TOM MURRAY

Chapter 11

EON: AUTHORING TOOLS FOR CONTENT, INSTRUCTIONAL STRATEGY, STUDENT MODEL AND INTERFACE DESIGN

Abstract. This paper describes the Eon system, a suite of domain independent tools for authoring all aspects of a knowledge based tutor: the domain model, the teaching strategies, the student model, and the learning environment.

1. INTRODUCTION

This Chapter describes the Eon system, a suite of domain independent tools for authoring all aspects of a knowledge based tutor: the domain model, the teaching strategies, the student model, and the learning environment. Though development work on the prototype system ended several years ago, it remains the only example of an authoring tool for intelligent tutoring systems (ITSs) that contains a fully integrated set of authoring tools for all aspects of ITS design, though it is designed to meet the needs of pedagogy-oriented tutors more than performance-oriented tutors (as explained in Chapter 17, pedagogy-oriented tutors focus on representing pedagogical knowledge and teaching non-procedural tasks). This Chapter is a companion Chapter 15. This Chapter focuses on describing the Eon authoring tools. Chapter 15 discusses generic principles and lessons learned during the Eon project. The reader can start with either Chapter, depending on whether one finds it most perspicuous to learn about generalities or a particular examples first. In describing the authoring tools I use our Implementation of the Refrigerator Tutor as the main example. At the end of the paper I describe four other tutors built with Eon.

1.1 Preliminary Research

This work is an outgrowth of an earlier investigation of tools for the acquisition of ITS domain and teaching knowledge which culminated in a dissertation thesis and a system called KAFITS (Knowledge Acquisition Framework for ITS) (Murray 1991). The problem being addressed by this work was the gap between the ITS research community and the educational research and design community, as ITSs of increasing complexity were being developed without tools that would allow practicing educators to participate knowledgeably in the design process (Clancey & Joerger 1988). Several tools were created, including a semantic network editing tool for visualizing topics the their relationships, and an editor for creating instructional strategies in the form of Parameterized Action Networks (similar to ATNs but replacing states with actions to create a planning rather than parsing formalism).[1] Our sixteen month case study of three educators using the tools to build a 41 topic tutor for high school Statics (representing about six hours of on-line instruction) was reported in (Murray 1996, Murray & Woolf 1992a, and Murray & Woolf 1992b). Table 1 shows the productivity data for the domain expert (physics teacher), the "knowledge base managers" who did most of the data entry, and the knowledge engineer who did most of the knowledge representation work. The Table gives an indication of the relative amounts of time spent on different activities:

- An analysis of productivity indicated that it took about 100 person-hours of development time per hour of instruction (596 total hours for 6 hours of instruction, or about 15 hrs per topic);
- The above data was interpreted cautiously but we did note that it compared favorably with the 100 to 300 hours of development per hour of instruction often given for building traditional (non-intelligent) CAI;
- The domain expert invested significant time in designing and debugging the tutor, 47% of the total, while the knowledge based managers worked 40% of the total time, and the knowledge engineer only worked 13% of the total time;
- Design constituted about 15% of the total time, and implementation the other 85% (time spent on formative evaluation is not included in the data); and
- Training totalled about 15% of the total time (vs. 85% for development).

	Domain Expert		KB Managers		Knowl. Engineer		All	
	Train.	Devel.	Train.	Devel.	Train.	Devel.	Train.	Devel.
Design	22.7	36.8	0	0	22.7	3.7	46	40.5
Implem.	14	203	6	234	20	32	40	469
Totals	36.7	240	6	234	42.7	35.7	86	510
	277		240		79		596	

Table 1: Person-hours vs. Participant Role in building the Statics Tutor

We also studied the design process itself, as well as several usability and representational issues. We learned of the importance of 1) providing clear visual representations of the underlying concepts and structures of the system; 2) providing features which reduced cognitive load on working memory (by reifying information

[1]In the more recently developed Eon system, described later, a flowline paradigm was used, which maintained all of the necessary functionality of PANs.

and structure) and long term memory (by providing remindings); 3) facilitating opportunistic design by not forcing the user to make decisions in a rigid order; and 4) allowing quick movement and iteration between testing and modifying what is being built.

In sum, this work, combined with experience the author later gained using COTS (common off the shelf) authoring tools to build training systems in industry, formed the conceptual foundation for the Eon authoring tools project. The productivity data, though only from a case study of building one tutor, have given us some guidelines for estimating productivity figures on other projects, and also yielded the encouraging suggestion that intelligent tutors can be built with human resources comparable to building CAI.

1.2 Supporting Authors in a Paradigm Change

Chapter 15 begins by describing intelligent tutoring systems (ITSs) and the need for ITS authoring tools. It then describes how a shift from traditional multimedia and computer based instruction (CAI) authoring tools to ITS authoring tools involves a change from a "story board" conceptualization of instructional content to a "knowledge based" one. In the knowledge based approach tutorial content is represented as modular reusable units that are distinct form the tutoring strategies that specify how and when the content is given to the learner. The Chapter also describes how ITS authoring tools should support authoring at the "pedagogical level" as opposed to the "media level" of design by supplying basic building blocks with pedagogical meaning, such as "hint", "explanation," "summary," "prerequisite." This support authors in expressing their design ideas at an appropriately abstract and meaningful level.

As mentioned in Chapter 17, there are power/usability tradeoffs in designing authoring tools. Our goal was to give the Eon tools a usability difficulty on par with applications such as FileMaker, Excel (in advanced spreadsheet design), or AutoCAD software, rather than the lower skill level and learning curve needed for tools like PowerPoint or Email. Unlike some other ITS authoring projects (see Ainsworth et al., Halff et al., and Merrill in this Volume) we do not aim to enable the average teacher or industry trainer to create an ITS, as I think that this is an overly ambitious goal. The goal is to empower design teams who have at least one person who is trained to author ITSs.

Though there are few ITS designers, there are many CAI instructional designers. This established user population uses off the shelf software to create instructional systems. Empowering these users to build more powerful instructional systems such as ITSs requires new tools and a paradigm shift in the way many of them conceptualize instructional systems. However, this shift should be accessible, incremental, and evolutionary. Rather than starting with a laboratory-based AI tutoring system and asking how it can be generalized to produce more generic shell, as is the case in the design of some ITS authoring shells, our design base-line was commercially available and widely used authoring systems such as Authorware, Icon Author, Macromedia Director, and ToolBook. Our goal was to extend rather than replace the capabilities afforded by such systems, to preserve the level of usability and some of the authoring methods instructional designers have become familiar with, and to add additional tools, features and authoring paradigms to allow more powerful and flexible tutors to be built. For this reason, on the surface many of the Eon tools have a look and feel similar to off the shlef tools, yet allow additional levels of abstraction, modularity, and visualization necessary for producing an ITS.

Stretching off the shelf tools to the limit. Actually, many “power users” of commercial authoring tools have begun this paradigm shift. They have built layers, shells, or macros on top of the existing authoring systems that capture the repetitive or modular format of their instructional application, so they don't have to repeat the same work with every new topic or question. But these additional layers are usually large and awkward code patches that result in increased authoring efficiency, but at a loss of generality, since they are created for a specific application. The powerful features of the authoring tool are compromised, because commercial packages *allow, but do not support*, this type of abstraction. For instance Authorware has an edit-in-place feature that allows the designer to pause the tutorial, click on text or graphics, and edit right on the screen. This is extremely useful, because the screen may contain a number of text and graphic items that were brought up at different times and specified in distant portions of a large curriculum control structure. With edit-in-place the user does not have to search through this structure to find each piece. But when a shell incorporating more general procedures that access a database of questions is built on top of Authorware, the edit-in-place feature is lost. When authors pause in such an augmented system to correct, say, a spelling error in a question, the display shows a variable called “the-current-question,” rather than the actual text of the question sought. To edit the actual text they have to switch to a different program (the database program), search for it in the data base, then edit it and return to the original program to see if their change resulted in the desired effect on the screen. This is one of many ways in which the powerful features of CAI authoring tools are compromised or lost when they are coerced into a form which allows separation of subject matter and instructional strategy.

2. EON ITS AUTHORING TOOLS OVERVIEW

Next I will describe the authoring tools by showing how they were used to build a tutor that teaches how a refrigerator works.

2.1 The Refrigeration Tutor

The Refrigeration Tutor was designed to be used in a new UMass course called "Engineering, the Human Enterprise," a sort of "engineering for poets" class to give non-technical students a sense for the history, concepts, processes, and wider social aspects of engineering and design. A large section of the class was be based upon the evolution of the needs for and engineering/scientific responses to societies needs for refrigeration. The instructor for this course was the domain expert for this tutor, and participated in its conceptualisation and pilot testing. Due to logistical constraints, the tutor was incorporated into the classroom activities in only a peripheral way.

The Refrigeration Tutor teaches about the operation of the refrigerator, and the relationships between temperature, pressure, volume, energy, and phase. Its goal is to allow students to understand and grapple with questions such as:

- How is it that we can cool something off by putting energy *into* it?
- Why is the refrigerant boiling in the *colder* part of its cycle?
- Why is it that if you leave the refrigerator door open in the kitchen the room gets hotter?
- Why would a fluid such as water not work as well as Freon and other materials used as refrigerants?

The Refrigerator Tutor is relatively simple (its interactions are mainly multiple choice and point-and-click interactions) and it does not fully utilize much of Eon's functionality, but it provides a good example case for introducing the tools due to this simplicity. Since our goal is to describe the authoring tools, not the tutor itself, I will only describe enough of the tutor to show all of the tools at work.

2.2 Design Steps and Authoring Tools

We will present the tutor design process in an order which best suits the tools description, but in fact the design of the tutor happened with several stages taking place in parallel, and in a much more opportunistic fashion. The Eon tools can be used in any order that is logically consistent. Top down, bottom up, and opportunistic design approaches are possible. Eon does not walk the author through a series of design steps, nor does it have "wizards" that instruct authors in certain steps (though these would be useful). The Refrigeration Tutor was authored primarily by two members of our lab, who worked closely with the domain expert, who did not have the time to learn to use many of the tools (except the Contents Editor).

Design of a tutor can start with a concrete, bottom-up orientation, designing the screens and interface the student will use, and sketching out story boards of typical interactions. However, in our description of authoring the Refrigeration Tutor I will start top-down, building the most abstract components of the tutor, i.e. the topics and the topic network. Following that I will jump down to the concrete level and describe tools for authoring the student screens and learning environment. Next I will describe how the student model is authored. Finally I will describe how instructional strategies are authored using flowlines. Figure 1 shows the relationship between the knowledge bases in Eon (domain model, teaching model, interface specification, and student model) and the authoring tools used to build them.

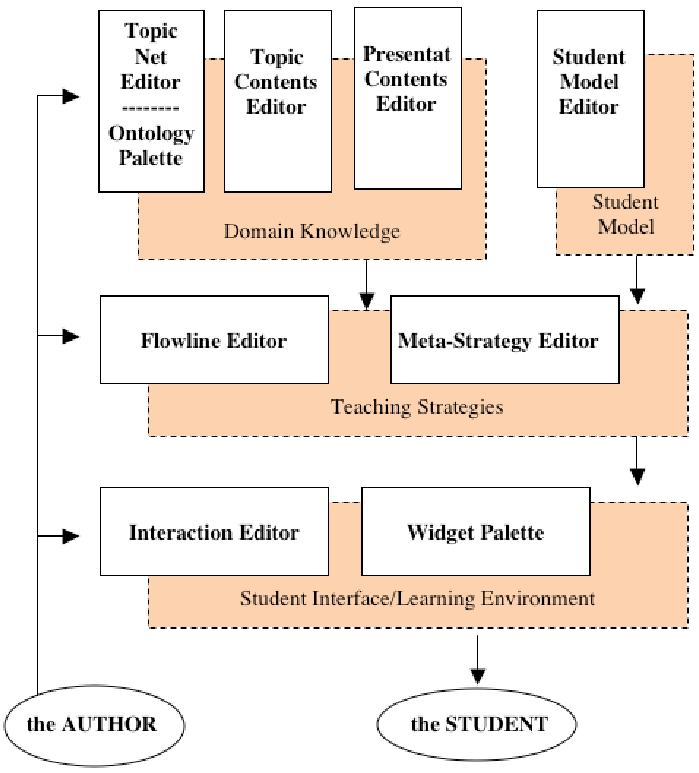
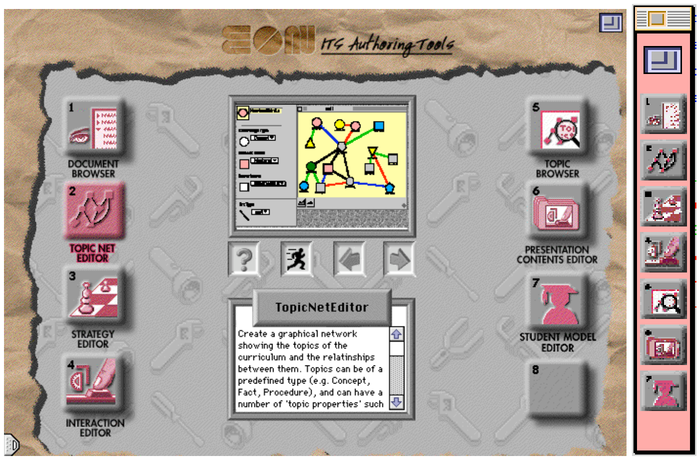


Figure 1: Eon Authoring Tools and Knowledge Bases



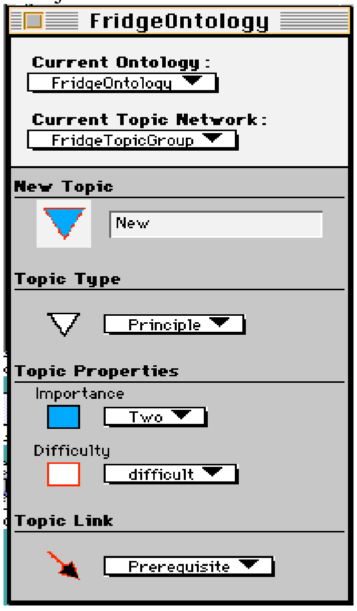
Create a graphical network showing the topics of the curriculum and the relatinships between them. Topics can be of a predefined type (e.g. Concept, Fact, Procedure), and can have a number of 'topic properties' such

Figure 2: Tool Launcher Screen and Tool Launcher Palette

When Eon is launched it shows the screen to the left in Figure 2, which allows the author to choose among the tools, and provides brief documentation on each tool. The use can toggle between this screen and a floating palette, shown to the right in the figure--both give access to all of the tools.

3. AUTHORING THE DOMAIN MODEL

A major difference between ITSs and conventional CAI systems is that ITSs contain an inspectable model of domain expertise. This expertise can be either runnable (usually a rule-based expert system) or non-runnable. The domain model also contains two types of information: performance information, which represents knowledge about the subject matter and problem solving in the domain, and pedagogical information (information relevant to learning or teaching the content). The domain model in Eon consists of a semantic network representation of the units of knowledge (called topics) that the tutor is designed to teach. Although this semantic network may represent domain expertise such as part-whole relationships and sequences of steps, the focus is on pedagogical information, i.e. links such as part-of and prerequisite which can be used in sequencing the instruction.


FridgeOntology

Current Ontology : FridgeOntology

Current Topic Network : FridgeTopicGroup

New Topic: New

Topic Type: Principle

Topic Properties

Importance: Two

Difficulty: difficult

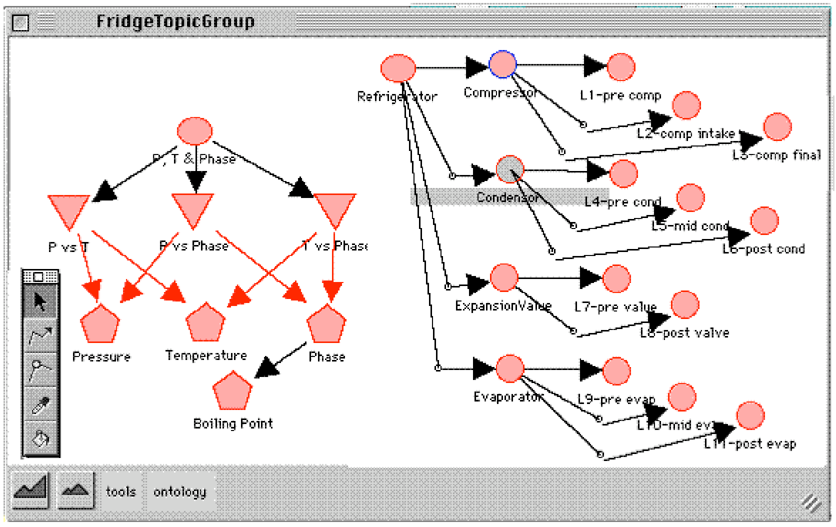
Topic Link: Prerequisite

Figure: 3 Topic Ontology Palette

Eon tutors are fairly "curriculum driven," though they can facilitate significant student control in the selection and style of the material. do not include rule-based representation of expertise. The Eon tools are not well suited for representing complex procedures or problem solving skills, yet should excel in domains where multiple teaching strategies can be written and predicated on inferred student knowledge. Tools are under development that support other classes of ITSs. Model tracing tutors and other tutors based on runnable models of domain expertise and student knowledge are extremely powerful in the situations where they apply (i.e. where the expertise can be represented at a fine gain size), and authoring tools for these are under development (Anderson & Pelletier 1991, Koedinger, personal communication). Another class of ITSs involves teaching about the functionality of equipment and associated diagnostic procedures. Authoring shells that allow the designer to build functioning simulations of equipment and diagnostic strategies have also been built (Towne & Munro 1988).

3.1 The Topic Ontology Palette

The first step in building a tutor is to map out the learning goals or topics and their relationships in the form of a topic network, using the Topic Network Editor. However, prior to this we must define the "topic ontology," which specifies the types of nodes and links allowed, and also the types of properties topics can have. In Chapter 15 I discuss the benefits of including customisable ontologies in ITS authoring tools. Primary among these benefits is the ability to adapt the authoring tool to fit the conceptual model and pedagogical assumptions of a particular ITS project. Figure 3 shows the Topic Ontology Palette which is a visual representation of the topic ontology and is used to draw topics onto the topic network. The topic ontology defines a number of "topic types" (or knowledge types), which are shown in a pop-up menu on the palette, with each topic type having its own shape. For the Refrigeration Tutor we defined these types: Fact (square), concept (pentagon), principle (triangle), physical component (circle), and unspecified (oval). The topic ontology also defines a number of "topic properties." For the Refrigeration Tutor I defined "importance" (with allowed values one to five associated with the topic node's color) and "difficulty" (with allowed values "easy, moderate, difficult" associated with the topic node's border color). The topic ontology also defines "topic link types," and for the Refrigeration Tutor we defined "prerequisite" (red), "part-of" (black), "is-a" (blue), and "context-for" (green). The topic ontology also defines "topic levels," which I will discuss later, and which are not shown graphically on the Ontology Palette or on the topic network (the are shown in the topic level editor).



Figures 4: Topic Network Editor

Eon does not yet contain an authoring tool for defining the topic ontology itself, and this must be done via a text file that is loaded in at the beginning of the authoring session. It contains lines like these:

```
DefineTopicOntology <name>
DefineTopicLinkType <name>
DefineTopicProperty <name> <allowedvalues>
DefineTopicLevel <name>
DefineTopicType <name>
    <allowedLinkTypes><allowedProperties>
    <allowedLevels>
```

Additional information is included in the text file to define how graphic properties (shapes, colors, etc.) are associated with these semantics. As is evident from the last line of pseudo-code, each topic type has a customized set of link types, properties, and topic levels.

When the ontology file is loaded it defines a topic ontology object, and the Topic Ontology Palette (also called simply the "Topic Palette") shows a visualization of the ontology.[2] To create a new topic the author selects the desired topic type and property values in the palette. The appropriate graphic attributes are shown in the "new topic" node on the palette. The author types the desired name for the new topic to the right of this new node. Finally, to instantiate this new topic the author drags the node onto the Topic Network Editor and drops it at the desired location.

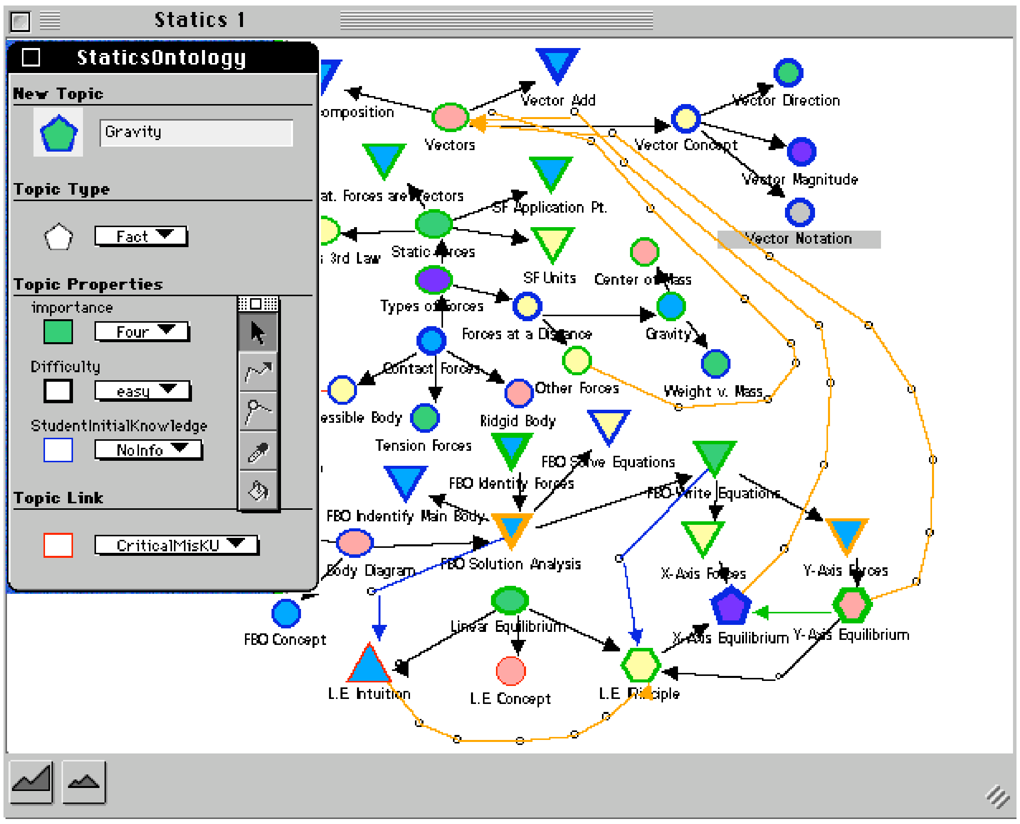
3.2 The Topic Network Editor

Once the topic ontology is defined the author can use the topic palette to draw out the topic network. Figure 3 shows the Topic Network Editor and the topic network developed by the Refrigeration Tutor domain expert. New topics are created as described above, and they can be repositioned on the screen by dragging them. To create links between the topics the author selects the link type at the bottom of the ontology palette, then clicks the "link creation tool" button on the Topic Editing Tools Palette, shown to the bottom left in the Figure, and draws a segmented line from one topic to another. In the Figure some lines are "part of" links and some lines are "prerequisite" links (they are black and red, respectively, on the computer screen).

The flow of instruction in the Refrigeration Tutor is organized around the parts of the refrigeration cycle. The student is taken on several trips around the cycle, each one having more difficult information and questions. The main components

[2]Note these other capabilities: 1) More than one ontology can be defined, thus the pop-up menu at the top of the palette the for selecting the ontology. 2) More than one "topic group" can be created, thus the pop-up menu near the top of the palette the for selecting the topic group. Each topic group is its own topic network, and each topic group uses one and only one topic ontology. 3) Each topic type is defined to specify which properties and links are allowed for it.

involved are the compressor, which compresses the gaseous refrigerant and heats it up, the condenser, which cools down the gas and turns it into liquid (fanning the heat to the outside), the expansion valve, which cools down the refrigerant, causes a drop in pressure and a partial phase change back to gas, and the evaporator, which absorbs heat from the inside of the refrigerator causing the refrigerant to boil and become completely gaseous again.


Statics 1

StaticsOntology

New Topic

Gravity

Topic Type

Fact

Topic Properties

importance Four

Difficulty easy

StudentInitialKnowledge NoInfo

Topic Link

CriticalMisKU

Figure 5: Topic Ontology Palette an Topic Network Editor for Statics Tutor

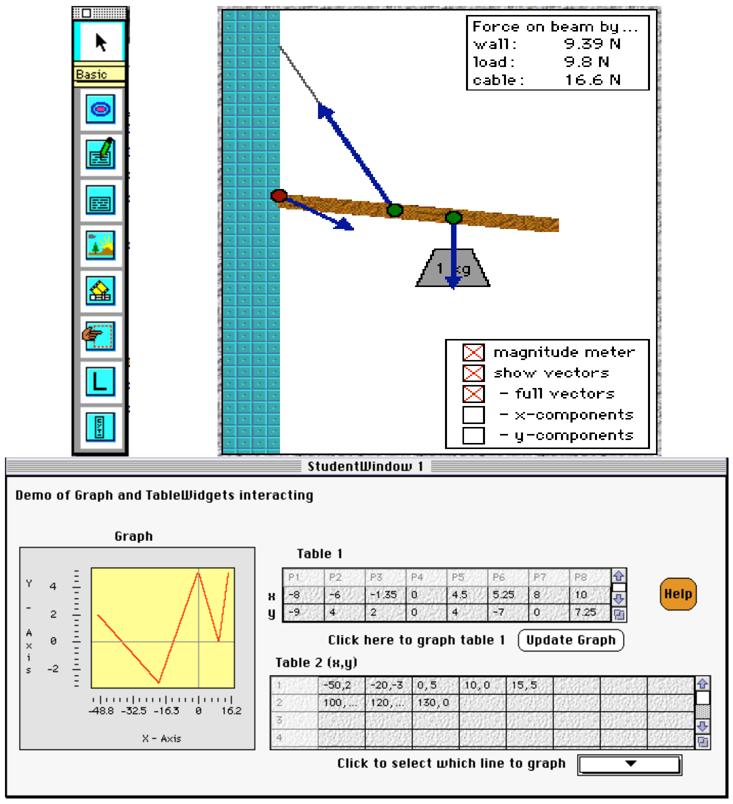
The domain expert has identified eleven important refrigerator components to focus on in this cycle, and these are shown in the topic network as sub-parts of each of the components (labeled Ln-XXX, to the far right in Figure 4). The main concepts being learned are also shown to the left in the topic network, including understanding the relationships between pressure, temperature, and phase. The student model (explained later) is used to infer the student's mastery of each of these topics. In the Refrigeration Tutor the eleven component locations are cycled through three times, with questions of higher difficulty given each cycle. The questions are linked to the concepts in the topic network as described later.

Figure 5 shows the ontology palette ("Statics Ontology") and the topic network defined for the Statics Tutor (described in Section 8). The ontology for this tutor was similar to the Refrigeration Tutor, but had an additional topic property: StudentInitialKnowledge, which was used to define the level of assumed initial knowledge level for the target student population for each topic (the vocabulary of topic types and link types also differed for the two tutors).

4. AUTHORING PRESENTATIONS AND CONTENTS

The aspect of most ITS authoring shells that is most sorely lacking, in relation to COTS CAI authoring systems, is student interface design, which for ITSs is also called learning environment design. Eon allows authors to completely customize the student interface to create highly interactive learning environments.

4.1 The Interaction Editor and Interface Extensibility



Demo of Graph and TableWidgets interacting

Table 1

	P1	P2	P3	P4	P5	P6	P7	P8
x	-8	-6	-1.35	0	4.5	5.25	8	10
y	-9	4	2	0	4	-7	0	7.25

Click here to graph table 1 [Update Graph]

Table 2 (x,y)

1	-50,2	-20,-3	0, 5	10, 0	15, 5			
2	100, ...	120, ...	130, 0					
3								
4								

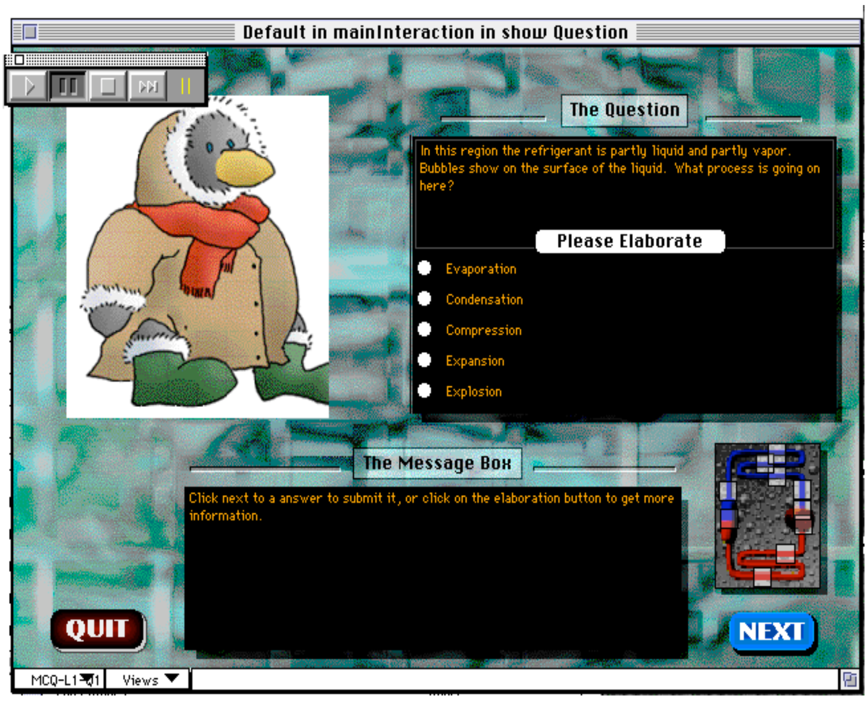
Click to select which line to graph

Figure 6 A, B, C: Widget palette, Table and Graph Widgets, Crane Boom Widget

Eon's Interaction Editor tool contains a hierarchical pallet of user interface components called widgets. There are simple widgets such as buttons, text, pictures, sliders, movies, and hot-spots, and more complex widgets such as multiple-choice dialogs, tables, and graphs. There are a total of 26 widgets in six categories: Basic, Controller, Geometrical, Complex, Special, and Custom. Widgets are selected for drawing onto the interaction screen using the Widget palette. Figure 6A shows the Basic widget set. The author clicks where "Basic" appears to see a a pop-up menu of the other widget categories. Figure 6B shows two tables and a graph, widgets from the Complex category.

The Widget Palette is extensible via the "Custom" category. Arbitrarily complex widgets can be programmed outside of Eon, and “dropped in” as needed for particular domains. These custom widgets can be device simulations or whole learning environments. For example, our Statics Tutor has a “crane boom” widget (see Figure 6C) which lets students manipulate positions of objects and cables and observe the resulting static forces. In order for custom widgets to inter-operate with the rest of the Eon tools, they must adhere to a simple protocol which involves specifying the “parameters” used to set a widget's properties, and the student “events” that the widget recognizes. The events can be simple as “button-pushed,” or require some processing as in a multiple choice widget's “correct answer” event, and can be arbitrarily sophisticated, as in a “student has moved the load past the tension limit of the cable” event in the crane boom widget.

Widgets are selected from the Widget Palette and drawn onto the Interaction Editor (Figure 7). Figure 7 shows the screen design for the Refrigeration Tutor's multiple choice interactions. It includes a picture, text areas for the question and answer choices, a text area for hints and an "elaboration," an icon-ized map of the refrigeration cycle (lower right) onto which a "you are here" indicator is placed. This is a reusable template screen which was built using text, graphic, multiple choice, and button widgets from the widget palette. All multiple choice questions are shown using this template screen, by filling in the question-specific information each time it is shown to the student. This is a significant and unique feature in the Eon tools. The screen design shown in the Figure is not for a particular multiple choice question, but for a large set of them. As explained later, a tool called the Contents Editor is used to manage the data for the many instances that get "plugged into" the template.

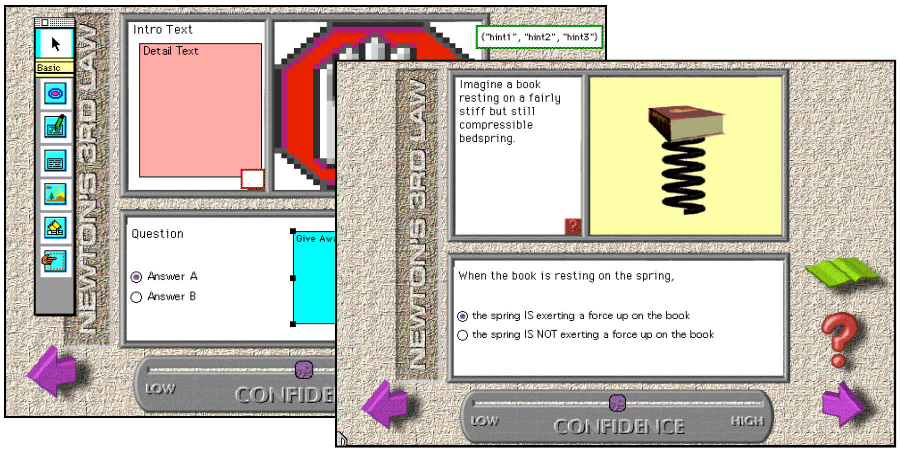


In this region the refrigerant is partly liquid and partly vapor. Bubbles show on the surface of the liquid. What process is going on here?

- Evaporation
- Condensation
- Compression
- Expansion
- Explosion

Click next to a answer to submit it, or click on the elaboration button to get more information.

Figure 7: Interaction Editor (Refrigerator Tutor)



Imagine a book resting on a fairly stiff but still compressible bedspring.

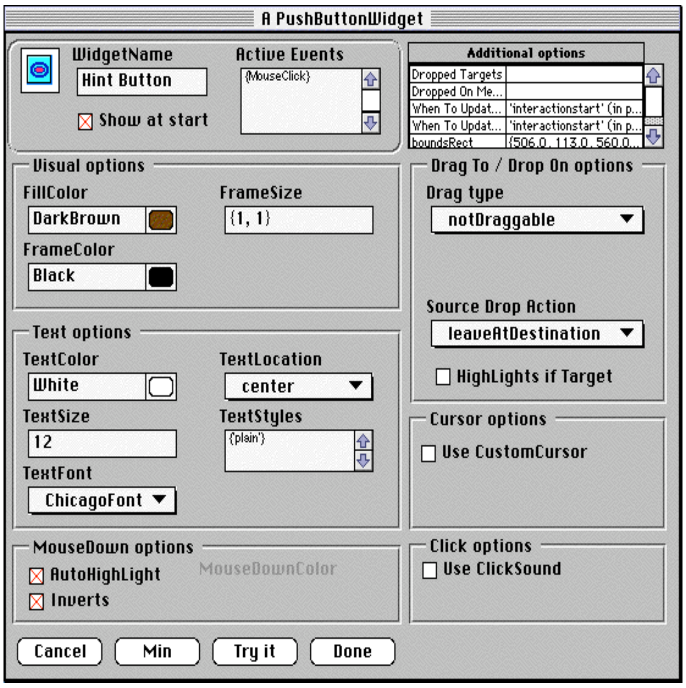
When the book is resting on the spring,

- the spring IS exerting a force up on the book
- the spring IS NOT exerting a force up on the book

Figure 8: Interaction Editor Template and Final Screen (BA Tutor)

To the left of Figure 8 is a template interaction screen for the Bridging Analogies tutor (described in Section 8). To the right is the template filled in with data from one of its Contents objects, as the student would see it.

Eon distinguishes two types of widget attributes: "options" and "properties." Options comprise most of the widgets attributes. They are set using the widget's options dialog and will usually remain as initially set during the tutorial. Figure 9 shows the options dialog for a push-button widget, and gives an indication of the complexity of Eon's widgets. Widget *properties* are the small set of the most important widget attributes that deal with student input or instructional content, for example, the text of a text widget, and the picture in a graphic widget. If the author is using template screen a widget's properties are likely to change during the course of the tutorial. The widget properties of the multiple choice widget are: the question text, the answer choices text, the correct answer index, and the answer selected by the student (which is set at run time by the student, not the author). Both options *and* properties can be manipulated dynamically at run time (using a scripting language), allowing for dynamic screens with content generated on the fly. Properties, however, have additional flexibility as described below.


A PushButtonWidget

WidgetName: Hint Button
Active Events: {MouseClick}
[✓ Show at start]

Additional options	
Dropped Targets	
Dropped On Me...	
When To Updat...	'interactionstart' (in p...
When To Updat...	'interactionstart' (in p...
boundsRect	{506.0, 113.0, 560.0...

Visual options
FillColor: DarkBrown
FrameSize: {1, 1}
FrameColor: Black

Drag To / Drop On options
Drag type: notDraggable
Source Drop Action: leaveAtDestination
HighLights if Target

Text options
TextColor: White
TextLocation: center
TextSize: 12
TextStyles: {'plain'}
TextFont: ChicagoFont

Cursor options
Use CustomCursor

MouseDown options
[✓ AutoHighLight]
[✓ Inverts]
MouseDownColor

Click options
Use ClickSound

Figure 9: Options Dialog for a Push-button Widget.

A large amount of effort was put into Eon's student interface builder, in order to allow authors complete flexibility in the design of the interface. As well as the template feature, making it easy for authors to create repetitive content, all widget properties can be manipulated via scripts, allowing the screens to be composed and modified dynamically during the tutorial. Visual and semantic properties of widgets can be made to depend on each other, allowing simple simulations to be built.[3]

Still, Eon is not as facile at authoring complex learning environments and simulations as tools built specifically for this purpose. For example, the RIDES system (Munro in this Volume) allows widgets such as simulated meters, levers, faucets, and motors to be connected by wires or pipes, and represents the interactions between these components in such a way that students can inspect how the device operates. RIDES, like other special purpose authoring tools built to date, has only limited abilities to represent curriculum, content abstractions, or multiple teaching strategies

4.2 Content Generation and Re-use

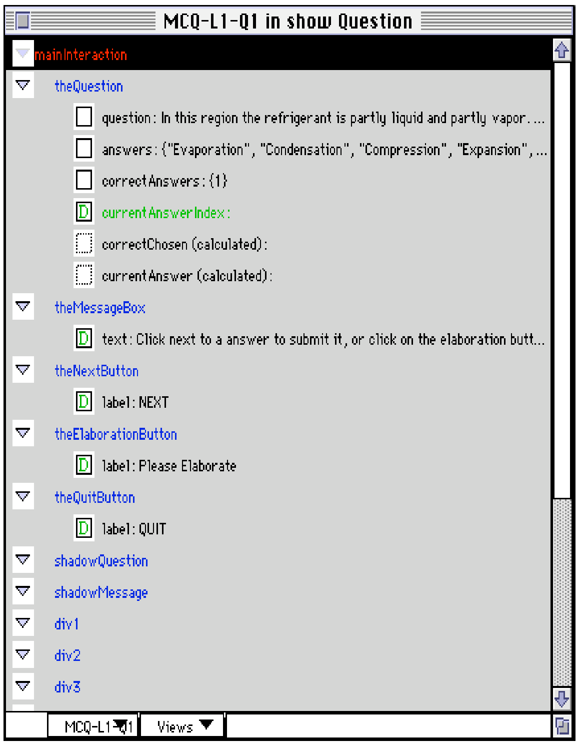
Widget properties, such as text and graphics, can be set in several ways:

1. **Single value**. E.G. a graphic widget always has the same picture.
2. **Calculated value**. The value can be attached to a script. E.G. each time the picture is shown in the tutorial, the script is evaluated to produce a new value (e.g. a pointer to a new picture).
3. **Template**. Screens containing widgets can be specified as "templates" which get reused. In such cases the properties of the widgets are stored in "Contents" objects (described below).

Using templates the author can create re-usable screen formats, and then create multiple "Content" objects to fill in a template dynamically. Figure 7 shows one of the template-based interaction screens authored for the Refrigeration Tutor. The Figure shows the default contents of this template. About fifty Contents objects have been created to fill in this template, each with its own question, picture, explanation, and hints. As the author adds widgets to the screen a data base entry form called the Contents Editor is updated to show the properties of all of the widgets. Figure 10 shows the Contents Editor for the multiple choice template screen, with the contents object MCQ-L1-Q1 shown. The Contents Editor shows all of the properties of all of the widgets on the student screen. Some properties are designated "D" to use the default value (a "C" would indicated that the value is computed dynamically, i.e. attached to an author-defined script). The author can edit widget properties directly from the Interaction Editor (which is WYSIWYG), or she can edit the properties using the Contents Editor (which is like a database form).

[3]However, the system gets slow and cumbersome for simulations with many interacting components, and in such cases it is better to program a separate custom widget, as we did for the Crane Boom simulation

In the bottom left of the Contents Editor and the Interaction Editor is a pop-up menu listing all of the contents that have been created for this template (MCQ-L1-Q1 is one of them), an item "default" to view the default contents, and an item "New" to create a new contents item. The "Views" button to the right of this button is for easy navigation between the three tools which constitute three views of this domain content: the Interaction Editor, the Contents Editor, and the Flowline Editor (see below).



MCQ-L1-Q1 in show Question

mainInteraction
theQuestion
question: In this region the refrigerant is partly liquid and partly vapor. ...
answers: {"Evaporation", "Condensation", "Compression", "Expansion", ...
correctAnswers: {1}
currentAnswerIndex:
correctChosen (calculated):
currentAnswer (calculated):
theMessageBox
text: Click next to a answer to submit it, or click on the elaboration butt...
theNextButton
label: NEXT
theElaborationButton
label: Please Elaborate
theQuitButton
label: QUIT
shadowQuestion
shadowMessage
div1
div2
div3

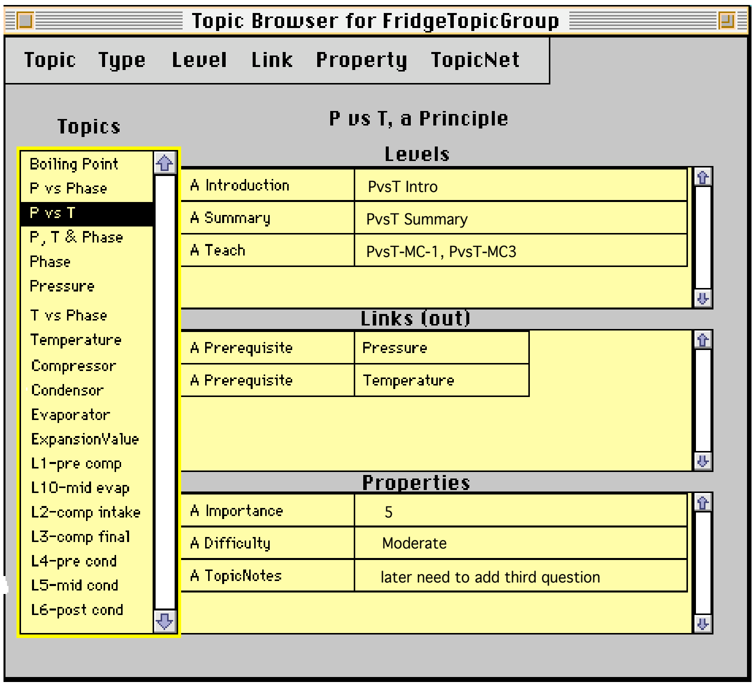
Figure 10: Contents Editor showing widget properties

To summarize, content reuse is facilitated by allowing an author to create interactive screens as templates. The widgets are drawn once using the Interaction Editor, and then numerous Contents are created using the Contents Editor. Additional flexibility is available by attaching widget properties to scripts (as opposed to canned material) so that interactive screens can be generated on the fly.

4.3 Connecting Topics to Contents

Thus far I have shown how the author works at the abstract level of the curriculum, in mapping out the topic network, and at the concrete level, designing the interactive screens and (if they are template screens) filling in data base forms to define the Content objects that fill in the screens with domain content (the questions, hints, explanations, etc.). In this Section I will describe how the abstract level is linked to the concrete level. Basically, Content objects are assigned to the topics, but it is a bit more complex than that, and to explain how, first I must add one detail about topics left out of the earlier discussion: Topic Levels.

In Eon topics need not refer to a single monolithic entity, but have an extra level of internal structure called Topic Levels (see Chapter 15 for a discussion motivating their inclusion). Each topic has one or more Topic Levels which can specify different aspects or uses for the topic, for instance: introduction, summary, teach, test, beginning level, difficult level, etc. In the Refrigeration tutor the topics levels "Introduce," "Teach," and "Summary" were defined. A list of Content objects are assigned to each Topic Level for each topic. Thus when we want to "introduce a topic" we give the Contents in its Introduction level, and when we want to teach a beginning level of a topic we give the Contents listed in its Beginning level. The list of Contents in a Topic Level is usually thought of as a sequence of contents to present, but it can also be a *set* of applicable Content objects, and selection and sequencing of these can be left to the teaching strategy (for example choosing randomly from the set). The topic levels used in a tutorial are defined in the Topic Ontology object, described previously.



Boiling Point
P vs Phase
P vs T
P, T & Phase
Phase
Pressure
T vs Phase
Temperature
Compressor
Condensor
Evaporator
ExpansionValue
L1-pre comp
L10-mid evap
L2-comp intake
L3-comp final
L4-pre cond
L5-mid cond
L6-post cond

A Introduction	PvsT Intro
A Summary	PvsT Summary
A Teach	PvsT-MC-1, PvsT-MC3

A Prerequisite	Pressure
A Prerequisite	Temperature

A Importance	5
A Difficulty	Moderate
A TopicNotes	later need to add third question

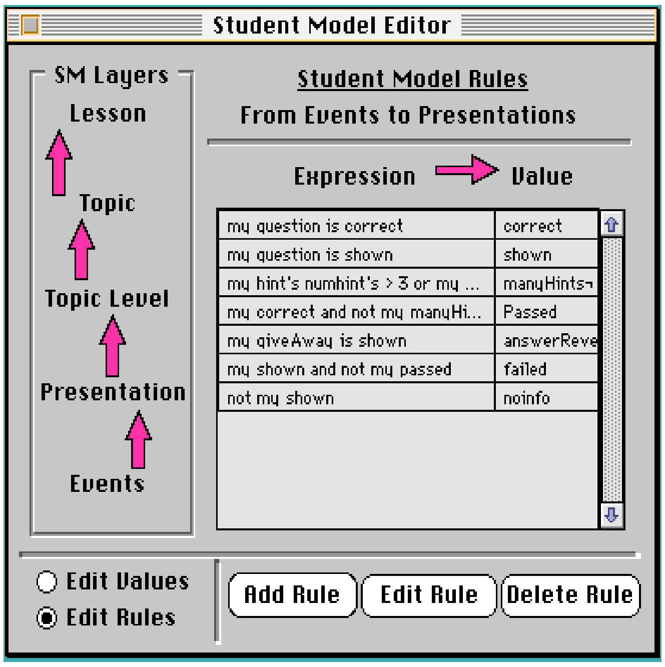
Figure 11: Topic Contents Browser

Thus far I have described the curriculum contents as it is stored in modular, declarative units: as Topic and Content objects. Later I will discuss teaching strategies, which are used to determine how this declarative information is used: i.e. how the topics are sequenced, when each topic level is given, how the Content objects are sequenced, when the interactive screens are shown, and how student behavior is responded to. For now, I will only remind the reader that all of the domain information discussed thus far is strategy-independent. It can be sequenced in arbitrary ways, as specified by teaching strategies.

The Topic Contents Browser. The Topic Contents Browser is an authoring tool that shows another view of the topic space, a tabular one as opposed to the graphic view given in the Topic Net Editor. Figure 11 shows the Topic Contents browser for the topics in the Refrigeration Tutor. The Topic Contents Browser shows a list of all of the topics, and the links and Properties of the selected topic. This is another view of the information available visually in the Topic Net Editor. In addition (unlike the Topic Network Editor) it shows the Topic Levels and the Content objects associated with each level.[4] The author used the topic contents browser to edit the values of the levels, links, and properties, either by typing into the form (topic Importance in the Figure) or by choosing from a list (selecting "PvsT Intro" from among the defined Contents for the Introduction level).

[4]The terms "principle," "introduction," "summary," "teach," "prerequisite," "importance," and "difficulty," seen in the Topic Contents Browser, and other terms which define the conceptual vocabulary for describing content and pedagogy in the Refrigerator Tutor, are defined in the Topic Ontology object. This illustrates how the conceptual vocabulary is customizable for each tutor.

5. AUTHORING THE STUDENT MODEL



Expression	Value
my question is correct	correct
my question is shown	shown
my hint's numhint's > 3 or my ...	manyHints¬
my correct and not my manyHi...	Passed
my giveAway is shown	answerReve
my shown and not my passed	failed
not my shown	noinfo

( ) Edit Values [✓ Edit Rules]

Figure 12: Student Model Editor

The student model is the component of the system that keeps track of student behaviors and makes inferences about what the student knows. Eon is the only ITS authoring tool that include the ability to customize how the student modeller works. Eon uses a variation of an "overlay student model" (Wenger 1987) in that mastery values are calculated to correspond with each topic. The Eon student model can also be used as a "bug library," since topic types for "mis-knowledge" such as misconceptions and buggy procedures can be defined to keep track of known classes of common deficiencies.[5] I call our student model a "Layered Overlay Student Model" because values are inferred at several "decision layers," as shown in Figure 12. Most overlay student models assign values to topics only. In Eon values are assigned to objects at several decision layers: Lesson, Topic, Topic Level, Presentation Contents, and Events.[6] Objects at each layer are composed of objects

[5]In our Statics tutor we defined a topic type called "Misconception," and a link type called "critical misconception" which was used to associate physics concepts to common related misconceptions, so that these common misconceptions could be diagnosed before the concept was taught

[6]Lessons are used to specify the top level teaching procedure and the goal topic or topics for a tutorial session. Lesson objects were not used in the Refrigeration Tutor, and they are defined in text files as no authoring tool for them has been built yet.

at the next lower layer, for example, running a lesson will invoke the teaching of a number of topics, teaching a topic will run a number of its topic levels, and each topic level consist of a number of Contents (representing student interactions or blocks of information). Within a Content, for example a multiple choice question, a number of low level student and tutor events will occur, such a selecting an answer or asking for help (a student event), or giving a hint (a tutor event). As shown in the Student Model Editor (Figure 12) objects at each level can have a value. The value of objects at any level is determined by the Student Model rules written for that level. These rules specify how the value of an object depends on the values of the objects at the next lower level.

Figure 12 shows the rules for mapping from the Events level to the Presentation Contents level for the Statics Tutor. For the Refrigeration tutor, some example rules are:

```
Topic Level to Topic:
    my Teach Level is KNOWN
          OR all of my Parts are KNOWN
    Topic value ==> KNOWN
    my Teach Level is SHOWN
          OR my Introduce Level is SHOWN
          OR my Summary Level is SHOWN
    Topic value ==> SHOWN

Presentation to Topic Levels:
    greater than 80% of my Presentation Contents are
    CORRECT
        TL value ==> KNOWN
    any of my Presentation Contents is SHOWN
     TL value ==> SHOWN
```

The student model is used to make the tutorial adaptive to the student's inferred state. This is accomplished by referring to student model values in decisions in teaching strategies (and meta strategies), for example, decision branches predicated upon whether a topic is “mastered” or whether a Contents has been already “shown”. The Refrigeration Tutor uses the student model to ask fewer easy questions if a topic is near mastery, and to give more hints if the topic is far from mastered.

The vocabulary of terms used to define the student model values (e.g. "known," "mastered," "suspected misconception") is customizable for each tutorial using the student model editor. We have found the current Student Model Editor to be too restrictive however, since values and rules can only be defined on a per-decision-level basis, i.e. every item in a decision level has the same rules and the same set of allowed values. We are working on an extension to this system to allow rules to be assigned to groups of objects. We would like some topics to use different student modeling rules than others. For example, in the Refrigeration tutor we would like the topics representing the important concepts (to the left in Figure 4) to use a different rule set than the topics representing the components and locations within the refrigerator (to the right in Figure 4).

6. AUTHORING THE TEACHING MODEL

To represent teaching strategies we use a flowline paradigm, a graphically portrayed procedural representation of strategic knowledge that explicitly shows structural and contextual control information. Our Strategy Editor (also called the Flowline Editor) has a look and feel similar to commercially available authoring tools such as Authorware and Icon Author. Eon Flowlines, like procedures in programming languages, can have input parameters, local variables, and can return values.[7] All variable referencing and naming is facilitated by menus and drag and drop tools, so the author does not have to memorize or type in these terms (as one does in scripting languages).

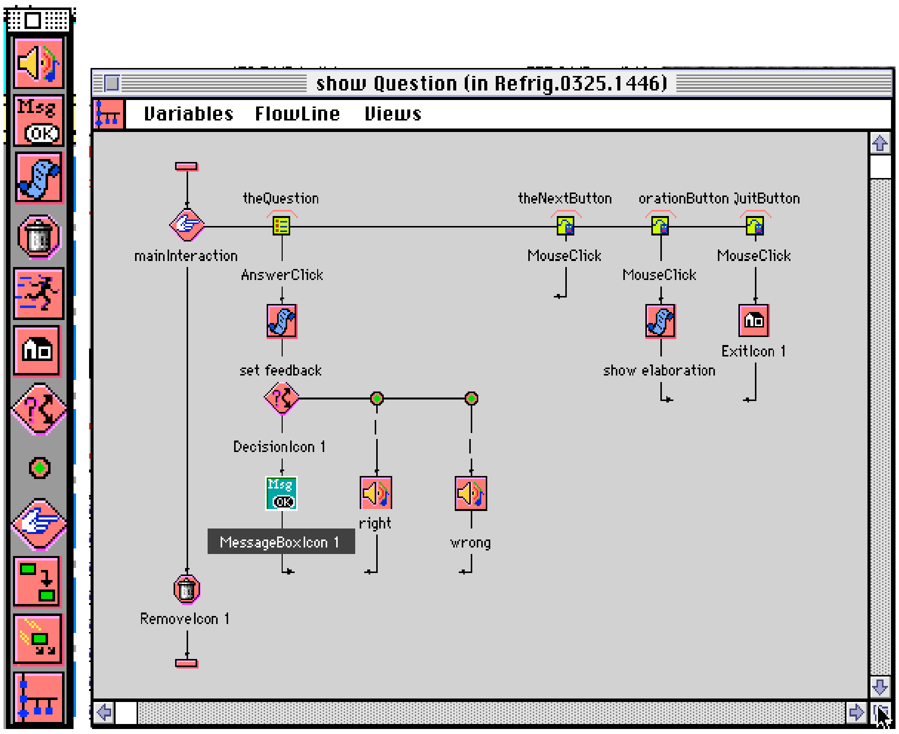


Figure 13: Icon Palette and Flowline (Refrigeration Tutor)

[7]Though of similar surface appearance, the Eon flowline system is actually much more powerful than those seen in commercial tools. For example, commercially available tools do not support input parameters or return variables. They are not "real" procedures that can be called recursively.

Figure 13 shows the Flowline Editor and the Icon palette used to drag and drop flowline icons onto the flowline. Flow of control travels down the flowline icons, and branches to the right for some icon types. The Run Icon invokes another flowline and the Return Icon is used to exit a flowline prematurely, or to specify a returned value. The icons in the palette are, from top to bottom:

- Sound-- to play any type of sound resource.
- Message Box -- quick way to show text to the student (without an interaction screen).
- Script -- arbitrary scripts which can refer to flowline local variables.
- Erase/Remove -- hide or show widgets.
- Run -- invoke another flowline, passing input parameters if needed.
- Home -- return from a flowline, returning a value if needed.
- Decision -- repeat loops, branching, If-Thens, etc.
- Branch -- individual decision branches.
- Interaction -- bring up an interaction screen and respond to student-generated widget events.
- Copy Data-- transfer data from one location to another (e.g. from a Content object to a local variable).
- Animate -- animate widgets or graphics along an arbitrary path.
- Composite -- a call to a sub-flowline.

The Flowline shown in the Figure 13 brings up the main multiple choice screen interaction, then branches are defined for giving feedback on student answer selections, for pushing the Next or Quit buttons, and for selecting the Elaborate button.

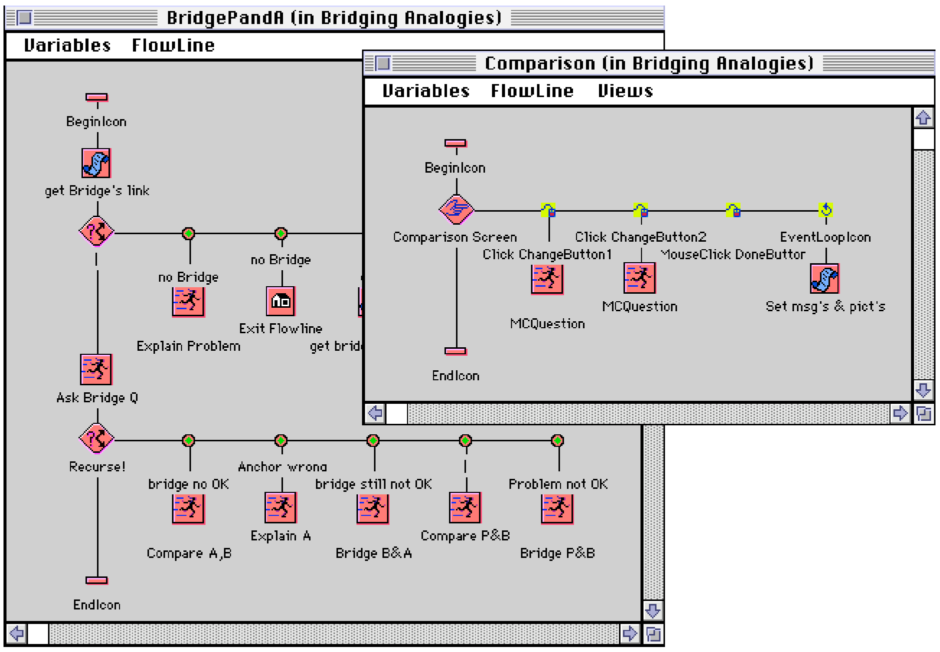


Figure 14: Flowlines for BA Tutor

Figure 14 shows two Flowlines from the Bridging Analogies tutor. The "BridgePandA" flowline is a "doubly recursive" procedure that calls itself twice at the second decision icon. The Comparison Flowline is one of the sub-flowlines called in the first Flowline. The "Ask Bridge Q" icon in the BridgePandA flowline calls a flowline that brings up the screen in Figure 8, and the Comparison Screen icon in the Comparison Flowline brings up the screen in Figure 17.

A variety of representational formalisms have been used for control and strategic knowledge in ITS shells and authoring tools. Some employ relatively sophisticated AI techniques such goal-based planning (Russell et al. 1988), black board architectures (W. Murray 1990), agents (Cheikes 1995), task decomposition (Van Marcke 1992), and production rules (Anderson & Pelletier 1991, Major & Reichgelt 1992). No framework or visual editor has yet been devised for any of these formalism which lowers the complexity level sufficiently for our intended user audience. These formalisms are highly modular, but control information elicited from human experts often has clearly defined structure (Gruber 1987), and high modularity can hide the structure of strategic knowledge, obfuscate the context of strategy decisions, and make strategy design unwieldy (Lesser 1984).

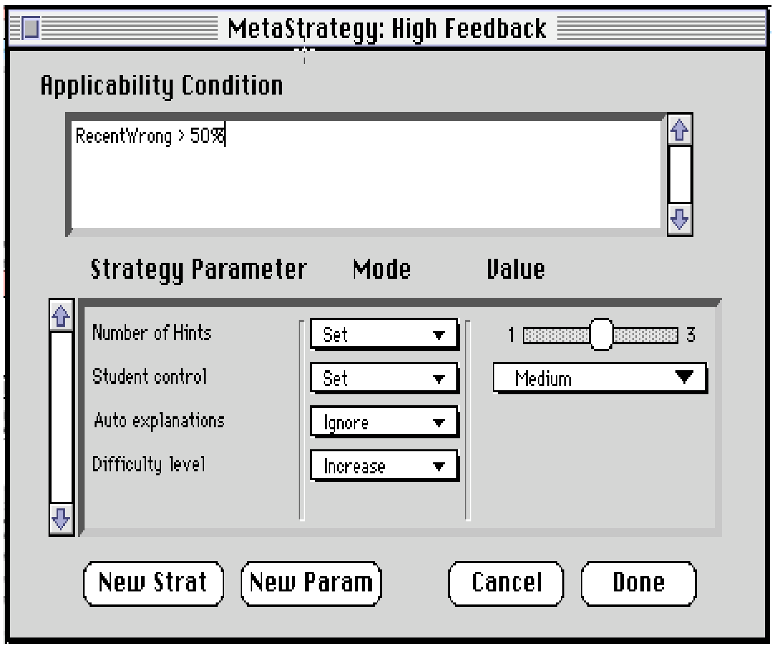
Strategy representation in Eon is based on a flowline paradigm for visual authoring, which has proven to be highly understandable and usable. It is not as powerful as the more AI-intensive methods mentioned above, because it does not allow a tutor to search a large space of potential instructional actions, or to reason about what it could do further along in the session.

Meta-strategies

Ohlsson (1987, pg. 220) points out that "in order to provide adaptive instruction, a tutor must have a wide range of instructional actions to choose from." Human tutors have more than one teaching method or style available to them, and likewise, intelligent computer tutors should be able to change teaching style dynamically depending on student characteristics. Spensley et al. (1990) describe a shell which allows meta-strategies to choose among pre-defined general strategies, including cognitive apprenticeship, successive refinement, discovery learning, abstraction, practice, and Socratic diagnosis. The strategies themselves are fixed however, and fine grained decisions can not be modified. REDEEM (see Ainsworth et al in this volume) is a highly usable authoring tool that allows teachers to set a number of teaching strategy parameters to customize and select applicability conditions for teaching actions. In this system some flexibility is traded for usability, since the underlying instructional strategies are pre-defined (though parameterized). Van Marcke's (1992) GTE system uses multiple alternative rule sets to carry out the actions of a given tutorial goal. This system is more flexible but difficult to author.

As a method for representing multiple strategies in Eon we considered implementing multiple flowlines with the same purpose (e.g. multiple "Give a Hint" flowlines) as in the GTE system, but this seemed too confusing for users. Eon uses a parameterized approach similar to REDEEM, but is more flexible since the strategies can be built from scratch.

Eon strategy parameters are like global variables that can be used in decision points in flowlines. Authors can define a number of strategy parameters, for example, “degree of hinting,” “degree of interruption,” “preference for general vs. specific information,” and “amount of information." A parameter called "number of hints" can be defined and used in a flowline to specify whether one, two, or three hints are given. A "student control" parameter can be defined and used in a flowline to decide whether to allow students to choose to skip topics. Strategy parameters values are set through meta-strategies, and are defined using the MetaStrategy Editor (Figure 15). Each meta-strategy includes a set of strategy parameters, and a setting for each of those parameters. Also, each meta-strategy has an "applicability condition" that defines when the MetaStrategy is triggered. For example, in the "High Feedback" MetaStrategy shown in Figure 15, the applicability condition is "Recent Wrong" (a variable in the student model) is greater than 50%. When this is true, the High Feedback strategy is triggered. When it is triggered it sets the values of several strategy parameters: number of hints, student control, auto-explanations, and difficulty level. Since these strategy parameters are used in branch and decision icons in flowlines, the behavior of the tutor will be changed accordingly.


MetaStrategy: High Feedback

Applicability Condition

RecentWrong > 50%

Strategy Parameter	Mode	Value
Number of Hints	Set	1 … 3
Student control	Set	Medium
Auto explanations	Ignore	
Difficulty level	Increase	

Figure 15: MetaStrategy Editor

Though the Meta-Strategy Editor is functional, it was not used in the Refrigeration Tutor before.

7. OTHER TOOLS, MISCELLANEOUS CAPABILITIES, AND IMPLEMENTATION DETAILS

The authoring tools have a WYSIWYG pause-and-edit feature, which allows the author to test run a tutorial, pause it at any point, and easily access the Interaction Editor, Contents Editor, or Flowline Editor, to modify the data bases of teaching strategies. Eon has a number of other tools not described above, including tracing and debugging tools, and a Document Browser which gives a hierarchical view of every object in a tutorial document. The Document Browser is shown in Figure 16. The document browser is an alternate way to view, access, and edit any object or object attribute in the system. Double clicking on an object in the document browser brings up the appropriate editor (e.g. the Flowline Editor if a flowline name is double-clicked).

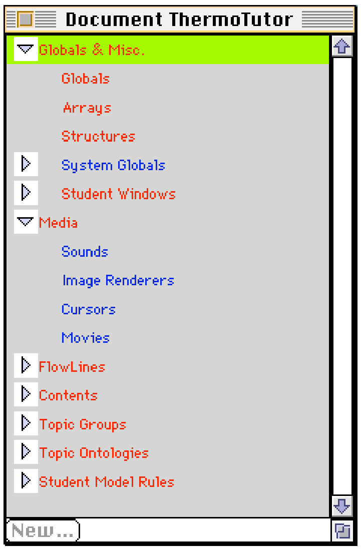


Figure 16: Document Browser

Eon is implemented in a high level programming language called SK8, developed by Apple Computer's Advanced Technology Group (see http://SK8.research.apple.com). SK8 is a very powerful high level language that was designed not only to build applications, but to build custom application builders. SK8 has the programming power of LISP, a graphics engine comparable to Photoshop, a visual programming paradigm analogous to Visual Basic, and a number of artificial intelligence information processing features. All of this allowed us to build the Eon prototype in a fraction (we estimate one fifth to one quarter) of the time it would have taken to build Eon in C++ (or, alternatively, given fixed resources we able to add four to five tiems as much functionality). Unfortunately SK8 development and support was discontinued while the language was still in an alpha stage and SK8, though a powerful programming environment, was still not optimised and was slow, not bug-free, and took up an inordinate amount of run time memory.[8] As a result, the Eon authoring tools are relatively slow, run only on Quadra-generation Macintoshes in an obsolete version of the Mac OS, and experience occasional crashes (all of these problems were to be fixed at the SK8 language level, had it continued as an Apple product). As a consequence the Eon tools are alpha prototypes, useful for demonstrating ITS authoring and capable of producing ITSs given certain restrictive hardware limitations on the runtime environment.[9] Despite these hindsight problems with using the SK8 development environment, we have been able to build a very large and complex proof-of-concept system, with a high degree of interactivity and usability, in a fraction of the time it

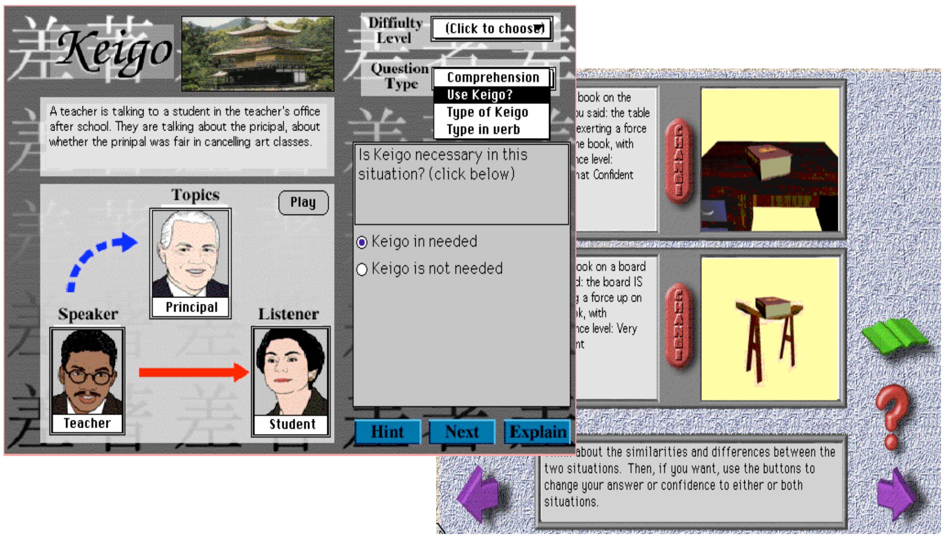
[8] SK8 source code it is now available in the public domain.

[9] The Eon software is not publicly available, nor would the code run on modern computers.

would have taken to build in a more traditional programming environment (we estimate 3 person-years vs. 10 to 12). Thus, from a research perspective we made the right choice in using SK8, but in order for the Eon to be a viable ITS development platform, the system would need to be re-implemented in another environment (C++ or Java, for example) at considerable cost. Though development on the Eon system stopped several years ago, we no of no other ITS development environment to date that matches the level of functionality, generality, and usability that it provided.

8. OTHER TUTORS BUILT WITH EON

Eon was used to build four other tutors, which I describe briefly below. The tutors are all prototypes that were not used in actual classroom situations. While most were designed to demonstrate and test specific Eon capabilities, the Keigo, Statics, and Refrigerator Tutors were designed to address real educational needs. Due to the above mentioned problems with SK8 these systems were later redesigned (in some cases removing some of the more intelligent features) in another language (Macromedia Director) so that they could be used for their intended purposes (the descriptions given below apply only to the Eon-built tutors).



A teacher is talking to a student in the teacher's office after school. They are talking about the pricipal, about whether the prinipal was fair in cancelling art classes.

Is Keigo necessary in this situation? (click below)

[✓ Keigo in needed]
Keigo is not needed

book on the
ou said: the table
exerting a force
he book, with
nce level:
hat Confident

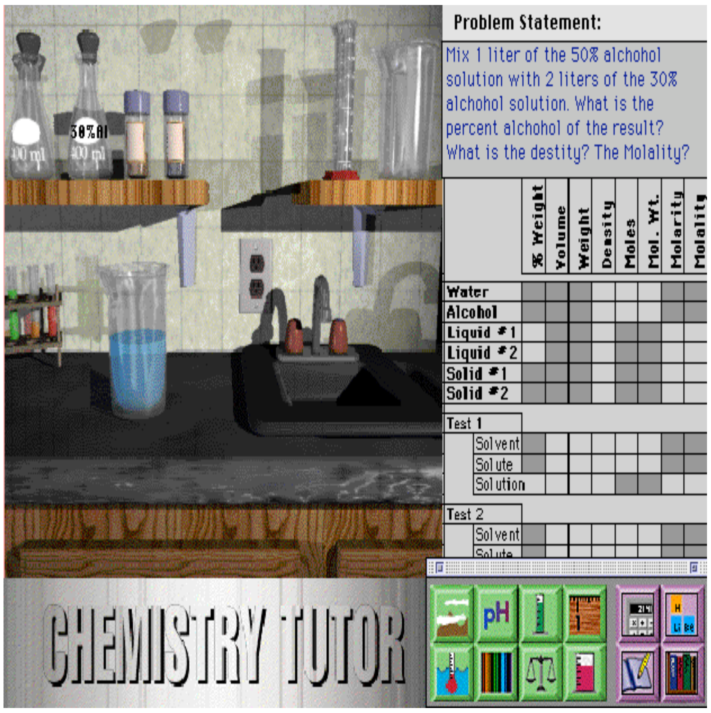
ook on a board
d: the board IS
g a force up on
ok, with
nce level: Very
nt

about the similarities and differences between the two situations. Then, if you want, use the buttons to change your answer or confidence to either or both situations.

Figure 17 & 18: Keigo Tutor & Bridging Analogies Tutor

The **Keigo Tutor** (Figure 17) teaches a part of Japanese language called “honorifics,” dealing with the complicated rules used to determine verb conjugations which appropriately honor the listener and topic of a conversation. For this tutor Eon is interfaced with a rule based expert system encoding rules about how to map from surface level features of the conversational situation to linguistically relevant properties of the situation (the expert system was written in the SK8 language). Students are presented with a variety of situations involving people of varying levels of status and formality, and are asked what types of verb conjugations are needed. Feedback is given based on which rules are applicable. This tutor has been used and evaluated in a Japanese Language class (Miyama working paper).

The **Bridging Analogies Tutor** (Figure 18) incorporates a Socratic teaching strategy developed and tested by cognitive scientists to remediate common persistent misconceptions in science (Murray et al. 1990). This tutor demonstrates the implementation of a recursive teaching strategy which was researched in human one-on-one tutoring and classroom teaching. The student is presented with a sequence of analogous situations in an attempt to bridge the conceptual distance between their current understanding of the phenomena ("does a flat surface push back up on something that sits on that surface?" in this case) and the correct physics interpretation. Occasionally students are presented with screens that encourage them to compare and contrast previous answers and they are given the option of changing a previous answer in light of new conceptual connections.



Problem Statement:

Mix 1 liter of the 50% alchohol solution with 2 liters of the 30% alchohol solution. What is the percent alchohol of the result? What is the destity? The Molality?

	% Weight	Volume	Weight	Density	Moles	Mol. Wt.	Molarity	Molality
Water	▨	▨	▨				▨	▨
Alcohol	▨	▨	▨				▨	▨
Liquid #1		▨	▨		▨	▨		
Liquid #2		▨	▨		▨	▨		
Solid #1	▨	▨	▨		▨	▨		
Solid #2	▨	▨	▨		▨	▨		
Test 1								
Solvent	▨						▨	▨
Solute	▨						▨	▨
Solution					▨	▨		
Test 2								
Solvent	▨						▨	▨
Solute	▨						▨	▨

Figure 19: Chemistry Workbench

The **Chemistry Workbench** (Figure 19) is the tutor which demonstrates the most open-ended learning-by-doing environment of the tutors built with Eon. Students can mix chemicals from the shelf by dragging and dropping them on beakers. Color and precipitates of the resulting solutions are shown visually in the beakers, and the instruments in the tool palette at the bottom right can be used to measure the volume, pH, etc. of the results. Unlike the Cane Boom simulation widget mentioned above (see Figure 6C), all of the graphical interactions in the Chemistry Tutor were created using Eon's built-in widgets. The goal of the tutorial is to learn about solvency and chemical reactions by interactively mixing chemicals and measuring the results. The student fills in the table as he accumulates data to answer the chemistry questions posed (see table widget above). This tutor was designed as an open-ended learning environment without feedback or analysis of student tasks. The knowledge engineer for this tutor was a University of Massachusetts Chemistry professor.

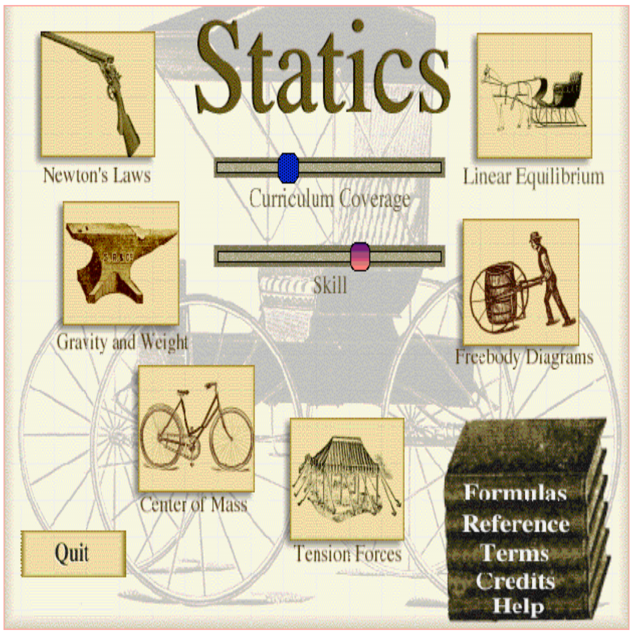


Figure 20: Statics Tutor

The **Statics Tutor** (Figure 20) teaches introductory Statics concepts, and includes the “crane boom” simulation widget (Figure 6C). This tutor has the largest topic network and curriculum knowledge base, comprising 40 topic and misconception nodes (see Figure 5), and hundreds of presentations with multiple-choice questions. Using topic levels called "Summary," "Teach Easy," "Teach Moderate," and "Teach Difficult," in conjunction with topic link types "Familiarity Prerequisite," "Easy Level Prerequisite," and "Moderate Level Prerequisite," simulates a "spiral teaching" strategy, in which the same set of topics are taught several times, but at a higher level of difficulty each time. Most of the knowledge engineering for this tutor was done as done in the precursor project to Eon, KAFITS, discussed in Section 1.1. This content was re-authored and extended using the Eon tools. The domain expert for the Statics Tutor was a "master" teacher in high school physics fro a local high school.

9. CONCLUSION

We have tried to produce a suite of highly usable tools that have enough similarity to off-the-shelf CAI/multimedia authoring tools to support users of these existing tools in making the transition to authoring knowledge based tutors. After using the tools for several tutors and several authors I believe that the authoring system, though fairly large (the user documentation is several hundred pages long), is composed of tools which are quite learnable and usable.

Though, for reasons described above, the Eon tools did not progress past alpha software, I believe that the project contributes to the field by uniquely exemplifying a full-featured, integrated set of tools for authoring all aspects of an ITS. In a companion Chapter to this one (Chapter 15) I describe an generalization of Eon's representational framework, and describe lesson learned under the following issue headings: "Who Are the Users of ITS Authoring Tools?" "From Story Boards To Knowledge Bases," "How Generic Can Teaching Strategies Be?" "Topic Modularity and Interdependence," "Knowledge Structure and Complexity," and "Non-Independence of Content and Strategies."

10. REFERENCES

Anderson, J. R. & Pelletier, R. (1991). A development system for model-tracing tutors. In Proc. of the International Conference on the Learning Sciences, (pp. 1-8), Evanston, IL.

Cheikes, B. (1995). Should ITS Designers be Looking for a Few Good Agents? In AIED-95 workshop papers for Authoring Shells for Intelligent Tutoring Systems.

Clancey, W. & Joerger, K. (1988). "A Practical Authoring Shell for Apprenticeship Learning." *Proceedings of ITS-88*, pp. 67-74. June 1988, Montreal.

Gruber, T. (1987). "A Method for Acquiring Strategic Knowledge from Experts." University of Massachusetts Dissertation.

Lesser, V. (1984). Control in Complex Knowledge-based Systems. Tutorial at the IEEE Computer Society AI Conference.

Major, N. P., (1993). Teachers and Teaching Strategies. *Proc. of the Seventh International PEG Conference*, Heriot-Watt Univ., Edinburgh.

Major, N.P. & Reichgelt, H (1992). COCA - A shell for intelligent tutoring systems. In Frasson, C., Gauthier, G., & McCalla, G.I. (Eds.) *Procs. of Intelligent Tutoring Systems* '92. Springer Verlag, Berlin.

Murray, T. (1993). Formative Qualitative Evaluation for "Exploratory" ITS research. *J. of AI and Education*. V. 4. No. 2.

Murray, T. (1991). *Facilitating Teacher Participation in Intelligent Computer Tutor Design: Tools and Design Methods.* Ed.D. Dissertation, Univ. of Massachusetts, Computer Science Tech. Report 91-95.

Murray, T. (1996a). Special Purpose Ontologies and the Representation of Pedagogical Knowledge. *In Proceedings of the International Conference on the Learning Scieces,* (ICLS-96), Evanston, IL, 1996. AACE: Charlottesville, VA.

Murray, T., Schultz, K., Brown, D., & Clement, J. (1990). An Analogy-Based Computer Tutor for Remediating Physics Misconceptions. *J. of Interactive Learning Environments* , Vol. 1 No. 2, pp. 79-101.

Murray, T. & Woolf, B. (1992a). Results of Encoding Knowledge with Tutor Construction Tools. *Proceedings of AAAI-92*. San Jose, CA., July, 1992.

Murray, T. & Woolf, B. (1992b). Tools for Teacher Participation in ITS Design. In Frasson, Gauthier, & McCalla (Eds.) Intelligent Tutoring Systems, Second Int. Conf. , Springer Verlag, New York, pp. 593-600.

Murray, W.R. (1990). A Blackboard-based Dynamic Instructional Planner. In *Proc. of AAAI-90*.

Ohlsson, S. (1987). Some Principles of Intelligent Tutoring. In Lawler & Yazdani (Eds.), *Artificial Intelligence and Education,* Volume 1. Ablex: Norwood, NJ.

Russell, D., Moran, T. & Jordan, D. (1988). The Instructional Design Environment. In Psotka, Massey, & Mutter (Eds.), *Intelligent Tutoring Systems, Lessons Learned,* Hillsdale, NJ: Lawrence Erlbaum.

Spensley, F., Elsom-Cook, M., Byerley, P., Brooks, P., Federici, M. and Scaroni, C. (1990). "Using multiple teaching strategies in an ITS," in Frasson, C. and Gauthier, G. (eds.), *Intelligent Tutoring Systems: At the crossroads of Artificial Intelligence and Education*. Norwood, NJ: Ablex.

Towne, D.M., Munro, A., (1988). The Intelligent Maintenance Training System. In Psotka, Massey, & Mutter (Eds.), *Intelligent Tutoring Systems, Lessons Learned,* Hillsdale, NJ: Lawrence Erlbaum.

VanLehn, K. (1988). "Toward a Theory of Impasse-Driven Learning." In Mandl, H. & Lesgold, A. (Eds.), *Learning Issues for Intelligent Tutoring Systems*. New York: Springer-Verlag.

Van Marcke, K. (1992). Instructional Expertise. In Frasson, C., Gauthier, G., & McCalla, G.I. (Eds.) Procs. of Intelligent Tutoring Systems '92. New York: Springer-Verlag pp. 234-243.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems.* Los Altos, CA: Morgan Kaufmann.

ACKNOWLEDGMENTS

Many thanks to the following people who were crucial in developing the Eon system: Matt Cornell for realizing the ideas in a gazilllion lines of pristine code; Kate Cremer for early versions of the Topic Net Editor; Erik Eide for widget dialogs and the student model editor; Erik Haugsjaa for continued code maintenance and numerous feature enhancements; Ryan Moore for tutor construction and enthusiasm. This material is based upon work supported by the National Science Foundation and the Advance Research Projects Agency under Cooperative Agreement No. CDA-940860.