and Boxer [3] [4], a programming environment designed to allow novice programmers to create programs and simulations. Second, there are the generic simulation tools developed for chemical engineering such as ASCEND, ASPEN+, gPROMS, PRO-II and VEDA [10]. These tools enable complex simulations, both dynamic and steady-state, of chemical processes to be constructed, often using graphical interfaces, and are frequently introduced and used throughout a chemical engineering curriculum. Their wide-ranging and sophisticated functionality is reflected in the complexity of their user interfaces. This complexity makes it difficulty to separate the student's ability to fundamentally understand the problem vs. their ability to comprehend the user interface syntax. These applications are designed to solve complicated general engineering problems and are not study educational hypotheses.

Our experiences suggest that fixed simulations as represented by software such as Maxis' SimCity [14], while easy to use, do not contribute to learning how to model, although they help understand the behavior of a given system. However, full model-building environments, while they can help students learn, require students to expend a great deal of energy learning to program. With DEVICE, we are exploring this tension between ease of use and learning, with the hope of focusing difficulty on segments that are most important that the student learn.

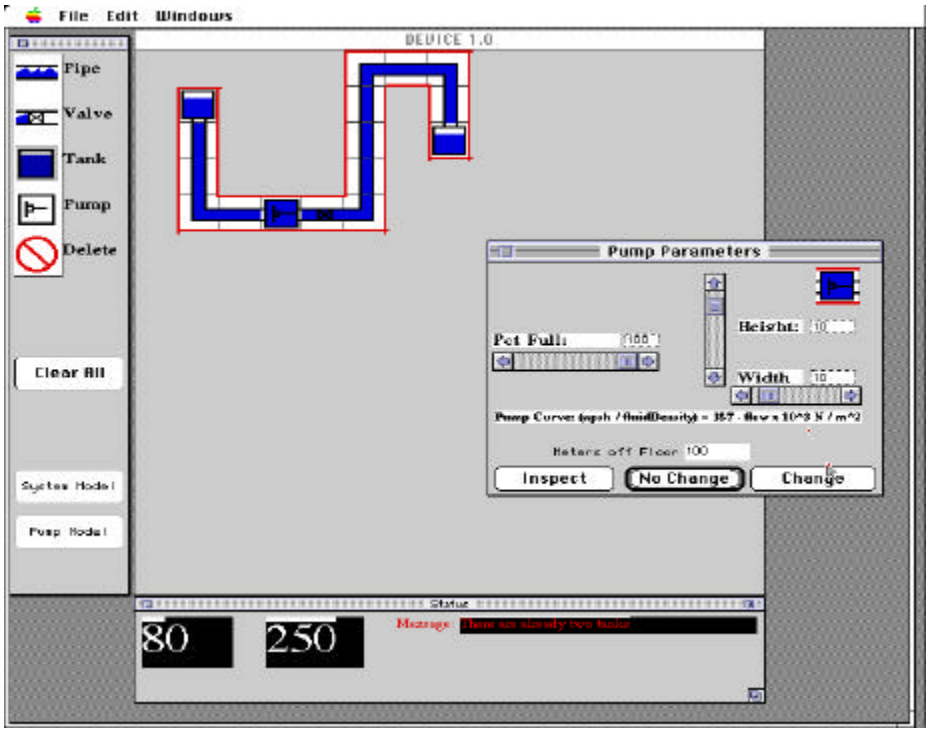

Figure 1: Student Designing Layout in DEVICE 1.0

DEVICE also differs from the other programs by providing support for the important task of testing the model against an alternative model that is a much better approximation of the actual situation. This is important for allowing the students to see mismatches of the model output to a real world behavior.

ORIGINAL DESIGN

The original version of DEVICE allowed students to calculate the flowrate for a given system with a pump already included. This discussion of the interface will focus on where, within the problem, students had the opportunity for interactivity.

What The Students Could Change

In DEVICE 1.0, the student had the most flexibility in creating the physical layout of the system. The student created a physical representation of the system, laying out the tanks, pump and pipe on a tile grid similar to SimCity. Dialog boxes associated with each object allowed the student to adjust the size and other characteristics of that object. Students could also control system level parameters such as the type of fluid being pumped. DEVICE 1.0 was written in SmalltalkAgents®, a commercial implementation of Smalltalk.

Figure 1 illustrates the process of laying out the system.

What Was Visible

This version of DEVICE only allowed students to see the equation model underlying the system, and did not allow them to change that model in any way. In particular, the student could not change which attribute of the layout contributed the value of a particular piece of the equation. They could, however, get a textual description of where the value came from.

The students were provided with an Equation Notebook in DEVICE 1.0. The notebook contained the equation itself, and each term in the equation was labeled, and its value is displayed. The notebook recorded values of the unknown pieces of the equation that the program had already calculated. Having stored the values, the program then uses them to calculate the final flowrate.

What The System Did For the Students
A number of system parameters within DEVICE 1.0 were not visible to the student. Most importantly, the equation that calculated the final flowrate based on the pump in use, while technically visible to the students, was in an awkward location and was not accessed by any of the test students.

FIRST EVALUATION

Method
DEVICE 1.0 was evaluated in a laboratory pilot study with three students who had already taken the undergraduate course that studies pump systems. The students were given a problem which required them to step through the design process of a pump system. Their goal was to achieve a desired flowrate. The task had the students perform the steps needed to calculate the flowrate in DEVICE, and then modify system parameters to change the flowrate. A similar problem had been used in a final exam in a previous quarter, and only four of 21 students were able to complete it successfully. Students were encouraged to think-aloud while solving the problem with DEVICE. Each session was videotaped. The students' problem had high-level directions for using DEVICE. The program designer was at the sessions to function as a help system and field student questions about the system.

Results
The student's ability to solve the problem using DEVICE was encouraging:

- During the pilot study, all three students completed the problem successfully in about twenty minutes each. (Students on the final exam had 30 minutes to solve the problem.)
- All three students remarked how easy the system was to use.
- All three students recognized the mechanical energy balance equation from their class work, so DEVICE was immediately successful in connecting with their theoretical understanding.
- All three students commented that they liked and understood the visualizations used, so DEVICE was also successful in providing understandable visualizations.
- All three students said they would recommend the system to others, and ratings for ease of use and value averaged higher the 4 on a 1 to 5 scale.

The review of the pilot study videotapes focused on issues of usability and of meeting learner's needs. Despite a number of minor problems, the DEVICE interface generally worked well. Every student was able to build a simulation and solve the problem. The tile placing layout was confusing to the students. It was not clear to the students exactly how to begin placing their layout. All three students had difficulty optimizing parameters. The object parameters, accessible by double clicking on tiles, were particularly difficult for students to find. These usability lapses did not keep the students from completing the task in the time allotted, however they did ask questions of the designer frequently.

What Students Learned
There is evidence that students learned some information about modeling pumping systems during the experience of using DEVICE. Anecdotal evidence from the videotapes suggest that they were developing new understandings while working with the system:

- "Oh, it doesn't matter where you put the valve to change [equivalent] length."
- A student who was adjusting the simulation to achieve the desired flow rate said. "Makes sense that [valve equivalent length] is more sensitive [in changing the equation] than that [pipe diameter]." This indicates that he was able to make predictions using the model.

What Students Did Not Learn

We were concerned that the students may not have understood the workings of the simulation very well. For example, one student, when asked after the test if the simulation helped him understanding the system equation, responded, "What equation?". Since this student did recognize the mechanical energy while using the system, it seems as though the student had seen the equation as only an output mechanism, and not as a model abstraction of the system behavior.

We concluded that DEVICE 1.0 did not meet the students needs in two primary ways

- *A fully visible model was not provided.* While we tried to let students inspect the model underlying their simulation, not all of it was understandable and not all of the assumptions were inspectable. Students had questions about the underlying model whose assumptions were not immediately inspectable (e.g., "I'll assume that the density changes when I select a new fluid" and "What changes equivalent length?")

- *The problem solving process was made too easy:* Since the students were not explicitly creating the connections between the equations and the layout, they were not encouraged to think about the relationship of the various parameters. All three students reached a point in their problem-solving process where they entered into a low-level optimizing cycle: Tweak the simulation parameters a little bit, check the equation values, then tweak some more. The equation view ceased to have meaning, and instead became an output mechanism. It wasn't clear that the students were thinking about the parameters and how they related to the equation, or if they were simply exploring the possible parameter space. DEVICE did work that the student needed to do in order to be able to think critically about the system.

DESIGN OF DEVICE 2.0

Our analysis of the pilot test data led us to decide that the 1.0 version of DEVICE would not be a useful learning tool for students. Version 2.0 of DEVICE was written in Prograph CPX.

Our goals in the redesign of DEVICE 2.0 were:

- **Making The System Selectively Harder:** The student participation in the first version was limited to setting up the layout. Essentially, DEVICE 1.0 made the task too easy for the students – interfering with their learning of

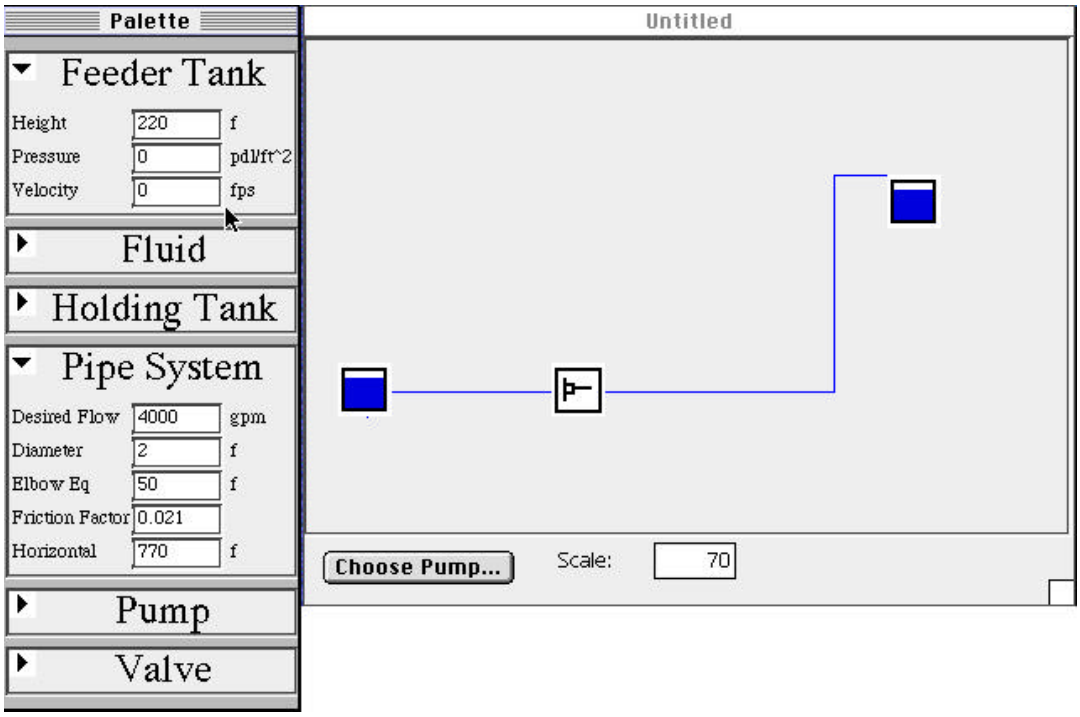

Figure 2: Entering the layout in DEVICE 2.0

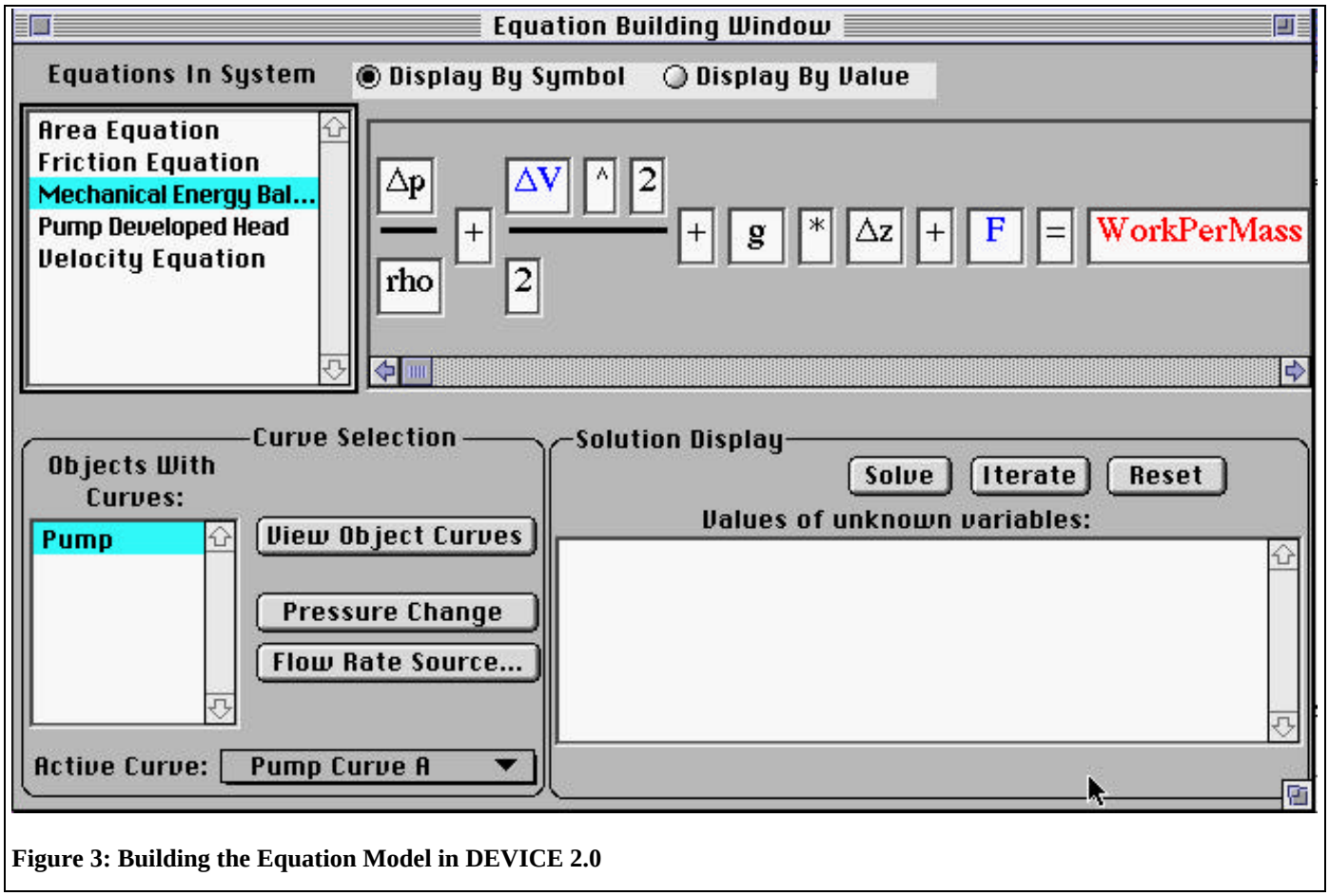

Figure 3: Building the Equation Model in DEVICE 2.0

the material. In DEVICE 2.0 we wanted the focus of the student activity to be on creating and adjusting the model, not the parameters. This moves to the student tasks that had been performed by the computer in DEVICE 1.0.

- **Increased Visibility:** Many of the attributes and assumptions in DEVICE 1.0 are hidden from the students, or are difficult to access. In DEVICE 2.0, we want the students to be able to see every attribute of interest to the system, and to understand the underlying assumptions that are critical to the performance of the system.

- **Tighter connection to real world:** DEVICE 1.0 provides no way for students to verify that their solution is correct or that it would work in the real world. Students using Emile could verify their kinematics simulations by performing analogous actions in the lab [5], but the DEVICE simulations are too complex for that. Instead, DEVICE 2.0 uses an external dynamic modeling environment called gPROMS [2], that will create "real world" values for a pump that the student can compare to the results of his or her static equations

Changes in Visibility

One goal in DEVICE 2.0 was to allow the student better access to the parameters of the system. Figure 2 shows the current DEVICE 2.0 interface for entering physical parameters. The student can enter parameters either on the numerical palette, or by dragging the components in the layout window. The interface allows consistent access to all the applicable parameters of the system. It has the secondary effect of allowing the layout parameters to be set more rapidly than before, without the tile-laying, which gave the students trouble. Typically, most of the default parameters will not be changed in different runs of the problem and can be provided to the student, allowing even more rapid entry.

Changes in Student Activity

The biggest change in DEVICE 2.0 is that the equation model has been made malleable to the students. We hope that this is where the largest part of the students interaction will take place. Ideally, in this step, the student is given

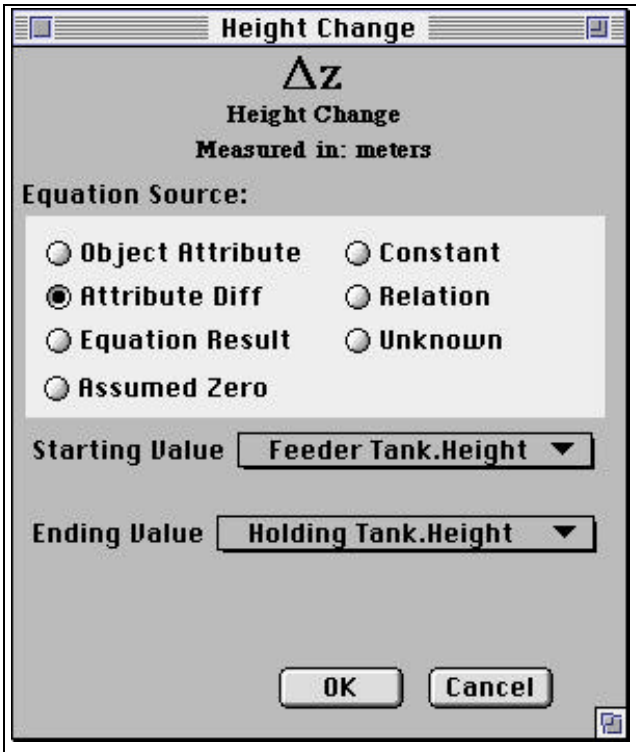

Figure 4: Variable Source Window in DEVICE 2.0

pieces of abstract equations described verbally (such as Change In Pressure), and must combine them to create a system of equations that models the system being designed. Those equations must then be explicitly linked to the parameters of the physical layout that specify the values of the equation – in contrast to the first version, where the linkage was hard-wired into the program.

Figure 3 shows the current interface for building an equation model. This interface allows the students to alter the source of the values for each of the variables in the equation and for which variable the equation can be solved. As a simplifying step, the student is provided entire equations, but still must link the variables to their source attributes. Clicking on a variable in the equation opens a source window, shown in Figure 4. In this window, the student must state the type of attribute desired, a value or a difference in the case of the mechanical balance, and then select the actual attribute from the menu. The system can, at the student's option, prevent the student from placing an attribute of an inappropriate type (a height attribute in a pressure variable, for example). A future version of the program will allow students to create their own equations from component parts.

Real World Connections

After completing the equation model, the student must test the model. This corresponds to the real world task of building the system to the students parameters and testing its performance. Within DEVICE, the student layout is sent to the remote modeler, which returns data about how the real pump would perform. The student can compare the data to the prediction given by their equation model. There are three possible outcomes. First, the model could be completely correct, in which case the student is done. Second, the model could be completely wrong, in which case the student needs to return to the model building step and correct the model. The system needs to provide support for fixing the model in this case. The third case is that the model is slightly off due to factors not considered in the students' model. In that case the student can choose to create a more accurate model, or stick with the current model with an understanding of how to adjust for the factors that could have caused the discrepancy (friction, for example.)

This is potentially a more complicated process than just tweaking parameters. The students will need support for the process, as well as support in understanding interactions of various parts of the model. However, before we started to design these types of scaffolding, we decided to observe students using these features, and work from their needs.

SECOND EVALUATION

During the DEVICE 2.0 development cycle, we did a very brief round of user testing. Findings from that test suggest:

- The goal of focusing student attention on the equation interaction seems to have been met. In the first round, students spent over half of their time preparing the layout. In DEVICE 2.0, well over half the time is spent using the equation.

- As a result of the above, the students seem to be turning their attention to facets of the problem that will be more helpful to them than in the first version – trying to determine where the values of the equation come from, rather than trying to figure out how to make the pipes have a different diameter. The former has the student thinking at the level of the equation model, while the latter is merely an arbitrary parameter tweak.

- The trade-off involved in giving the students more of the task to do is that the program may have become too complicated to expect students to be able to sit down and immediately begin problem solving. Comments from students suggest that a short explanation or sample problem would alleviate this problem.

- It is possible for students to use the layout to illuminate the equation model, however it helps greatly if this is suggested to the student. For example, a student who was unsure of the origin of the system values for starting and ending pressure was able to figure it out after is was suggested that she look at the physical layout to determine this.

FULL EVALUATION

During Fall Quarter 1996, we performed a full scale user test integrated into the curriculum of the course that introduces the mechanical energy balance. This test involved 26 students.

Evaluation Design

The students were introduced to the program during a one-hour lab session that took place during normal class time. In this session, they were to walk through a self-directed tutorial showing how to use DEVICE to solve pump problems. Following the session, they were given two weeks to work on three homework problems on their own time. The three homework problems were more open ended than the tutorial. The three problems also were progressively less similar to the tutorial.

Data Sources

We gathered data from four sources:

- The student performance on the homework itself

- A post-evaluation survey which asked for students subjective response to the program, and asked them to replicate the process they used in DEVICE

- Log files that recorded user interaction with DEVICE

- A comparison of student performance on the aforementioned baseline task with a previous course that had not used DEVICE.

In addition, there was some informal observation of students using the program in the lab setting.

Findings

The following characterize some of our findings:

- Students were able to successfully use DEVICE to solve problems in the domain. Almost all of the students were able to solve the two most similar homework problems.

Although students didn't mind entering the variable sources, their overall subjective enjoyment of the system was rather low (2.4 on a standard 1-5 scale). Much of this was due to some stability and speed problems in the system. In particular we experienced significant problems with our connection to gPROMS. Table 1 gives the time spent at each part of the program. Idle time (more than 5 minutes without a mouse click) was ignored. This shows that about one fifth of the time was spent waiting for gPROMS. The model execution time in gPROMS was a trivial fraction of this. This wait is a double-edged sword: it promotes the student spending time getting the initial model right and doing as few explorations on the real world as possible but it may also act as a source of frustration.

Window	% of time
Layout Palette	27.9%
Equation Window	23.9%
Waiting for gPROMS	22.1%
Variable Source Window	13.7%
gPROMS Data View	9.3%
Layout Window	1.6%
Curve View	1.5%

Table 1

We did not solve the tweaking problem. Students continued to perform tight tweak iterations – solving for answers in far more precision than needed for a real world task. This is again demonstrated by the data in Table 1 where the time spent in the layout palette is indicative of time spent changing parameters. We performed further analysis based on the log file information. We studied the students behaviour in trying to adjust the pipe diameter to achieve a specified flowrate. We calculated the log base 10 of the amount changed in each tweak, a measure of the convergence tolerance to which the students were calculating the answer . This was done for two homework problems associated with the use of DEVICE.

Log10	1	-1	-2	-3	-4
Count	199	361	251	223	67
%	17.0%	30.8%	21.4%	19.0%	5.7%
Log10	**-5**	**-6**	**-7**	**-8**	**-9**
Count	29	16	9	15	1
%	2.5%	1.4%	0.8%	1.3%	0.1%

Table 2

The second analysis was a count in the actual homework of the maximum number of decimal places that students calculated diameter to in the first two homework problems (and documented in their homework).

Decimal places	1	2	3	4
Count	2	16	22	9
%	3%	28%	38%	16%
Decimal places	**5**	**6**	**7**	**8**
Count	5	3	0	1
%	9%	5%	0%	2%

Table 3

- Students did not spend a large enough percentage of their time working on the equation model. This can be seen from Table 1 where it is indicated that the students spent about one fourth of their time in the equation window.

- The students really appreciated the automatic equation solver. Many of the students also specifically mentioned liking the gPROMS solver, but another group was frustrated with this step.

- Student performance on the baseline task improved significantly with respect to the control group. Average score on the baseline task improved from 3.7 to 5.2, significant with p=0.026

CONCLUSIONS

The requirements of interfaces designed to support learning are different than for interfaces designed to support performance. There is a necessary trade off between ease of use and facilitating learning by creating opportunities for the student to think. Specifically, there is a difference between an interface which merely allows a student to observe a simulation, and one which allows the student to actively specify and run a simulation model.

In testing DEVICE 1.0, we found that a simulation–only interface does not facilitate learning. The ease of use that makes the program a useful support tool can work against it as a learning tool, by negating the learner's need to think about what they are doing. Our needs analysis shows that the chemical engineering students with whom we are working need to improve their modeling skills. To do that, we need to provide for them an environment that encourages them to use those skills without overwhelming them with new types of models.

Early results on DEVICE 2.0 suggest that it may be successful in focusing student attention on areas of the problem that students typically have the most difficulty with. This version of DEVICE has made more progress at fulfilling our first objective (teaching modelin skills). However, scaffolding needs to be built into the environment in the form of help that can be faded as the student gains expertise with both the software environment and the concepts. The gPROMs solver provides the capability of achieving objective 3 (comparing models with real systems), although it is not clear if students have developed an appreciation for these differences. The third objective (understanding degrees of freedom) was not explicitly addressed by our previous evaluations. Our next planned classroom evaluation will require students to answer questions about a pump design problem that will explicitly address whether the appreciate the origin of the differences between their model and the real system (approximated by a more realistic gPROMS simulation). Students will also be asked questions to determine if they better understand how degrees of freedom change due to constraints such as characteristic pump curves.

ACKNOWLEDGMENTS

DEVICE is funded by NSF grant #RED-9550458 and by the EduTech Institute.

REFERENCES

1. Abelson, H., Computation as a Framework for Engineering Education, .
2. Barton, P.I. and C.C. Pantelides, The Modelling of Combined Discrete/Continuous Processes. AIChE Journal, 1994. **40**: p. 966-979.
3. diSessa, A.A. and H. Abelson, Boxer: A reconstructible computational medium. Communications of the ACM, 1986. **29**(9): p. 859-868.
4. diSessa, A.A., H. Abelson, and D. Ploger, An overview of Boxer. The Journal of Mathematical Behavior, 1991. **10**(1): p. 3-15.
5. Guzdial, M., Software-realized scaffolding to facilitate programming for science learning. Interactive Learning Environments, 1995. **4**(1): p. 1-44.
6. Guzdial, M., et al. Simulated Environments for Learning Real World Contexts in Chemical Engineering. in International Conference on the Learning Sciences. 1996. Evanston, IL.
7. Jackson, S., et al. The ScienceWare Modeler: A Case Study of Learner-Centered Software Design. in CHI. 1995.
8. Jackson, S.L., et al. Model-It: A case study of learner-centered software for supporting model building. in Proceedings of the Working Conference on Technology Applications in the Science Classroom. 1995. Columbus, OH: The National Center for Science Teaching and Learning.
9. Mandinach, E., Model-building and the use of computer simulations of dynamic systems. Journal of Educational Computing Research, 1989. **5**(2): p. 221-243.
10. Marquardt W., Trends in Computer-Aided Process Modeling, Proceedings of PSE'94, Kyongju, Korea, 1994.

11. Rappin, N., et al. Teaching Chemical Engineering Modeling through Simulated Environments. in American Educational Research Association. 1997. Chicago, IL: (submitted).

12. Tinker, R.F., Teaching theory building, . 1990, Technical Education Research Centers, Cambridge, MA.

13. Tinker, R.F. and S. Papert, Tools for science education, in 1988 AETS Yearbook: Information Technology and Science Education, J.D. Ellis, Editor. 1988, Association for the Education of Teachers in Science. p. 5-27.

14. Wright, W., SimCity, 1989, Maxis.

Biographies

Noel Rappin is a PhD student at the Graphics, Visualization and Usability Center of the Georgia Institute of Technology, where he received his Masters degree. He received an undergraduate degree in Computer Science and History from Brandeis University. Noel's current work deals with teaching modeling skills to students in various domains.

Mark Guzdial is an assistant professor in the College of Computing at Georgia Institute of Technology, associated both with the GVU Center and the EduTech Institute. He completed his joint doctorate in Education and Computer Science at the University of Michigan in Fall, 1993. His research focus is on technological support for students' learning in project design and construction. He emphasizes use of collaborative learning environments, multimedia composition, and modeling and simulation.

Pete Ludovice is an assistant professor in the Chemical Engineering Department at Georgia Institute of Technology. He received his doctorate from the Massachusetts Institute of Technology and his research interests emphasize the use of computer simulation of macromolecular systems. Dr. Ludovice’s educational research interests include the use computer and internet based tools to provide scaffolded learning in chemical engineering.

Matthew Realff is an assistant professor of Chemical Engineering at the Georgia Institute of Technology. He received his doctorate from the Massachusetts Institute of Technology, and his broad research interests include process design, simulation, scheduling and control. His interests in education research focus on enabling students to acquire modeling skills through self-directed learning and on tools to promote collaboration in design and modeling.