Simulated Environments for Learning Real World Contexts in Chemical Engineering

Mark Guzdial, Noel Rappin
Georgia Institute of Technology
College of Computing
guzdial@cc.gatech.edu, noel@cc.gatech.edu

Matthew Realff ,Pete Ludovice
Georgia Institute of Technology
School of Chemical Engineering
matthew_realff@chemeng.chemse.gatech.edu , pete_ludovice@chemeng.chemse.gatech.edu

Abstract: The goal of DEVICE (Dynamic Environment for Visualization in Chemical Engineering) is to facilitate student learning through construction of models and evaluation of the models executing as simulations. In DEVICE, students solve real world problems in a simulated environment with visualizations that help establish the realistic context. The model underlying the simulation is visible, manipulable, and represented in a form that easily connects with the students' theoretical understanding. Students construct their own models to explain the behavior of the system. With DEVICE, we hope to find a middle ground between merely viewing a simulation and building one from scratch where the students are able to actually complete and learn through modeling tasks. The first version of DEVICE allowed students to view, but not manipulate, the model. User tests showed that the models need to be malleable at the modeling level, not just the simulation level.

Goals of DEVICE

Since World War II, engineering schools have emphasized theory over practice. [Augustine & Vest 1994] Although this emphasis has been successful in teaching theory, many engineers feel that students need more applied problem-solving skills. In particular, students often learn the principles of engineering science without learning current practice in applying the principles to real analysis situations. Most engineering students are not taught the principles of engineering design, especially as it is applied in practice. There have been many calls to reform the current methods of teaching engineers towards offering students the opportunity to actually participate in a realistic activity [Dixon 1991] . However, such activities are often too complex or expensive to be easily performed by novices. Computer simulation can be a way to give students such experience more easily.

The goal of DEVICE (Dynamic Environment for Visualization in Chemical Engineering) is to facilitate student learning through construction of equation-based models and evaluation of those models executing as simulations. Modeling – creating a conceptual representation of reality – is a major part of engineering practice that is rarely explicitly taught [Abelson 1991]. Students in Chemical Engineering learn a wide variety of theoretical issues (e.g., equations, laws of chemistry and physics) but may not learn how these relate to real-world tasks. This suggests that in addition to learning modeling, students also need to learn skills in a way that enables transfer.. In DEVICE, students solve real world problems in a simulated environment with visualizations that help establish the realistic context. The model underlying the simulation is visible, manipulable, and represented in a form that easily connects with the students' theoretical understanding. Students construct their own models to explain the behavior of the system. This will allow them to better transfer knowledge to the real-world situations they will face as practicing engineers.

Our research extends the work of Guzdial in Emile [Guzdial 1995], a model-building environment in which students learned physics by constructing physics simulations. One of the findings in the Emile studies was that students spent more time exploring and validating their simulation than building the actual model. We can improve on Emile's support for learning by providing visualizations and tools to support this evaluation of the simulation. In particular, DEVICE focuses on the question of evaluating a model by comparing its predictions to the real world results.

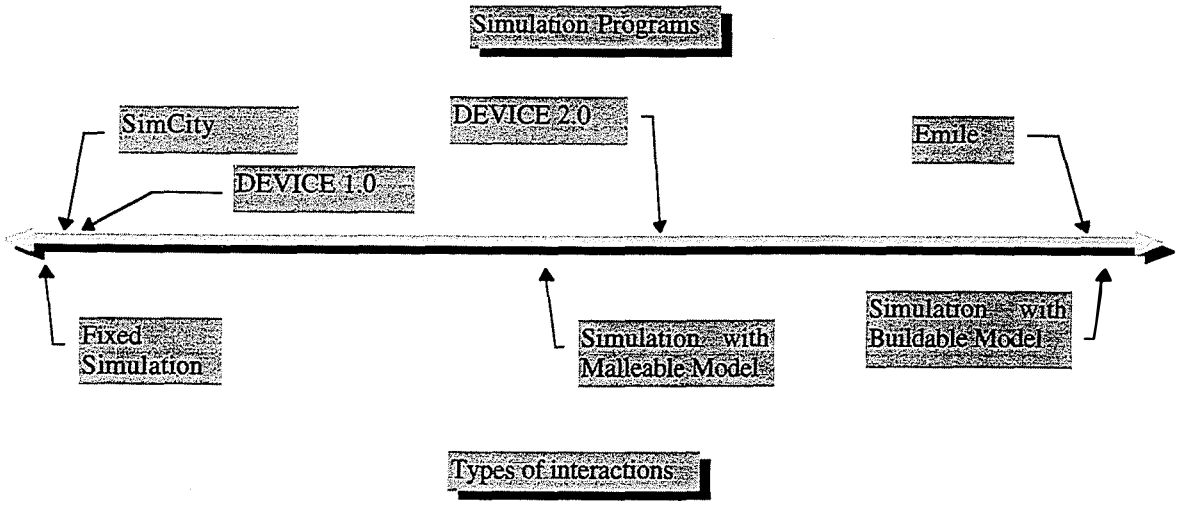

Figure 1: Continuum of Simulation interaction

Student interaction with a model-based simulation can be thought of as occurring somewhere on a continuum.

- At one end is only viewing a simulation, or possibly being able to tweak surface simulation parameters, the level of interaction of someone using SimCity. Actually being able to view the model that drives the simulation is a deeper interaction, available in SimCity only by buying a strategy guide that contains the formulas.
- The next step in model interaction is malleable models, which would be possible if SimCity actually allowed you to change the formulas that actually drive the simulation.
- Finally, the model can actually be built by the students, from scratch or from prefabricated pieces, which is the way that Emile works.

We believe that exploring models is an effective way to learn a domain, and that merely viewing and exploring the simulation is not enough to facilitate learning. However, building a complex model is an extremely difficult task for novices. With DEVICE, we hope to find a middle ground where the students are able to actually complete and learn through modeling tasks. The first version of DEVICE allowed students to view, but not manipulate, the model. User tests showed that the models need to be malleable at the modeling level, not just the simulation level.

DEVICE Prototype

The first version of DEVICE only allowed students to see the underlying system model, and did not allow them to change that model. DEVICE 1.0 explored mechanical energy balance equations associated with pumping systems. Students laid out a pumping system in DEVICE 1.0 using an understandable visualization of pumps, valves, pipes, and tanks. They could manipulate the simulation by double-clicking on any component and changing the various parameters through scroll-bars and other interface components. To analyze the simulation the students have built, they could call up the mechanical energy balance equations, which describe how the physical settings and structure of their simulation maps to the energy balance equation. The values of the equation changed as the students adjusted simulation settings. For example, adding a valve to the system increased friction in the mechanical balance equation, which also increased the flow rate in the system. [Fig. 2] shows the interface to the system layout construction, and how the student was presented the underlying model [Rappin, Guzdial, et al 1995].

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 which changed 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 videotape. The findings included:

- 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 classwork, 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
- However, we were concerned that the students may not have understood what happened in 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 started to see the equation as only an output mechanism, and not as an abstraction.

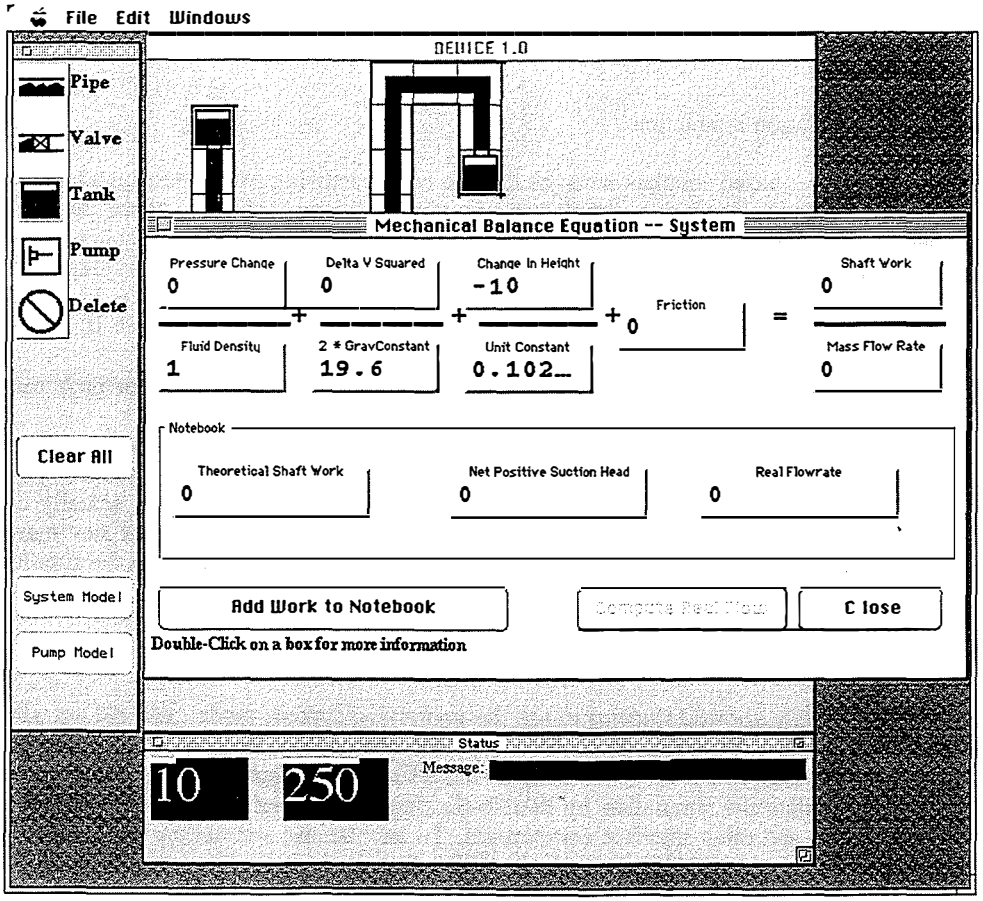

Figure 2: Low-level (equation-based) specification of components and their relationships

Analysis of pilot study

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 usability lapses did not keep the students from completing the task. The students liked the system and seemed to understand the majority of it. All of them said that they would recommend the system to a student learning about pump systems.

The learning issues are more complicated. There is evidence that students learned during the experience of using DEVICE. Quotes 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.

However, DEVICE 1.0 did not meet learner's needs in three key areas:

- *Supporting alternative solution paths.* When one of the students first read the problem, he asked for a calculator – he wanted to try to solve the problem via equations first. Later, he made comments suggesting that he wanted to take measurements of his system and finish the problem via calculation. While we want to encourage students to use the simulation to solve the problem, we should also allow students to try to solve the problem via their own methods – if possible.
- *Making the underlying assumptions visible.* 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 made comments at various points about terms that they were unsure of or re-discovered during the problem (e.g., "Now I remember equivalent length"). Students had questions about the underlying model whose answers were not immediately inspectable (e.g., "I'll assume that the density changes when I select a new fluid" and "What changes equivalent length?")
- *Encouraging thoughtful exploration of the system:* All three students reached a point in their problem-solving process where they entered into an optimizing cycle: Tweak the simulation parameters a little bit, check the equation values, then tweak some more. 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. 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.

Re-design of DEVICE 2.0

The design of DEVICE 2.0 is based on our analysis of the pilot test data. The usability problems are relatively minor to repair. The problem with supporting alternative solution paths is also relatively easy to fix – we can provide a calculator and a number of measurement devices within the simulation to support student inspection of their simulation, something that had always been part of the plans for DEVICE. The latter two learner's needs are more complicated.

One of the issues in making more of the underlying assumptions visible is the representation for these assumptions. Not everything in a computer-based simulation can be easily expressed as a static equation. We have decided to use gPROMS as our underlying model representation language. gPROMS is an equation oriented language enhanced with statements that enable the user to declare how the model structure changes with events, either planned or unplanned [Barton & Pantelides, 1994] . Because is it equation-based, it is isomorphic to the equations with which the students are already familiar. gPROMS is growing in popularity in the Chemical Engineering community, so learning gPROMS is a useful skill for the students that may transfer into later modeling and simulation activity.

Sample gPROMS program:

```
TASK Simple_Start_Up

SCHEDULE
   SEQUENCE
      # Start feed pump to tank - outlet valve closed
      RESET
         Feed_Pump.status := Feed_Pump.on ;
      END

      # Wait until the tank reaches its level set point
      CONTINUE UNTIL Tank.Liquid_Level > Level_Control.Set_Point

      # Concurrently...
      PARALLEL
         # Start product pump
         RESET
            Product_Pump.Status := Product_Pump.On ;
         END

         # Close the level control loop
         REPLACE
            Level_Control.Control_Action
         WITH
            Level_Control.Bias := 12.7 ;
         END # replace

         # Eliminate any reset windup accumulated while control loop
   # was open
         REINITIAL
            Level_Control.Integral_Error
         WITH
            Level_Control.Integral_Error = 0 ;
         END # reinitial
      END # parallel

      # Continue simulation for another 100 time units
      CONTINUE FOR 100
   END # sequence
END # Simple_Start_Up
```

In DEVICE 2.0, the student will able to inspect the underlying gPROMS code, but will more often deal with the simulation using equation based constructs that correspond to pieces of the gPROMS simulation. The underlying assumptions that drove the simulation in DEVICE 1.0 will all be explicitly placed in a form which the students can link to pieces of the physical layout of the system in DEVICE 2.0. For example, a student would have to explicitly tell the system where the start and ending points are for calculating pressure changes, rather than having that information hard-wired into the system.

The last learning issue is the most complicated: Encouraging students to think about the simulation parameters and the role of these parameters in the overall model. In some sense, ease-of-use is an enemy here. We need to encourage students to reflect on their actions, and not simply perform them as effortlessly as possible. By explicitly linking system parameters to pieces of the model, hopefully reflective thought can be encouraged in the students as they create their models.

DEVICE 2.0 will guide the student through a four step process in solving modeling problems:

1. Defining the Physical Layout
2. Building the Equation Model
3. Committing to a Model
4. Evaluating the Model

In the first step, the student defines the physical layout by entering in the known parameters of the system to be designed. For the pumping problems currently being used, those parameters include the height of the tanks, the fluid being pumped and whether there is a valve in the system. [Fig. 3] 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.

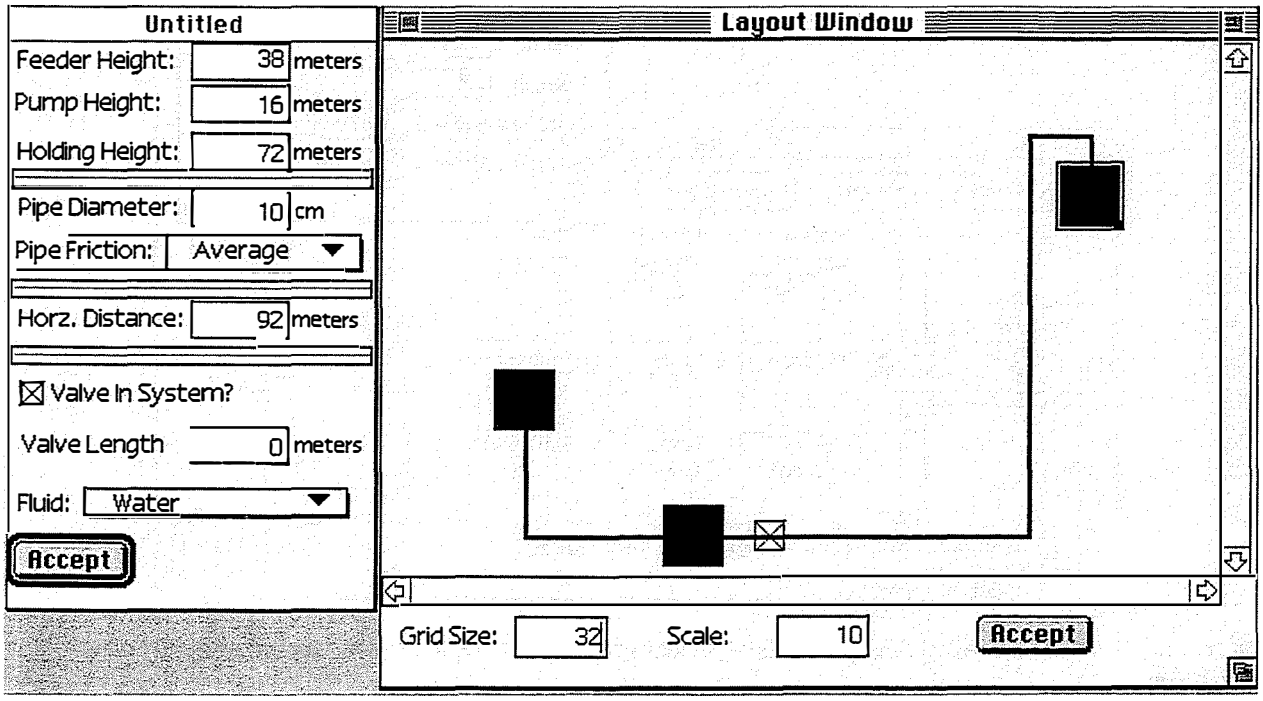

Figure 3: Entering the layout in DEVICE 2.0

The second step, building the equation model, is where the largest part of the students interaction will take place. In this step, the student is given pieces of abstract equations, and must combine them to create a system of equations that models the system being designed. Those equations must be explicitly linked to the parameters of the physical layout that specify the values of the equation.

[Fig. 4] is a mock-up of what the equation-linking interface might look like:

- The student here is linking the Pressure Change box in the Mechanical Energy Balance (MEB) equation. The system knows that the Pressure Change must be a delta term (difference between two values), so it can provide menus of possible values for the initial and final pressure.
- The outlined form of the term in the Shaft Work box indicates that this is the output term of this equation, for use as input to some other equation.
- The top buttons demonstrate possible kinds of process support: *What Next?* might suggest the next step in the process; *Critique* might point out problems that the system can identify; and *I Have Done...* might present the process (perhaps in a checklist form) with an indication of what's completed so far.
- The bottom row of buttons is used to create new equations, link equations, test various parameters in the equations, and apply the equations to the simulation.
- The upper list box (on the left) shows the equations in the DEVICE library (such as MEB).

- The lower list box shows equation components which can be dragged in when defining an equation.

Thus, we have re-designed DEVICE 2.0 such that internal parameters are set by defining an equation for the parameter, in terms of givens, standard equations, and terms that can be computed by the student. When a student wants to change a parameter in DEVICE 2.0, they will construct an equation that defines what value the parameter should have in terms of the rest of the system model. The system model will be available in the same equational form as the equations that the student writes, so it will be possible for students to reuse (by "linking" in equations) equations written earlier or equations already in the system when specifying a parameter.

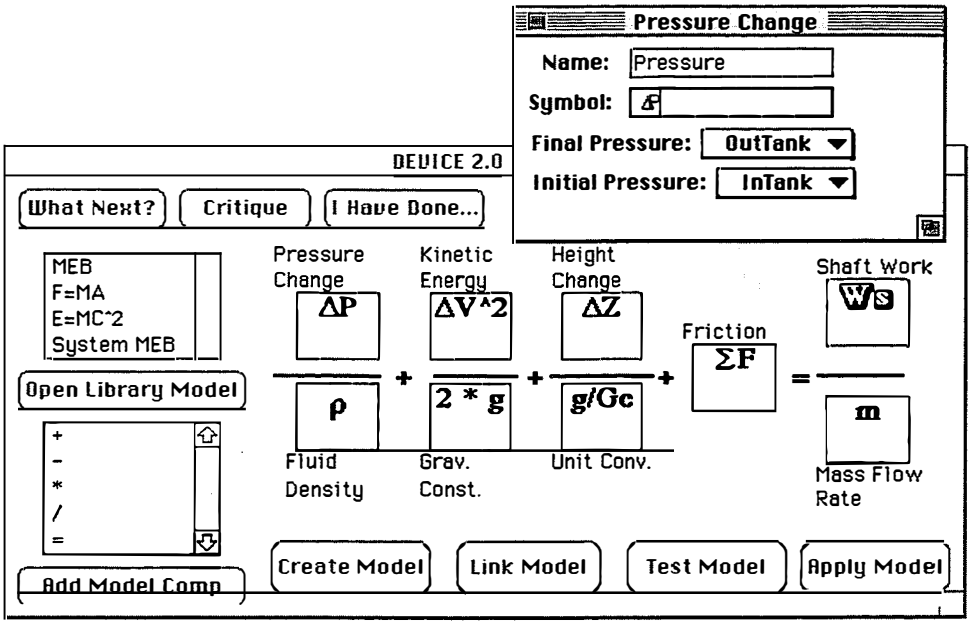

Figure 4: DEVICE 2.0 Mock-Up Screenshot: Demonstrating how students will use models to control the simulation.

For example, the primary method of controlling flow rate in DEVICE 1.0 was by manipulating the amount that a valve as open (actually, manipulating the amount of frictional drag that the closing of the valve added to the fluid). In DEVICE 1.0, the students input a value for the valve, and checked the equations to see if they had reached their target. If not, they iterated, by trial and error. In DEVICE 2.0, the student will set up an equation for the valve based on the fluid density, and the pump type, and other parameters, testing then to see if they can control the valve length to produce the desired flow, or to understand how flow rate and valve length interact.

This is 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. Critics and guides will be implemented to support the students in their tasks. We also plan to implement a qualitative reasoner to point out how a change in one part of the system will effect other parameters. Representations will need to be provided to give a meta-level view of how equations relate in defining an entire system model. DEVICE will have knowledge of what the types of values should go into an equation so that it can warn the student if, for instance, the student puts a height measure in a slot that expects a pressure measure.

After the student has build a model, they must commit to the solution they have created. This mimics the real-life step where the engineer doing the design would buy equipment and build based on the design. For our purposes, this is intended as a further control to keep the student from a tinkering loop that interfered with reflection on the model in the prototype. At the point of commitment, the student has created a physical design, as well as a model that makes a prediction of what the performance of the design will be.

In the final step, that model is evaluated. In the real world, the pump would be constructed, and meters would generate data to tell if the system performed as intended. In DEVICE, we will rely on a very realistic "real world" simulation that will generate data, based on the "purchased" equipment, for the student to compare to the prediction of their own 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.)

These changes to the DEVICE environment will allow students a much deeper interaction with the underlying models that we hope they will learn. In addition, the students will learn about the process of modeling and how it fits into the real-world tasks of a practicing Chemical Engineer.

Since DEVICE is based on equational models, it is potentially applicable to a wide range of domains, especially within engineering. To facilitate the creation of DEVICE content for other domains, a separate authoring program called IDEA (Instructor's DEVICE Editing Assistant) is being created. IDEA will allow instructors to enter equational models, along with information that DEVICE will use to guide its scaffolding of the students. With IDEA, we hope that DEVICE becomes a general tool for educational modeling activities in Chemical Engineering.

DEVICE 2.0 is currently under development, and is scheduled to be used in a Chemical Engineering classroom during Fall Quarter, 1996, with further development planned based on the results. The content for this test will be generated using IDEA, however by the developer and not an instructor.

References

[Abelson 1991] Abelson, Harold, 1991. "Computation as a Framework for Engineering Education" in "Research Directions in Computer Science: An MIT Perspective." ed. Albert R. Meyer, John V. Guttag, Ronald L. Rivest, Peter Szolovits. MIT Press: Cambridge, Mass.

[Augustine & Vest 1994] Augustine N., Vest C.M., eds. (1994)*Engineering Education for a Changing World.* American Society for Engineering Education:Joint Project Report by the Engineering Deans Council and Corporate Roundtable by the American Society for Engineering Education

[Barton & Pantelides 1994] Barton P.I., Pantelides C.C. (1994) "The Modelling of Combined Discrete/Continuous Processes." *AI & Chemical Engineering Journal*; 40:966-979.

[Dixon 1991] Dixon J.R. (1991) "The State of Education, Parts I & II." *Mechanical Engineering*; (February and March):64-67 (February), 56-62 March.

[Guzdial 1995] Guzdial M. (1995) "Software-realized scaffolding to facilitate programming for science learning." *Interactive Learning Environments*; 4(1):1-44.

[Rappin, Guzdial, et. al 1995] Rappin N., Guzdial M., Ludovice P., Realff M. (1995)*DEVICE: Dynamic Environment for Visualizations in Chemical Engineering.* AI-Education Conference. Washington DC: : Poster Session.

Acknowledgments

This research was funded by a grant from the National Science Foundation #RED-9550458, and by the EduTech institute at Georgia Tech through a grant from the Woodruff Foundation.