An Application Framework for Creating Simulation-Based Learning Environments

Anne Rose, David Eckard**, and Gary W. Rubloff+*
Human-Computer Interaction Laboratory*
Institute for Systems Research+
University of Maryland, College Park, MD 20742
North Carolina State University**
Email: rose@cs.umd.edu, david_eckard@usa.net, rubloff@isr.umd.edu

ABSTRACT
While there are numerous types of electronic learning environments including collaboratories, construction toolkits, systems with "scaffolding" and simulations, it is difficult to find authoring tools to build these systems. We have developed an application framework for constructing simulation-based learning environments called SimPLE (Simulated Processes in a Learning Environment). Environments developed with SimPLE use dynamic simulations and visualizations to represent realistic time-dependent behavior and are coupled with guidance material and other software aids that facilitate learning. The software architecture enables independent contributions from developers representing educational content (e.g., simulation models, guidance materials) and software development (e.g., user interface). We provide a user interface template and accompanying software aids to reduce the software development effort.

KEYWORDS: Simulation, education, learning environments, authoring tool, software engineering

INTRODUCTION

Traditionally, the focus of educational curricula has been on content and its delivery (i.e., subjects are broken down into smaller, more manageable subtopics and taught in the classroom). Today there is a strong movement toward "learner-centered education" which focuses on the needs of the learner [10]. While the success of "learner-centered education" does not rely on technology, electronic learning environments serve as a powerful catalyst for change. These environments include:

- collaboratories that facilitate group communication [3][12],
- construction toolkits that teach design and modeling skills [4][7],
- systems with "scaffolding" that allow learners to start simple and build complexity [11][13][14], and
- simulations that support "learning by doing" [2][6][8].

Simulation environments are powerful learning tools that encourage exploration by allowing learners to manipulate parameters and visualize results. In academic settings, they are used to enhance lectures, supplement labs, and engage students. In the workplace, they are cost-effective training mechanisms. There are two basic groups of simulators: inanimate (off-line) and live (on-line, real-time) [9]. Inanimate simulators are used to evaluate complex equations and models. They do not simulate real-time operations of physical systems so user interaction is limited. However, live simulators are highly interactive. They closely resemble the physical system while allowing learners to explore situations not possible with the actual system. We have developed an application framework for constructing live simulation-based learning environments called SimPLE (Simulated Processes in a Learning Environment).

Most commercial simulation packages (e.g., Excel, Matlab, and VisSim™) are built to handle a variety of situations so they provide generic user interfaces that require modest training. While these packages are very useful for creating simulations for a variety of domains, the emphasis of an effective learning environment must be on the concept being taught, not on learning how to use another tool. SimPLE pairs the power and flexibility of a generic simulation package with the advantage of a custom front end.

Learning environments developed with SimPLE use dynamic simulations and visualizations to represent realistic time-dependent behavior and are coupled with guidance material and other software aids that facilitate learning. The software architecture allows independent contributions by developers representing educational

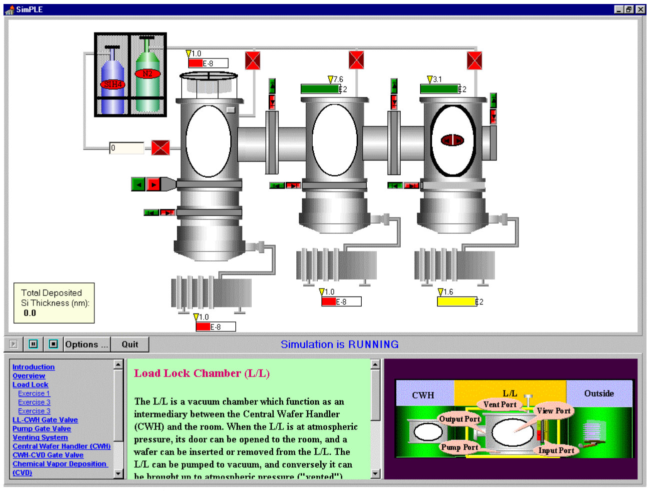

Figure 1. ClusterSim for teaching advanced cluster tool processing for semiconductor manufacturing

content and software development. This allows the educational content (e.g., simulation models and guidance materials) to be developed by the educators who have the domain knowledge. Using the user interface template provided and several software aids, SimPLE allows custom front ends to be created quickly and with minimal coding.

In our initial efforts, three learning environments have been created: ClusterSim and VacTechSim for learning about semiconductor manufacturing, and NileSim for learning about the hydrology of the Nile (all are available for download from http://www.isr.umd.edu/SimPLE/).

Semiconductor Fabrication Examples

The semiconductor manufacturing and associated equipment industries face a critical shortage of skilled workers and a staggering challenge in the education and training of new personnel. Using SimPLE, we have developed two modules with associated guidance material: ClusterSim, for teaching advanced cluster tool processing (Figure 1) and VacTechSim, for learning about vacuum and gas flow technology (Figure 2). Potential users include graduate students, operators, and practicing engineers.

Learners can open valves, start pumps, and change temperatures and observe the effects on parameters, like silicon deposition and chamber pressure. They can also read guidance material, perform selected exercises, highlight simulator objects, and send mail to educators. Advanced learners can view the detailed simulation logic running in the background. Both systems are currently being evaluated in academic and industrial settings.

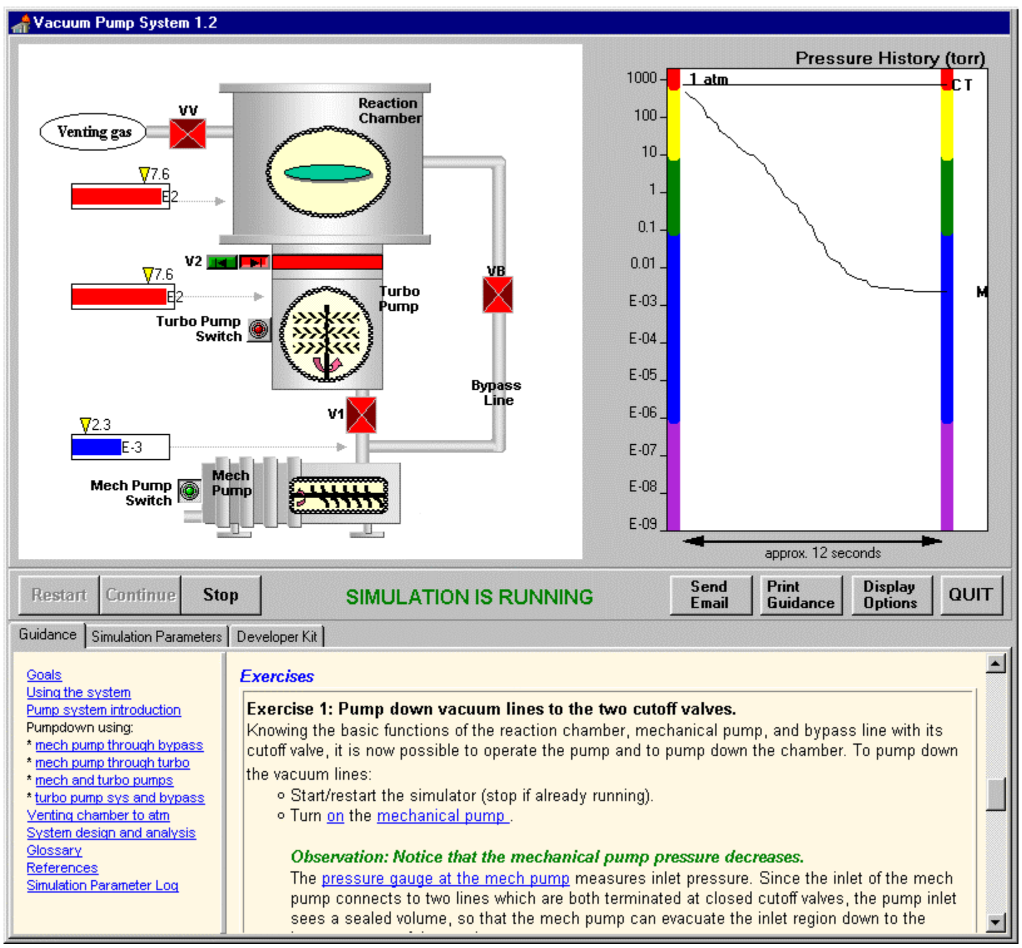

Figure 2. VacTechSim for learning the basic principles of vacuum pump technology as needed for semiconductor manufacturing

Nile Example

Based on 200 years of river records, NileSim was developed to help explain complex river behavior and management (Figure 3). NileSim is used in a large, multidisciplinary freshman course on the 5000 years of hydraulic civilization in the Nile Valley of Egypt. This course comprises both technical and non-technical majors, and faculty from engineering, biology, and government and politics. Teams of students use NileSim to study how different schemes for managing the scarce water resources of the basin affect the natural environment of the Nile and the economics of riparian countries. The reaction to NileSim has been so positive that it has been proposed as the central focus for the class in the future.

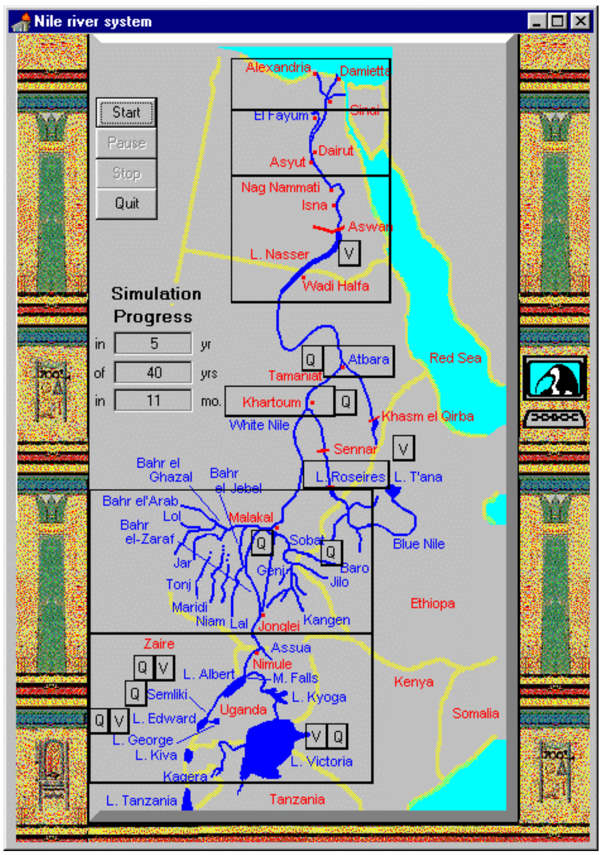

Figure 3. NileSim for learning about the hydrology of the Nile river

APPLICATION FRAMEWORK

Application frameworks must provide appropriate modularization of function and separation of concerns, just as database management systems and user interface tools promote dialog independence [1][5]. In creating SimPLE, we felt it was very important to allow independent contributions by user interface designers and educators. The architecture of SimPLE uses the common strategy of separating the graphical user interface from the application, in our case a simulation engine, by using a separate dialog component to handle communication between the two (Figure 4). The guidance material, another part of the educational content, is also separated from the user interface. In the current implementation, the user interface is in Delphi, the dialog component is a dynamic linked library (DLL), the simulation engine is VisSim^{TM}, and the guidance material is in HTML.

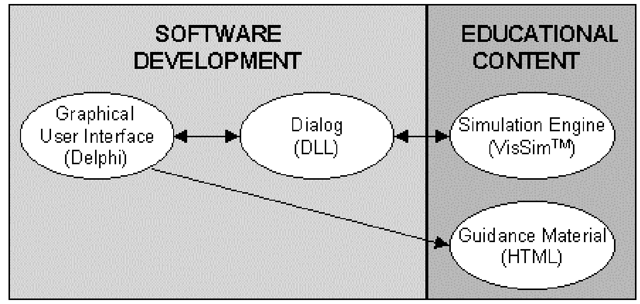

Figure 4. SimPLE software architecture

Using SimPLE, educators can quickly create simulation-based learning environments for a variety of educational domains. The two main steps involve creating the educational content and designing the user interface. Educators start by creating the simulation engine and the guidance material using existing tools. Good guidance material includes a system overview, exercises to perform, a glossary of terms, and additional references. The user interface for the educational content is developed using a template and several "plug-in" software aids that reduce the development effort by providing a rich array of learning tools.

Simulation Engine

Educators start by creating a simulation model and specifying the inputs and outputs that will be available to the learner. We chose VisSim^{TM} to create the simulations because it is a commercial product commonly used in engineering education. However, substituting a different simulator or a different type of software package would only require modifications to the dialog component since the user interface is separated from the application. The only requirement is that the application used must be able to accept inputs and return outputs.

Figure 5 shows the VisSim^{TM} model for VacTechSim. The dark gray boxes are compound blocks that contain multiple layers of detailed logic. For simplicity, only the top layer is shown here. The only requirement imposed by SimPLE is that the model must specify the input and output variables available to the learner. In our example, these variables are specified in the VSin and VSout sections.

Because the simulation engine is not tightly coupled with the user interface, educators have found it useful to create their models so they can be controlled both via the user interface ("remote") and by direct manipulation of the simulator ("local"). This allows the content author to operate and modify the simulator in a completely standalone mode. This is especially useful during the

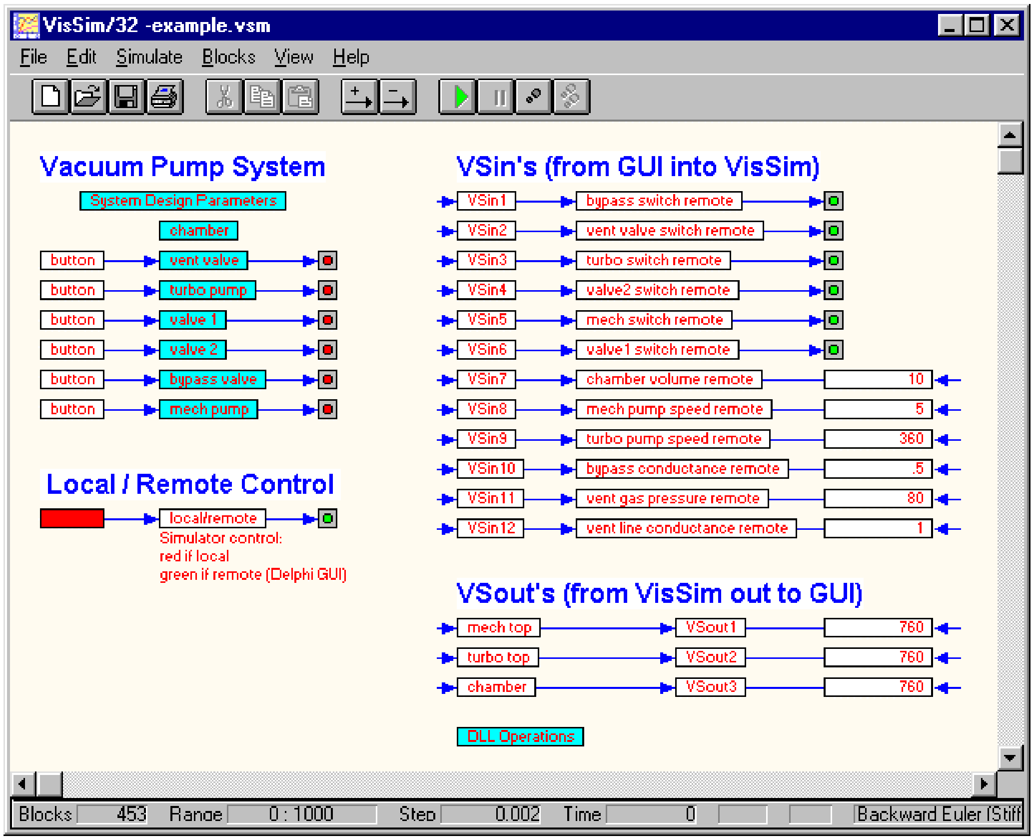

Figure 5. VisSim model overview for VacTechSim

development and debugging stage, since a change in the simulation logic should not change the user interface (unless inputs or outputs are added/removed).

Dialog

The key to linking the user interface and the educational content is the dialog component. Currently implemented as a C++ DLL that uses continuous polling, the dialog component handles all communication regarding user inputs and simulator outputs between the user interface and the simulation engine. For example, when a user clicks on a valve in VacTechSim, the dialog component notifies the simulator, which opens the valve in its internal model. It then passes the simulator outputs (which might show a change in pressure) back to the user interface for display to the user. In this way, the learner "runs" the physical system depicted as a dynamic model on the user interface, with response and feedback to their actions determined by the underlying, physically realistic dynamic simulator.

Adding error messages and warnings increases the educational benefit to the learner but also makes the communication more complex. Rather than simply performing a user action, the user interface must "request" (via the dialog component) that an action be performed. The user interface then waits for the simulator response. If the action is approved, the action is performed. Otherwise, the user interface conveys a useful message to the learner indicating why the action could not be performed.

User Interface

Without adequate authoring tools, developing custom front ends can be very time consuming and difficult. When surveyed, most educators site lack of time as the reason for not using new technology in their classrooms. Because of limited time and software expertise, authoring tools that minimize creation time but still allow customization are critical to the wide spread success of learning environments.

Using a template and several "plug-in" software aids, custom user interfaces can be developed very quickly in SimPLE. The user interface template is a Delphi project that consists of three main areas: an empty simulation panel, a control bar, and a tabbed notebook (Figure 6). Customizing the simulator panel involves three steps: making a background image (which can be drawn with any image editor), positioning the user interface controls, and mapping the user interface controls to the simulation variables. General functionality, like starting and stopping the simulation and printing the guidance material, is provided in the control bar. The guidance material will be displayed automatically in the notebook area along with any simulation design parameters. The notebook area also

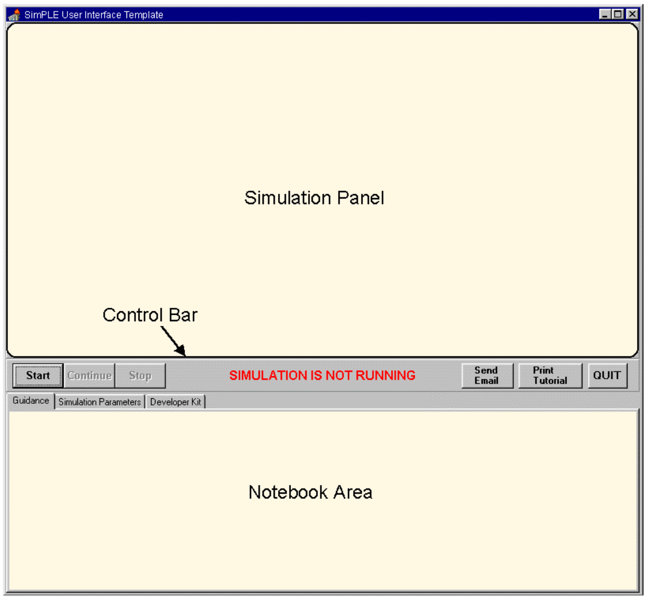

Figure 6. User interface template provided by SimPLE for creating custom front ends for simulation-based learning environments

contains a developer kit that allows educators to further customize the system.

To help developers create rich learning environments, the user interface template is coupled with several software aids, implemented as Delphi components. These components include:

- TDLLConnect, for mapping the user interface controls to the simulation variables,
- TDesignParameters, for creating simulation design parameter controls on-the-fly,
- TActionLinks, for enhancing the guidance material to support actions, like highlighting simulator panel objects, and
- TCommunication, for allowing students and instructors to communicate via email and attach simulation files.

A library of flicker-free user interface controls (including valves, switches, and pressure) has also been created to support the semiconductor manufacturing domain.

TDLLConnect. TDLLConnect makes it simple to map user interface controls to simulation variables. Specifically, TDLLConnect generates two events that are raised when input data is requested and when output data is received. Developers simply create callback procedures that map the user interface controls to the simulation variables. Figure 7 shows an example procedure that maps user interface controls to simulator inputs. To make this simpler, SimPLE might allow developers to click on a control and select the value to send or update.

```
procedure TMainForm.BeforeVisSimCall(Sender: TObject);
begin
  with DLLConnect1 do begin
    DatatoVisSim[1] := integer(VB.Value = Opened)
    DatatoVisSim[2] := integer(VV.Value = Opened);
    DatatoVisSim[3]:= integer(TurboPumpSwitch.Down = True);
    DatatoVisSim[4] := integer(V2.IsOpen());
    DatatoVisSim[5] := integer(MechPumpSwitch.Down = True);
    DatatoVisSim[6] := integer(V1.Value = Opened);
  end;
end;
```

Figure 7. Example callback procedure that maps user interface controls to simulator inputs

TDesignParameters. Manipulating simulation design parameters is one way to help learners further explore how a system works. Creating a custom dialog box that allows learners to specify design parameters such as chamber volume and turbo pump speed would be very tedious and time consuming, especially when there are several design parameters. With SimPLE, educators can use TDesignParameters, accessible from the developer kit notebook page, to create design parameters controls on-the-fly simply by specifying the parameter name, default value, and VisSim™ variable number (Figure 8). The parameter controls will appear automatically on the simulation parameters page, along with a comment area, and a log that records any changes made by learners (Figure 9). A logical extension would be to allow multiple parameter settings to be saved and loaded.

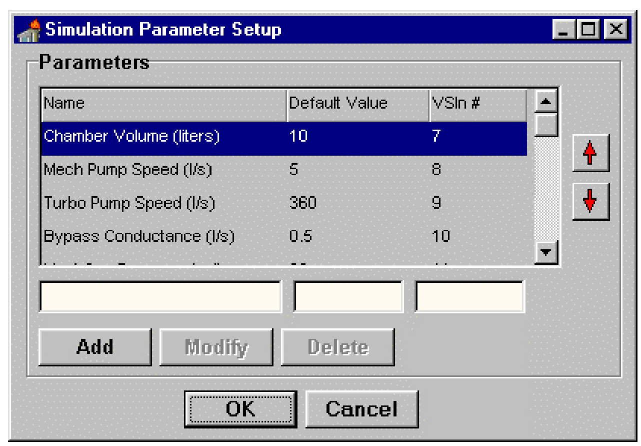

Figure 8. TDesignParameters dialog box for creating simulation design parameter controls

TActionLinks. Many simulation-based learning environments do not provide any guidance material and when it is provided there is normally no coupling between the simulation and the guidance material. The result feels like two independent systems. We wanted our learners to have the benefits of a tight coupling between the simulation and guidance material while still maintaining the separation that benefited developers. TActionLinks links the guidance material to the simulation panel while still maintaining the desired separation. Specifically, TActionLinks extends the HTML syntax used to create the guidance material to support simulation actions. The actions are processed as the result of an error being raised when an unknown URL is encountered. For example, the HTML link "Highlight [object1 object2 …]" is used to highlight multiple objects and "PlaySound [filename]" is used to play audio files. This simple notation can be extended to include a variety of other actions, such as playing video or controlling a stopwatch.

Since the objects are different for each simulation, TActionLinks automatically generates the list of object names, along with instructions and samples of each. This information is automatically displayed on the developer kit page for educators to consult when creating the guidance material (Figure 10).

TCommunication. Communication and collaboration is another important aspect that learning environments must support. Learners need to be able to communicate with other learners, with their instructors when they have problems, and even with other domain experts. TCommunication builds on a commercial Delphi email component to provide a basic facility for communicating about the system. It allows learners to easily attach files, like the simulation model and parameter log, to their messages with the click of a button. It also provides a setup dialog box that educators can use to create a default list of email addresses for the learners (e.g., class list). Educators are also interested in how TCommunication might help them manage the email they receive. For example, one suggestion is to automatically annotate the messages with a few words specified by the instructor that would make it easier to sort messages by class or project.

FUTURE WORK AND CONCLUSIONS

We are in the process of selecting different application domains and identifying educators interested in building SimPLE learning environments that will further test our framework. Until now the environments have been developed by graduate students at the university. For example, a civil engineering graduate student with no Delphi or VisSim experience (but with computer experience) created the NileSim environment in only two months (without many of the tools described since they were still under development). While SimPLE makes it possible for educators with limited software development experience to build simulation-based learning environments, we believe the primary role of the educators will be as content authors, with user interface development delegated to those with more computer experience.

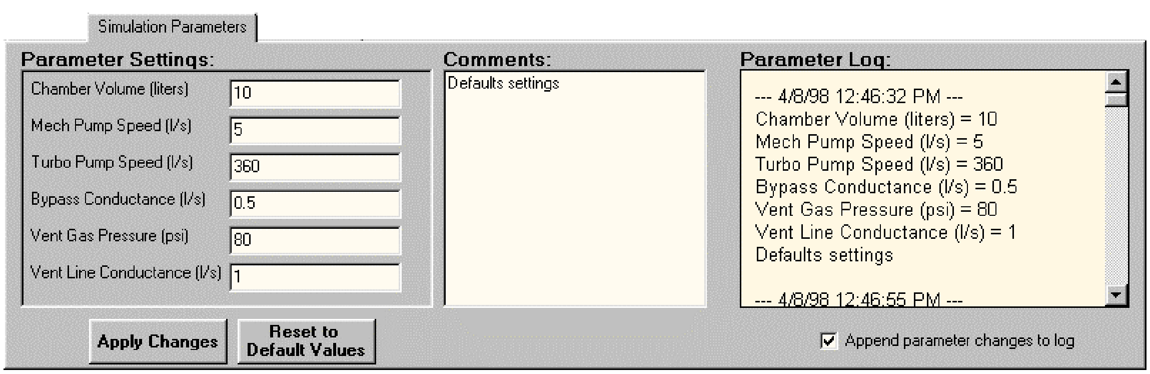

Figure 9. Simulation parameter controls created as the result of TDesignParameters

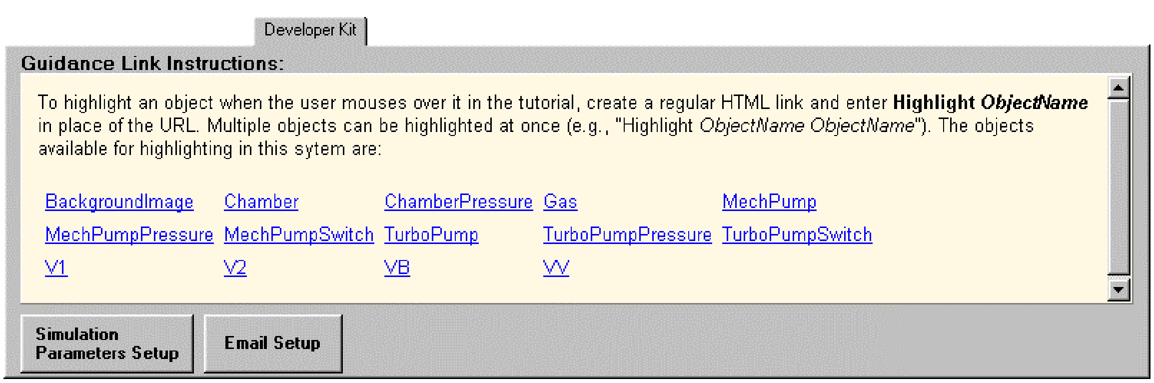

Figure 10: Developer Kit page showing automatically generated guidance link instructions

Initial feedback from both learners and educators has been very positive. We are in the process of conducting a pilot study comparing the use of SimPLE (VisSim + front end) to VisSim only to teach students about a vacuum pump system. Our initial pilot results show a strong preference for the SimPLE version.

We are expanding SimPLE to make developing simulation-based learning environments easier to build. Specifically, we plan to create more software aids, to enhance the existing ones, and to investigate the use of other simulation engines, like Excel. One software aid we have started to explore creates learning histories. At the simplest level, learning histories would capture the sequence of user actions and allow them to be played back. These histories could then be used by learners when posing questions (e.g. why did this happen?), or created by educators and replayed in the guidance material, or they might even be turned in as homework (e.g., demonstrate the quickest way to create the lowest pressure in the chamber).

While there are numerous examples showing the benefits of learning environments, we believe that authoring tools, like SimPLE, that support their creation will be critical to their wide spread success. While our framework is still rudimentary, it is a key step in making simulation-based authoring tools accessible to educators.

ACKNOWLEDGMENTS

We thank Catherine Plaisant and Ben Shneiderman for thoughtful reviews and Yatin Sankholkar for implementation help. We also thank Ben Levy and Gregory Baecher for creating NileSim. This work is supported by the National Science Foundation under grant EEC 96-96212.

REFERENCES

[1] Ackerman, M. (1995) Social Activity Indicators: Interface Components for CSCW Systems, *Proceedings of UIST '95*, ACM, New York, 159-168.

[2] Cole, R. and Tooker, S. (1996). Physics to Go: Web-based tutorials for CoLoS physics simulations, *Proceedings of Frontiers in Education '96*, IEEE, 681-683.

[3] Edelson, D., Pea, R., and Gomez, L. (1996). Constructivism in the Collaboratory. In B. G. Wilson (Ed.), *Constructivist learning environments: Case studies in instructional design*, Educational Technology Publications, Englewood Cliffs, NJ, 151-164.

[4] Eden, H., Eisenberg, M., Fischer, G., and Repenning, A. (1996). Making Learning a Part of Life. *Communications of the ACM 39*, 4, 40-42.

[5] Hudson, S. and Smith, I. (1997). Supporting Dynamic Downloadable Appearances in an Extensible User Interface Toolkit, *Proceedings of UIST '97*, ACM, New York, 159-168..

[6] Jones, P. M. and Schneider, K. J. (1996). Learning environment for magnetic resonance spectroscopy (LEMRS): Supporting apprenticeships learning in operational environments. *Journal of Educational Multimedia and Hypermedia 5*, 2, 151-177.

[7] Kafai, Y., and Resnick, M., eds. (1996). *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World.* Mahwah, NJ: Lawrence Erlbaum.

[8] Lu, G. B., Oveissi, M., Eckard, D., and Rubloff, G. (1996). Education in semiconductor manufacturing processes through physically-based dynamic simulation. *Proceedings of Frontiers in Education '96*, IEEE, 250-253.

[9] Nahvi, M. (1996). Dynamics of student-computer interaction in a simulation environment: Reflections on curricular issues. *Proceedings of Frontiers in Education '96*, IEEE, 1383-1386.

[10] Norman, D. and Spohrer, J. (1996). Learner-Centered Education, *Communications of the ACM 39*, 4, 24-27.

[11] Rosson, M. and Carroll, J. (1996). Scaffolded examples for learning object-oriented design. *Communications of the ACM 39,* 4, 46-47.

[12] Sebrechts, M., Silverman, B., Boehm-Davis, D., and Norman, K. (1995). Establishing an electronic collaborative learning environment in a university consortium: The CIRCLE project. *Computers in Education 25*, 215-225.

[13] Soloway, E., Jackson, S., Klein, J., Quintana, C., Reed, J., Spitulnik, J., Stratford, S., Studer, S., Jul, S., Eng, J., and Scala, N. (1996). Learning theory in practice: Case studies of learner-centered design. *Proceedings of CHI '96*, ACM, New York, 189-196.

[14] Woolf, B. and Hall, W. (1995). Multimedia pedagogues: Interactive systems for teaching and learning. *IEEE Computer*, May 1995, 74-80.