

# The Design of Guided Learner-Adaptable Scaffolding in Interactive Learning Environments

Shari L. Jackson, Joseph Krajcik, Elliot Soloway

College of Engineering, School of Education, School of Information  
University of Michigan  
1101 Beal Ave.  
Ann Arbor, MI 48109, USA  
E-mail: sjackson@umich.edu

## ABSTRACT

To address the needs of a population of users who are also learners, the learner-centered design of software suggests the need to design scaffolding—fadeable supports—into educational tools. This paper describes a particular approach, *Guided Learner-Adaptable Scaffolding (GLAS)*, in which the learner controls the changing and fading of scaffolding, with guidance and support provided by the system. Using this approach, we have developed a tool, *TheoryBuilder*, that supports learners in building and testing dynamic models of complex systems. We have conducted extensive classroom testing with ninth grade students who used the tool several times throughout a year. An analysis of the data demonstrates the success of the GLAS approach in developing an adaptable tool to support diversity and the development of expertise.

## Keywords

## INTRODUCTION

In our recent work, we have proposed a new methodology for the learner-centered design (LCD) of software, to address the needs of a population of users who are also learners (Soloway, et al., 1994; 1996). Learners often require extra support and motivation to engage in unfamiliar tasks. Learners are a diverse population, varying in knowledge, skills, interests, and learning style. And finally, learners will *learn*—as they grow and develop understanding and expertise over time and through interaction with the software, their needs for software support and functionality will change as well.

The needs of learners suggests the use of *scaffolding* in educational software. Scaffolding refers to support provided so that the learner can engage in activities that would otherwise be beyond their abilities. Scaffolding should also support the learning of those activities, so that it can gradually be faded as it is no longer necessary.

Scaffolding, as provided by human tutors, has been well-established as an effective means of supporting learning (Wood, Bruner, & Ross, 1975; Collins, Brown, & Newman, 1989; Rogoff, 1990). Building scaffolding into software offers the opportunity to support diversity—through individualized support that accommodates learners of different skills, backgrounds, and learning

styles, and growth—through options that provide more powerful functionality as the learner develops expertise.

A critical component of scaffolding is therefore that it be capable of fading—as the learner’s understanding and abilities improve, the computer, much like a human tutor, needs to back off and give the learner more autonomy, fewer hints, etc. In the field of educational software research, scaffolding is a new concept that is still being defined [Collins et al., ?; Rogoff, 1990; Guzdial, ?]. Many techniques have been explored that provide various supportive structures for learners, but typically that support does not fade within the software itself. Further extending the problem to address how best to facilitate that fading, then, is even more of an open question.

The HCI problem we have addressed is therefore how to design software with fadeable scaffolding. We have developed an approach of *Guided Learner-Adaptable Scaffolding (GLAS)*. GLAS is designed as discrete, gradual layers to provide fading through smooth transitions. Fading between layers is under control of the learner, with guidance from the software to aid the learner in making informed decisions. We have developed scaffolding design and fading mechanisms, based on the GLAS approach, for a wide range of scaffolding types.

GLAS has been applied to the design of *TheoryBuilder*, a learner-centered tool to support the complex and educationally valuable task of scientific modeling and theory building. *TheoryBuilder* builds on our work with *Model-It* [Jackson, 1996; Stratford, 1996, Jackson, 1997]. Research with *Model-It* has provided us with a great deal of information about the kinds of supports and functionality that learners need to engage in modeling activities. *TheoryBuilder* incorporates a broad range of scaffolding to specifically target those needs. And whereas *Model-It*’s supports did not fade, *TheoryBuilder* implements the GLAS approach, providing fading of each of its scaffolds under the learner’s control.

To evaluate the effectiveness of the GLAS design strategies, *TheoryBuilder* has been thoroughly tested in actual classroom use with 100 students over several projects throughout a school year. Data collected include the models that the students constructed, software-generated log files, and videotapes of student conversations and computer activity. The data is being

analyzed in-depth and used to evaluate the impact of the specific GLAS techniques proposed for implementing fadeable scaffolding, as well as the overall success of the GLAS approach.

In this paper, we first provide some background on our previous research. We next describe GLAS and its application to the TheoryBuilder software. Then, we describe our study: its methods, data, and results. We will present both an overview of the entire set of data, and a qualitative analysis of two representative students' use of the program over time.

## BACKGROUND

TheoryBuilder represents the next phase of our research with Model-It—a tool to make modeling accessible to high school science students (Jackson et al., 1994, 1996). Model-It has a learner-centered design that hides much of the complexity of the modeling task, provides grounded concrete visual representations, bridges between verbal and graphical representations, and tightly links user actions and events. With Model-It, students can easily construct, verify, and analyze qualitative models by simply linking together the factors that comprise high-level objects.

To build a model, the student begins by selecting from or creating a set of objects, represented by photo-realistic graphical icons. Then the student defines factors, measurable quantities or characteristics associated with each object. Finally, the student defines the relationships between the factors, either qualitatively, (Figure 1) or quantitatively, by clicking on the points of the graph or inputting a table of values.

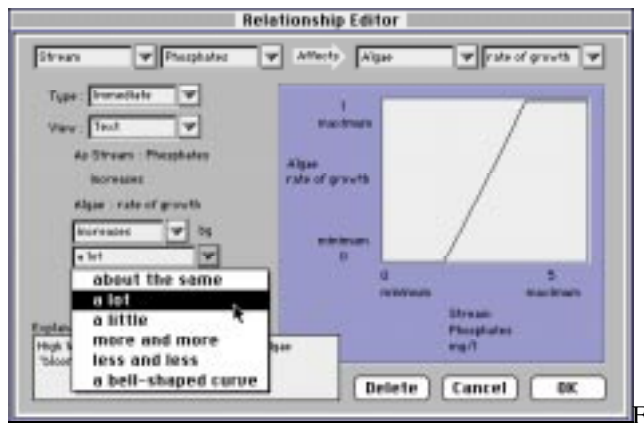


figure 1: qualitative relationship definition

As students build and test their model, they can view their model either with a high-level “world” view of the objects in the model, or an interactive “map” view in which icons representing each object’s factors are linked together with clickable arrows that represent the relationships (Figure 2). Once objects, factors, and relationships have been defined, students can run their models to see the results. Meters show the current value of a chosen factor, and graphs show the values of different factors as they change over time. While a model is running, students can change the value of

factors in real-time, and the meters and graphs will update to show the results (Figure 2).

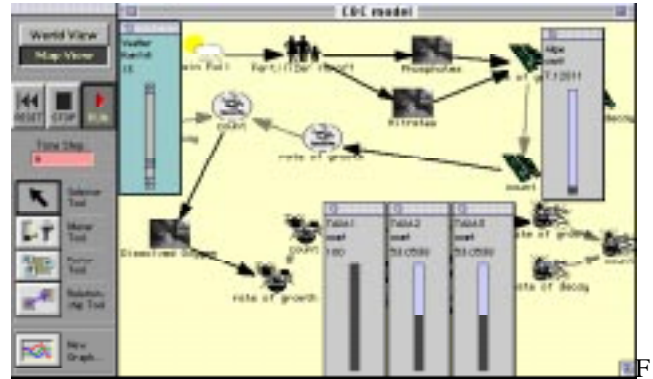


figure 2: Map view of a model, with meters open.

Versions of Model-It have been used by hundreds of ninth and tenth grade students at a local high school over the past four years. Four significant classroom studies were conducted, each culminating in an open-ended modeling task, where students were asked to create their own models to represent some chosen environmental phenomenon. The final models typically exhibited reasonable scientific validity and significant complexity, reflecting student understanding about a range of topics, from habitat and biodiversity to the impact of man-made pollution sources on an ecosystem.

With Model-It, students appropriately focused on high level concepts—defining the factors in the system and the relationships between them, running simulations with different factor values, and determining the validity of the results. Students were comfortable expressing themselves qualitatively, and once they had done the thought work of conceiving of a relationship, they were able to quickly add it to their model. We found significant evidence that Model-It can make modeling accessible to high school science students. (Jackson et al., 1996; Stratford, 1996).

Our research also suggested areas where students would have benefited from additional support to learn the task and to engage in modeling activities. For example, our previous studies included up to six 50-minute class periods of self-paced tutorial instruction to the software, but by starting with a simpler and more qualitative interface, with interactive help and guidance, the time needed for students to start building models could be significantly shortened. Research on the cognitive activities in which students engaged while building models found that some students required more support in order to understand and engage in these activities—for example, planning, making predictions, testing and evaluation (Stratford, 1996). Finally, interviews with students and teachers suggested advanced functionality that might be useful, for example, a weighted average to show relative importance of variables, and delay so that relationships don’t happen all at once.

The next phase of our project, then, had two goals. First, to broaden the range of supports provided by the

program, in order to address the issues highlighted by our research. And second, to specifically address the idea of what it means to implement *scaffolding* in learning environments, by designing supports that can be faded as the learner's understanding and abilities improve.

### **APPROACH: GUIDED LEARNER-ADAPTABLE SCAFFOLDING**

We call our approach *Guided Learner-Adaptable Scaffolding* (GLAS) to emphasize that, in keeping with our goals of designing tools for learners to create artifacts of their own design (as opposed to, e.g., tutoring programs in which the software directs the interaction), the learner will also be in control of the changing and fading of scaffolding. However, we recognize the potential need for guidance from the system (i.e., to supplement, not to replace the classroom teacher) in making those decisions. The goal is for the student, with the software's help, to be able to take on some of the responsibility for his or her learning.

The basic idea of GLAS is that scaffolding should be designed in layers, so that fading is a gradual change in the level of support provided. The system should provide guidance to the learner in making decisions about adapting the scaffolding, for example, through context-sensitive information associated with fading options. For purposes of our research, and based on the literature [Collins et al., 1989; Rogoff, 1990; Guzdial, 1995] we have also more specifically defined scaffolding as covering the following three categories. For each, we describe its implementation in TheoryBuilder and its fading, summarized in Table 1.

#### **Supportive scaffolding**

Supportive scaffolding is support for doing the task. The task itself is unchanged; scaffolding is provided alongside the task, to offer advice and support, e.g., demonstrative examples, context-specific help about what to do next. Examples of supportive scaffolding are guiding, coaching, and modeling. As supportive scaffolding fades, the task is the same as it was before, but the goal is for the learner to have internalized the procedures and concepts which had been scaffolded.

In Model-It, guiding scaffolding is provided through messages which appear as relevant (Figure 3), and which can be faded through an associated "Stop reminding me" button. However, if the learner continues to neglect an activity after fading a reminder, eventually a message will reappear to say "I know you asked me not to remind you about this, but I noticed that you haven't been..." Note that, in keeping with the learner-centered approach in which the learner is in control of the tool and not vice versa, these reminders are phrased as suggestions; they do not force the learner into an action, and can be ignored.

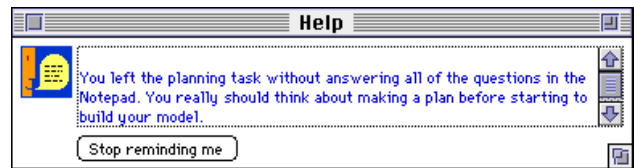


Figure 3: Example of a guiding scaffold. This scaffold is faded by clicking the "Stop reminding me" button

Coaching and modeling scaffolds provide help and examples explaining the various modeling concepts. They are invoked through contextualized help buttons that appear throughout the software, usually "?" (Figure 4) or "Show me an example". When invoked, they display explanations and examples using both text and pictures. Being passive scaffolds, these buttons do not fade per se, but rather are faded simply through not being invoked. We did experiment with causing the help to appear automatically the first time the user encountered an interface element (e.g., the first time the user clicked in a reflective text field, help appeared to explain the question in more detail), but it was found that users mostly ignored the help unless they had invoked it themselves.

#### **Reflective scaffolding**

Reflective scaffolding is support for thinking about the task (e.g., planning, making predictions, evaluating). It also doesn't change the task itself, but instead it makes the task of reflection explicit by eliciting articulation from the learner. A crucial part of learning is this reflection by the learner about what he is doing, is it working, what are his goals, etc. Norman makes a distinction between two kinds of cognition, *experiential* and *reflective*, and asserts that reflective cognition, the mode of "comparison and contrast, of thought, of decision-making," is essential for learning, although more rarely supported by technology [ref.].

Reflective scaffolding in TheoryBuilder is provided by the Notepad window (Figure 4), and by the description fields that are part of each component's editing window. The learner reflects by typing plans, descriptions, predictions and evaluations into the appropriate fields of the Notepad, as relevant for each subtask. Many of the guiding scaffolds refer to these text fields, prompting the learner to fill them in when appropriate, as in Figure 11 above. Another passive scaffold, the text fields themselves do not fade, but once the reminders are faded they are more easily ignored.



Figure 4: The Notepad page for the Test subtask

### Intrinsic scaffolding

Intrinsic scaffolding is our name for supports that change the task itself, by reducing the complexity of the task and focusing the learner's attention (e.g., training wheels on a bicycle, outlines in coloring books), or by providing mechanisms for visualizing or thinking about a concept (e.g., maps and models for visualization) [Carroll, et. al., 1984]. As the scaffold fades, the task is changed, but associations should remain so that the learner can progress from simpler, more structured, or more concrete tasks to variations in which more of the underlying complexity or abstractness is introduced.

Intrinsic scaffolding is implemented in TheoryBuilder as defaults which hide all but the simplest tools from the novice learner, but make advanced features available as the learner grows and develops expertise. Intrinsic scaffolding is manifested as different views and representations of model components, or different sets of enabled controls and tools. Figure 5 shows the mechanism by which the learner fades intrinsic scaffolding for the relationship editor by turning on advanced options that were previously hidden. The dialog also provides guidance for selecting appropriate options by showing what each option will let the user do. Other advanced features accessible through Options dialogs are listed in Table 1.

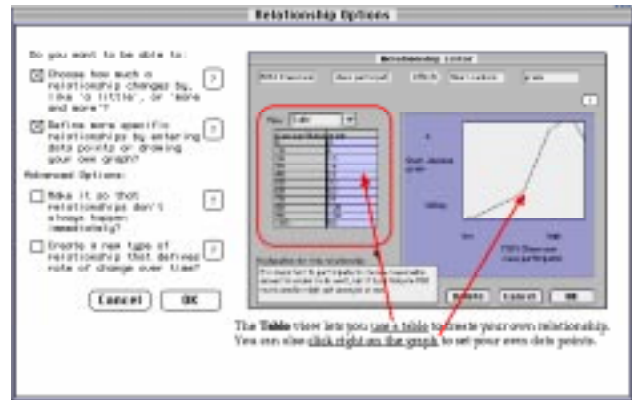


Figure 5: Example of the fading mechanism for intrinsic scaffolding for the Relationship Editor.

Scaff. Type	Model-It Implementation	Fading
Supportive Scaffolding	Guiding through subtasks (e.g., plan before building, test periodically). Coaching and modeling throughout the software, providing context sensitive help and examples.	Guiding faded with "Stop Reminding Me" button Coaching, modeling are invoked by learner; "faded" as not invoked.
Reflective Scaffolding	Eliciting articulation with forms and prompts for: <ul style="list-style-type: none"> <li>• Planning: stating goals of model, ideas for building it.</li> <li>• Explanations for objects, factors, and relationships.</li> <li>• Testing: predicting before test, evaluate result of test.</li> <li>• Evaluating: deciding how well the model works, what could be changed.</li> </ul>	Optional and available. "Faded" as ignored. Also linked to guiding scaffolding - reminders to fill in each form - faded with "Stop Reminding Me" buttons.
Intrinsic Scaffolding	Multiple, linked representations (from simple to advanced): <ul style="list-style-type: none"> <li>• Factor definition (qualitative text labels, numGregal values)</li> <li>• Relationship definition (textual, graphical, table, rate)</li> <li>• Viewing model (world view, map view)</li> <li>• Viewing factor values (meters, graphs)</li> </ul> Hiding complexity, but making advanced options available: <ul style="list-style-type: none"> <li>• Weighting - set relative weights of relationships.</li> <li>• Delay - specify time duration for relationship to take effect.</li> <li>• Combining - combine effects of relationships in different ways (e.g., sum, product, difference, maximum, minimum)</li> </ul>	"View" menus let user switch between representations. Often different views are displayed simultaneously as linked representations.  "Options" buttons open structured dialog boxes that allow options to be turned on and off, with associated help to guide the learner in making decisions.

Table 1: Fadeable scaffolding strategies and their implementation in TheoryBuilder

## METHODS

We worked with a four classes of 9<sup>th</sup> graders (about 100 13 and 14-year-olds), who used TheoryBuilder for three modeling projects over a school year, each project taking about 4 hours over one week, and the projects spaced about two months apart. For each project, the students, in pairs, built models to represent their understanding of some aspect of the topic currently being studied by the class. The specific goal, content and structure of their models was determined by the students themselves. Formal instruction on the software was kept to a minimum: for the first project, students were introduced to the program via a one-hour, self-paced tutorial, and in successive projects, a 20-30 minute demo and discussion, supplemented by handouts, was used to make students aware of advanced features of the program.

In the first half of the year, curriculum centered around a study of stream ecosystems. For the first modeling project, in November, students built models of the physical and biological factors of a stream, and for the second, in January, students chose their own topic relating to some aspect of the whole stream ecosystem, including physical, biological, and chemical factors. In the spring, the class studied weather and global climate issues such as acid rain, global warming, and the greenhouse effect. For the third project, students chose a research question based on these issues, used the internet to search for information, and built models to demonstrate their theory or understanding about the topic they had researched, using data from their research where appropriate.

In order to achieve an in-depth understanding of students' usage of and reactions to the software, a focus group of eight pairs of students was chosen as a representative subset. For this focus group, process video was collected continuously during each project (approx. 100 hours of videotape), consisting of audio recordings of their conversations coupled with video recordings of their computer screen interactions. Post-interviews with these pairs were conducted to further clarify their motivations for their activities and reactions to specific scaffolding and fading mechanisms.

Data collected from all students included the models created for each project, and software-generated log files that record student interactions with the program. This data will be used to compile statistical information about the use of scaffolding and fading mechanisms over time. Changes in the size and complexity of the models built by the students over the three projects may show the students' development of understanding and expertise in model-building. The process video data and interviews of the focus group will be used to identify specific use of and reaction to each specific scaffolding component, and overall, to determine the success of the GLAS approach in supporting students learning the task and adapting and fading the software to meet their changing needs.

## RESULTS

### Overview

The data from these studies is currently being evaluated. Preliminary analysis of the full set of models indicates that the software was quite successful. For the first project students were able to use the basic program to build useful models with only brief instruction, aided by the supportive scaffolding, and shielded by the intrinsic scaffolding from the potentially confusing array of options. In successive projects, a short introduction to advanced features was sufficient for students to choose to fade selective scaffolds and build larger, more complex, and more accurate models.

**Supportive scaffolding:** The use of coaching and modeling scaffolding faded over time, as students learned how to use the software, although help that was associated with advanced features continued to be used as these features were turned on. More students faded guiding scaffolds over time as they learned the task; by the third project 63% of the students had turned off at least one reminder. Others simply learned to ignore the reminders. In general, in later projects students tended to ignore all supportive scaffolding unless they had invoked it themselves (e.g., by clicking on a help button).

Table 2 shows how often each reminder was faded, for each project<sup>1</sup>. The most commonly faded reminder, faded in the second and third projects by 45% of the students, was the reminder to test the model periodically. This reminder first appeared after three relationships had been created or modified, and every five relationships thereafter, so it was probably the most frequently encountered. Despite turning off the reminder, students tested their models more frequently in later projects, suggesting that the strategy had been internalized. The reminder itself, however, could be annoying, and having made its point, students were pleased at the ability to turn it off. Other reminders, such as those to open meters, or to fill in descriptions and explanations for objects, factors, and relationships, were faded less frequently, because students usually did these tasks without prompting and therefore encountered the reminders less often. Reminders to reflect are discussed in next.

Reminder	Proj 1	Proj 2	Proj 3
Fill in plan before building	1	9	8
Describe objects	0	6	5
Describe factors	2	9	11
Explain relationships	2	9	14
Test periodically	3	20	19
Predict before testing	1	14	18
Open meters before testing	2	5	7
Assess results after testing	3	14	18

<sup>1</sup> Regretably, there was no mechanism for remembering student preferences between projects, so all reminders were initially turned on at the start of each project.

Table 2: Number of models (out of 45) in which each reminder was faded, over the three projects.

**Reflective scaffolding:** Students were required by the teacher to fill in the text fields that comprise reflective scaffolding (plans, descriptions, explanations, evaluations) as part of their grade. As a result, rather than fading, there tended to be increased and more articulate usage of these fields in successive projects. The exceptions were the prediction and assessment questions that accompanied the testing task, which were not printed out and therefore not graded. The result was that students tended to skip these questions in later projects, and in fact, as Table 2 shows, the guiding reminders to fill in these fields were the second and third most commonly faded supportive scaffolds.

As further evidence that students had developed the thoughtful habit of filling out reflective scaffolding, we saw that even though there was no reminder to fill in the explanation field for the advanced weighting and combining tool, students still tended to write an explanation.

**Intrinsic scaffolding:** More students faded the intrinsic scaffolding, i.e., used more of the advanced features, in later projects as they developed expertise, as shown in Table 3. For example, for the first project the only options used by any students were the most basic: 20% of the student pairs used the option of specifying the slope of a relationship, and 9% used the table option to draw their own relationship graph. By the third project 85% of the models used one or both of those options, and of the most advanced options, 41% used delay and 26% used rate relationships, 28% used the weighting tool, and 11% used the combining function to calculate the product or difference of two factors' effects.

The table option was used most often in the second project, because the textbook used for that topic included a number of applicable graphs, while for the third project, on climate, the resources that students were able to find usually did not include that level of detail. The third project did see the most use of the other advanced options, even the rate relationships which our previous research showed was rarely, if ever, used. Here, though, owing perhaps to the growth of expertise due to the increased time students had spent with the program, many students were able to use rate relationships to model such temporal phenomena as the release of greenhouse gasses, the depletion of the ozone layer, or the melting of the polar ice caps.

	Num-ber	Slope	Table	Delay	Rate	Weighti-ng	Combi-nations
1	0	20	9	0	0	0	0
2	37	78	54	30	4	20	0
3	50	72	37	41	26	28	11

Table 3: Percentage of models using each advanced option, over the three projects [make table vertical if room]

## User Profiles

In this section, we present a qualitative analysis of two students from our focus group. These two students represent the diversity of our learner population, having very different backgrounds, approaches, and learning styles<sup>2</sup>.

### User Profile 1: Greg

Greg is mathematically and logically inclined, even enrolled in a higher level math course than most of his classmates. He is quick to catch on, and from the first day of the project, branched out to explore and try new ideas. His approach to TheoryBuilder is to turn all advanced options on and try them out, pushing the computational limits of the software. He is thoughtful about model-building, reasoning about chains of relationships. He tends to be impatient about reminders, which are mostly about filling in the reflective scaffolds that he would rather ignore, although he reflects verbally as he builds.

Figure 6 shows the model created by Greg and his partner for the second project. It shows the effect of precipitation on water quality. The emphasis is on how stream discharge affects dissolved oxygen, with contradictory effects: a small (“by a little”) direct increase due to riffles stirring up the water, and a longer term decrease due to erosion increasing the turbidity (cloudiness) of the water, raising the temperature and lowering the dissolved oxygen it can hold. They wrote good explanations, used advanced features well, especially the Table view, e.g., even with no contaminants, the stream still has some natural turbidity.

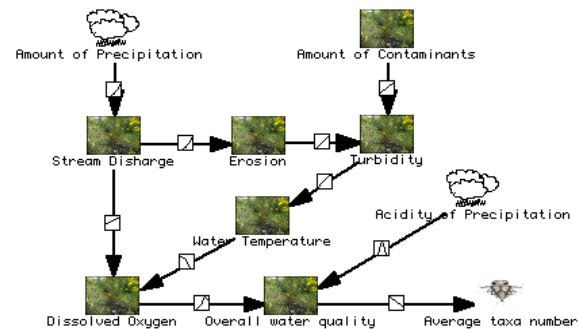


Figure 6: Greg's model for Project 2

Figure 7 shows the model Greg built for project 3. Greg had read that an increase in global temperature would increase storm severity, and he built a model to express his theory about why this would occur: that flooding would cause greater humidity in coastal areas as compared to inland areas, and this difference would create a pressure differential that would increase storm severity. To implement this model, Greg used all the advanced features of TheoryBuilder, including the

<sup>2</sup> Although students worked in pairs, each of these focus students tended to be the dominant partner of their pairs, so the models are a good reflection of their attitudes.

combining tool to calculate the difference in humidity. He also used feedback loops, something that was never explicitly taught, in order to show that the more water there is on land the more is evaporated, and in turn the less water is left on land.

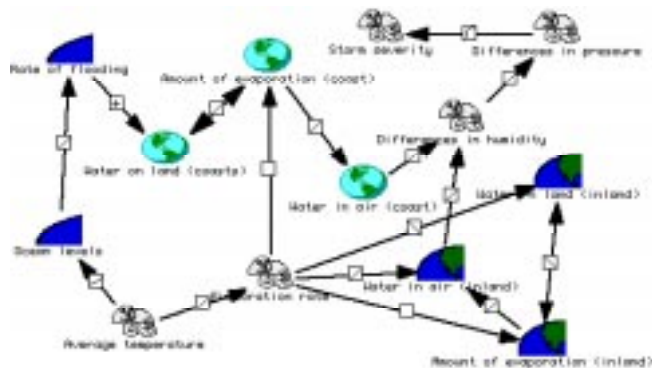


Figure 7: Greg's model for Project 3

*User Profile 2: Amanda*

Amanda describes herself as “not comfortable with computers.” Nevertheless, she applies herself to the modeling task and tries to learn the software, focusing on what is necessary to get a good grade. She has a very good understanding of the scientific content, but doesn't always understand how to translate that into a model. She spends a lot of time writing detailed descriptions and explanations of the components of her model and how they relate, but she tends to stick with the simpler layers of functionality that are easier for her to understand.

Figure 8 shows the model Amanda built for Project 2. Her goal was to show where dissolved oxygen comes from and how it affects the stream organisms. Amanda and her partner had some misunderstandings about how to use the program, but they managed to express their ideas, and wrote detailed explanations that demonstrated their scientific knowledge. During the building of this model, a fellow classmate “helped” them by turning on all the advanced relationship options, and the teacher, looking at their model, showed them how to use the Table view. But while they followed others' suggestions, left on their own they stayed with the most basic options.

Figure 9 shows Amanda's third project, modeling the human impact on ozone depletion, and suggesting that alternatives to CFC-producing products could help alleviate the problem. In building this model, Amanda's goal was to get the basic model working, and add advanced options later. In fact, they did manage to add weighting and combining at the end, to make their model more accurate. When asked how her use of the program had changed over time, Amanda said that even though she tried to keep things simple, she felt she “understood it better now and I think we did a more thorough, better job on this one.”

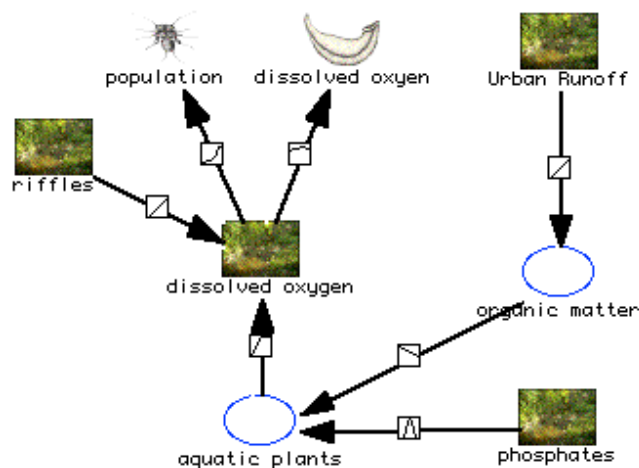


Figure 8: Amanda's model for Project 2

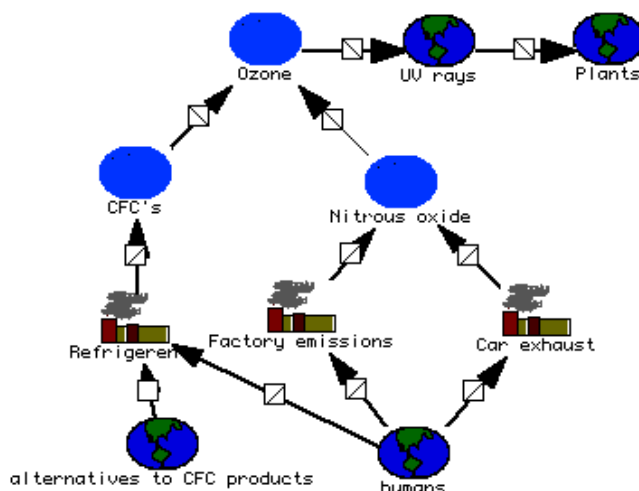


Figure 9: Amanda's model for Project 3

**Supportive scaffolding:** Both Greg and Amanda used the coaching help buttons the most in the first project, Greg slightly more often in keeping with his exploratory learning style. The only uses of coaching in later projects were to learn advanced options such as rate or weighting. In post-interviews, both said the buttons were most helpful early on, but they were “nice to have around.”

The reminders were also most useful at the beginning. During the first project, each read and followed a reminder 4 or 5 times, but in the second and third projects, reminders were only followed once, and usually ignored or faded. The reminders did help with learning the task; for example, both students internalized the habit of periodically testing the model, and realized that it did help them find and correct mistakes. Greg was more impatient with the reminders and used the “Stop reminding me” button to fade them, while Amanda thought they were definitely helpful at the beginning, and even in the third project when she mostly ignored them, she said that they “didn't bug me 'cause I knew I could turn them off whenever I wanted.”

**Reflective scaffolding:** Amanda thought that writing explanations for each relationship "helped so that you felt like you could explain the model more... and it made the relationships clear in your mind, 'cause you knew exactly how everything affected everything." Amanda even felt that planning ought to be required, because if not she might skip it, and then not be making the best use of her time. Greg was more impatient with the reflective scaffolds, describing himself as a "verbal thinker" and feeling that he already had the whole model planned out in his mind, and just wanted to build it. He thought the testing questions were particularly "pointless, because I can think of what I expect to happen and compare it to what happens without writing it down." Both students liked the evaluation questions, however; Amanda because it helped her plan what to say in her presentation, and Greg because he was proud of his model.

**Intrinsic scaffolding:** Greg and Amanda were most different in their approach to and use of advanced features. Greg tended to turn them all on right away and try them out. He said that while he understood that options were hidden so as not to confuse students who didn't want to use them, if he were the only user he would like them all on from the beginning. Amanda said that they had chosen a simple project and just used what they needed to make it work. When asked what she thought the program would be like if it started with all the advanced options visible, she said "I think it would have been confusing.... But if it would've been explained really well, probably we could've done it, I don't know."

### Discussion

In analyzing the effectiveness of GLAS, we recall our first and fundamental motivation in designing learner-centered software—to make the task accessible. Here we certainly feel that we succeeded. Second, the motivation for providing *scaffolding* was to allow students to gradually adapt the software to meet their changing needs. Students turned off or ignored help features that they no longer needed, and accessed more advanced features as they felt ready.

Model-It was successful where students could think in terms of the domain concepts, not "programming" a model, but representing their ideas in a way that was natural to them. Similarly, we note that the advanced options designed into TheoryBuilder were successful in proportion to how qualitatively and conceptually they were presented. For example, the advanced option of *delay* allowed students to specify whether a relationship ought to happen "immediately, quickly, or slowly." Underneath, this was a complex mathematical function, but it is presented in terms the students understand, so it became one of the most frequently used features.

Another option, the *weighted average*, was a little more advanced but it still fit well with students' thinking, and was used correctly by many students. As Amanda said while building her third model, "We want to weight it so nitrous oxide has less of an effect on ozone depletion

than CFC's do." Students found the interface understandable, although a qualitative option might also have been useful since students did not always have quantitative data—Amanda noted that, not having found data, their choice of relative percentages was "an educated guess." (Figure 10).

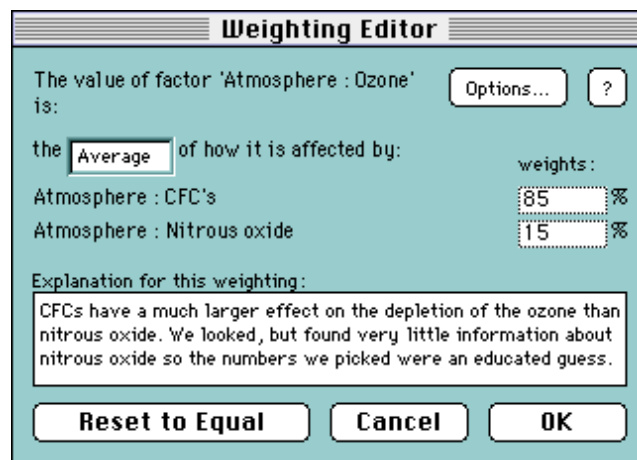


Figure 10: Weighting in Amanda's 3rd model

The most advanced option was that of *combining* functions—choosing a function (average, sum, product, difference) used to combine the effects of different factors. Students who were used to thinking mathematically had no trouble with this concept, for example, Greg found a perfect use for it in calculating the difference between coastal and inland humidity. In presenting the option to the class, however, we tried to help students think about it conceptually. For example, if the combined effect depends on two factors each having a high effect, than a product might be most appropriate. Amanda, remembering this explanation, wanted to use it for her third model to show that CFC's are high if and only if there are both a large population of humans, and a low usage of non-CFC producing alternatives. She had discovered during testing that, because of the default which averages the effects of multiple factors, her model was showing a medium value of CFC's being produced, even when the human population was set to 0, if she also set alternatives to 0.

The implementation of the combining option, however, was awkward and took many steps. She would first need to realize that a multiplying combination was called for. Then, because each factor has a range of possible values that defaults to 0-100, she would have to change the range of one of the factors to a 0-1 range so that the product would still be in the 0-100 range. Basically, our design had fallen out of the conceptual level into lower level mathematical programming, and few students could follow there on their own. With help, Amanda was able to implement the function and achieve her goal, but when asked later if she understood it she said that "I understand the concept of what we did, but I might not have been able to do that on the model." Clearly more scaffolding is needed, to bridge from conceptual to mathematical representations. The interface should

allow students to specify the desired effect (perhaps by filling in sentences about when  $x$  is high, and  $y$  is low, then  $z$  is {low, medium, or high}). Once students had mastered this level, fading from conceptual to mathematical could help students see how their ideas are translated into mathematical equations.

## RELATED WORK

The HCI research that is relevant to our work covers the general topic of the design of adaptive interfaces (those that change automatically) and adaptable interfaces (those that are changeable by the user). (DietGreh et al., 1993; Fischer, 1993). Similarly, scaffolding can be adaptive, adaptable, or some combination:

*Adaptive Scaffolding* is found in many computer-aided instruction and intelligent tutoring systems, which vary their coaching and critiquing advice, or the problem posed, based on an evaluation of the learner's understanding. (E.g., Lesgold, Eggen, Katz & Rao, 1992; Lajoie, 1993). The user model determines the amount of help or level of problem initially offered to the student, and in turn, the student's reaction is used to adjust the user model.

The main problem is that an extensive model of the learner's knowledge may be hard to specify or evaluate in more open-ended domains. Also, because the evaluation may sometimes be incorrect or at least incomplete (Self, 1988; Laurillard, 1992), it is suggested that the model be inspectable, and preferably modifiable, by the learner. (e.g. Psozka, et al., 1988; Jones et al., 1992; Laurillard, 1992; Bruslovsky, 1994). Furthermore, HCI research in the field of adaptive interfaces suggests that fully computer-controlled adaptivity should not be used for changes in interface layout or selection of interaction style. This type of adaptation is too complex to be handled automatically (Malinowski & Schneider-Hufschmidt, 1993), and may leave the user feeling out of control (Browne, 1990).

*Adaptable Scaffolding* is implemented through giving the learner options to choose a level of scaffolding. Adaptable scaffolding fits well with learner-centered pedagogy, and offers advantages for students' development of autonomous and reflective learning and thinking skills (Reusser, 1993; Salomon, 1993). In its simplest form, adaptable scaffolding can be designed by providing the option of a beginner mode, as in the Training Wheels word processing program (Carroll & Carrithers, 1984) or KidPix's (Broderbund) option of "Small Kids Mode." Ideally, however, scaffolding should support more gradual fading (Rogoff, 1990).

A problem is that it may be hard for the learner to make fading decisions. To address this, the software can be designed to provide information and advice, and to encourage self-evaluation, helping the student to measure their progress and understanding (Nathan, Kintsch, and Young, 1990). Another issue in the design of adaptable scaffolding is that the learner needs to understand what the options are. As Fischer points out,

"demands to learn cannot originate from users if users are unaware that additional functionality exists in the system." (Fischer, 1992).

The question of how to support learners in controlling the fading of different types of scaffolding is therefore still an open one. The consensus on adaptive user interfaces appears to be that *computer-aided adaptation*, a combination of adaptive and adaptable techniques, is the appropriate direction for future research (Malinowski & Schneider-Hufschmidt, 1993). Our guided, learner-centered approach for the design of fadeable scaffolding follows along these lines.

## CONCLUDING REMARKS

A future direction for research suggested by this work is to focus more on the computer's role in helping the user make decisions about adapting the interface. For example, the system may be able to provide feedback about the appropriateness of the current level of scaffolding, by identifying and informing the learner of problems that might have been caused by fading too much too rapidly, and suggesting alternative scaffolding options.

The extent to which the system can provide this feedback depends on the domain knowledge available to the system. An open question is to identify the subset of knowledge that the system can use to provide guidance and feedback. For example, Model-It is a domain-independent tool that contains no knowledge of stream ecology or any other subject, yet it stills provide procedural scaffolding using knowledge of general and modeling-specific problem-solving strategies (e.g., planning, periodic testing). Furthermore, the learner's self-evaluation can also be taken into account—if the learner can articulate, through predictions, how he expected his model to behave, the system may then be able to help the learner in modifying the model to meet his expectations. TheoryBuilder did provide a simple debugging tool that suggested ways to find problems; more elaborate help might suggest advanced options that could be used to implement the desired behavior.

One goal of this research is to take what we've learned about scaffolding and fading design, and propose some general design guidelines. For example, we believe that an interface that presents lots of options will initially confuse many students, even if there are useful defaults. But allowing students to choose the options they want to use will help them tailor the interface in ways that make more sense to them. Through our analysis, we hope to identify more specific guidelines, and the context for their use.

The potential advantages of tools with fadeable layers of scaffolding are to provide learners with differentiable support that meets their changing needs. In this way, we allow learners to take on more of the responsibility for doing the task as they are ready to do so, and to make advanced levels of functionality available as they want access to more complex tasks.

## ACKNOWLEDGMENTS

## REFERENCES

- 1.