Component-Based Construction of a Science Learning Space

Kenneth R. Koedinger, Daniel Suthers, Kenneth D. Forbus

International Journal of Artificial Intelligence in Education (1999), 10, 292-313

Component-Based Construction of a Science Learning Space

Kenneth R. Koedinger *Human-Computer Interaction Institute, Carnegie Mellon University*
koedinger@cs.cmu.edu

Daniel D. Suthers *Information and Computer Sciences, University of Hawai'i*
suthers@hawaii.edu

Kenneth D. Forbus *Institute for the Learning Sciences, Northwestern University*
forbus@ils.nwu.edu

Abstract. We present a vision for learning environments, called Science Learning Spaces, that are rich in engaging content and activities, provide constructive experiences in scientific process skills, and are as instructionally effective as a personal tutor. A Science Learning Space combines three independent software systems: 1) simulations in which experiments are run and data is collected, 2) representation construction tools in which data is analyzed and conceptual models are expressed and evaluated, and 3) tutor agents that provide just-in-time assistance for acquiring higher order skills like experimental strategy, representational tool choice, conjecturing, and argument. Achieving the Science Learning Space vision will require collaborative efforts facilitated by a component-based software architecture. We have created a feasibility demonstration that serves as an example and a call for further work toward achieving this vision. In our demonstration, we combined 1) the Active Illustrations simulation environment, 2) the Belvedere evidence mapping environment, and 3) a model-tracing Experimentation Tutor Agent. We illustrate student interaction in this Science Learning Space and discuss the requirements, advantages, and challenges in creating one.

THE SCIENCE LEARNING SPACE VISION

Imagine an Internet filled with possibility for student discovery. A vast array of simulations is available to explore any scientific field you desire. Easy-to-use data representation and visualization tools are at your fingertips. As you work, intelligent tutor agents are watching silently in the background, available at any time to assist you as you engage in scientific inquiry practices: experimentation, analysis, discovery, argumentation. This is our vision for Science Learning Spaces. Table 1 summarizes how this vision contrasts with typical classroom experience.

Table 1. What Science Learning Spaces Have to Offer

	Typical Science Class	**Science Learning Space Vision**
Content	Lectures, fixed topics, fixed pace, focus on facts	Vast options, student choice and pace, learning both concepts and process skills
Activity	Inquiry process hampered by mundane procedure & long waits	Simulations speed time leave technique lessons for later
Tools	Paper and pencil	Data representation & model construction tools
Assistance	Limited, 1 teacher for 30 students	Automated 1:1 assistance of tutor agents
Assessment	Large grain, limited assessment-instruction continuity	Tutor agents monitor student development at action level

Unlike the traditional science classroom of lectures, fixed topics and a fixed pace, in a Science Learning Space the students choose areas of interest, work at their own pace, and engage in learning by doing. Unhampered by "test tubes and time," they work with simulations

that simplify, accelerate, and make possible otherwise mundane, time-intensive, dangerous, or even impossible experiments and investigations. In addition to recording data and ideas on paper, students use sophisticated computer-based tools for data representation and model construction. Rather than competing with 30 other classmates for help from one teacher, each student can always get assistance from one or more computerized tutor agents in addition to occasional help from the teacher. In effect, the classroom has a teacher's aid for every student. Finally, when it comes to assessment, the teacher can supplement occasional paper-based assessments with the detailed, continuous assessment provided by tutor agents. Unlike paper assessments, tutor agents capture students' success or difficulty with specific concepts and skills, and do so automatically and in the context of complex activities.

This powerful combination of content, activity, tools, assistance, and assessment is achieved within a Science Learning Space by coordinating software components of three types:

simulations in which experiments are run and data is collected,

representation construction tools in which data is analyzed and conceptual models are expressed and evaluated, and

tutor agents that provide just-in-time assistance in acquiring higher order skills like experimental strategy, representational tool choice, conjecturing, and argument.

Although the full Science Learning Space vision is currently out of reach, we have created a demonstration of its feasibility. This demonstration serves as a model for future work and a call for further community efforts directed toward achieving this vision.

The Need for Collaborative Component-Based Development

Research in intelligent learning environments typically involves designing and implementing an entire system from scratch. Time and resources spent on software engineering is taken away from the education and research the software is designed to support. Today the typical solution in research labs and industry is to work within the context of an in-house software investment, evolving each new system from previous work. This makes replication and sharing more difficult and can lead to maintenance and deployment difficulties as restrictive platform requirements accumulate over time.

This situation is growing intolerable, and so recently there has been a surge of interest in architectures and frameworks for interoperable and component-based systems for education [Ritter & Koedinger, 1997; Roschelle & Kaput, 1995; Roschelle *et al.* 1998; Suthers & Jones, 1997]. This has led to a number of successful workshops on the topic,[1] several standards efforts specifically targeted to advanced educational technologies,[2] and new repositories for educational object components.[3] These efforts gain leverage from the rise of interactive Web technology and its associated emphasis on standards-based interoperability. Emerging solutions for component-based systems include development frameworks,[4] shared communication protocols,[5] markup languages,[6] and metadata formats.[7] Although the component-based solutions developed to date are useful, they are inadequate for those building component-based intelligent learning environments in which the components must respond to the meaning of the

1 E.g., "Architectures and Methods for Designing Cost-Effective and Reusable ITSs" at ITS'96 (http://advlearn.lrdc.pitt.edu/its-arch/), and "Issues in Achieving Cost-Effective and Reusable Intelligent Learning Environments" at AI-ED'97 (http://www.fu.is.saga-u.ac.jp/aied/workshop2.html).

2 E.g., ARIADNE's Educational Metadata Recommendation (http://ariadne.unil.ch/metadata.htm), the IEEE 1484 Learning Technology Standards Committee (http://www.manta.ieee.org/p1484/n2index.htm), and Educom's IMS (http://www.imsproject.org/).

3 E.g., the Educational Object Economy (http://www.eoe.org/) and Gamelan (http://www.developer.com/directories/pages/dir.java.educational.html).

4 E.g., Java Beans (http://java.sun.com/docs/books/tutorial/javabeans/index.html).

5 E.g., CORBA (http://www.omg.org/news/begin.htm).

6 E.g., XML (http://www.w3.org/TR/PR-xml.html).

7 E.g., the IMS/LTSC work mentioned in footnote 2.

content as well as its form and presentation. We see the development of techniques for sharing semantics across components and applications to be a critical research direction for the field.

Recently we conducted a demonstration of the feasibility of integrating three different, independently developed components. Two of the components were complete intelligent learning environments in their own right. Active Illustrations [Forbus, 1997] enable learners to experiment with simulations of scientific phenomena, and to receive explanations about the causal influences behind the results [Forbus & Falkenhainer 1990; 1995]. Belvedere [Suthers & Jones, 1997; Suthers et al., 1997] provides learners with an "evidence mapping" facility for recording relationships between statements labeled as "hypotheses" and "data". A Scientific Argumentation Advisor [Paolucci et al., 1996] guides students to seek empirical support, consider alternate hypotheses, and avoid confirmation biases, among other things. The third component was an instance of a model-tracing Tutor Agent [Ritter & Koedinger, 1997] that contains a cognitive model of general experimentation and argumentation process skills. This "Experimentation Tutor Agent" dynamically assesses student performance and is available to provide students with just-in-time feedback and context-sensitive advice. Our Learning Space Demonstration took place in the context of meetings of DARPA's Computer Aided Education and Training Initiative program contractors. Using a MOO as a communication infrastructure [O'Day, Bobrow, Bobrow, Shirley, Hughes, Walters, 1998], we demonstrated a scenario in which a student poses a hypothesis in the Belvedere evidence-mapping environment, uses the simulation to test that hypothesis in the Active Illustration environment and sends the results back to Belvedere for integration in the evidence map. Throughout this activity the Experimentation Tutor Agent was monitoring student performance and was available to provide assistance.

From this experience we abstracted the notion of a Science Learning Space. In our demonstration, the Space was filled with Active Illustrations as the simulation component, Belvedere as a representation construction tool, and the Experimentation Tutor Agent and Argumentation Advisor in tutor roles. In this paper we discuss how interoperability of these components was achieved through the use of Translator components that enable communication between existing functional components with little or no modification to them. We begin by examining the constraints that developing intelligent learning environments impose on the nature and types of components and their interactions, focusing on the importance of semantic interoperability. We then describe the demonstration configuration in detail, showing how it exploits a limited form of semantic interoperability. Finally, we reflect on the requirements, advantages, and future directions in creating Science Learning Spaces.

COMPONENTS FOR INTELLIGENT LEARNING ENVIRONMENTS

Component-based development has a number of purported economic and engineering benefits. Component-based systems are considered to be more economical to build because prior components can be re-used, saving time for new research and development efforts. They are easier to maintain due to their modular design and easier to extend because the underlying frameworks that make component-based development possible in the first place also make it easier to add new components. We can also expect better quality systems as developers can focus their efforts on their specialty, whether in simulation, tool, or tutor development.

However, there is a deeper reason why we believe component-based educational software is important: It will enable us to construct, by composition, the multiple functionalities needed for a pedagogically complete learning environment. Various genres of computer-based learning environments have had their advocates. Each provides a valuable form of support for learning, but are insufficient in themselves. Yet today, the high development costs associated with building each type of environment leads to the deployment of systems with only a small subset of desirable functionality.

For example, *microworlds and simulations* enable students to directly experience the behavior of dynamic systems and in some cases to change that behavior, experimenting with

alternate models. These environments are consistent with the notion that deeper learning takes place when learners construct knowledge through experience[8]. However, simulations lack guidance: Taken alone, they provide no tools for the articulation and reflection on this knowledge and no learning agenda or intelligent assistance.

On the other hand, *intelligent tutoring systems* provide substantial guidance in the form of a learning agenda, a model of desired thinking and performance, and intelligence assistance. This form of guidance is particularly important in domains where the target knowledge is not an easy induction from interactions with the artifact or system of interest. In such domains, intelligent tutors can lead to dramatic "one sigma" increases in student achievement [e.g., Koedinger, Anderson, Hadley, & Mark, 1997].

However, while intelligent tutoring systems are sometimes designed with sophisticated and novel interfaces [e.g., Koedinger, 1991; Reiser et al., 1991], such interfaces are not the core of the approach. Tutor interfaces are not typically designed to afford and support desired student thinking processes like data manipulation, pattern search, or the expression and testing of new knowledge. A third type of educational component, *representational tools* , fill this need. Representational tools range from basic office software such as spreadsheets, graphers, and outliners to "knowledge mapping" software and other specialized tools designed based on cognitive analyses to address particular learning objectives [Koedinger, 1991; Reiser et al., 1991; Reusser, 1993]. Properly designed, representational tools can function as *cognitive* tools that lead learners into knowledge-building interactions [Collins & Ferguson, 1993; Goldenberg, 1995; Kaput, 1995], and guide collaborative as well as individual learning interactions [Roschelle, 1996]. As learner-constructed external representations become part of the collaborators' shared context, the distinctions and relationships that are made salient by these representations may influence their interactions in ways that influence learning outcomes [Suthers, 1999].

Yet representational tools provide only a subtle kind of guidance. As with simulations and microworlds, direct tutoring interventions are sometimes needed as well. Fortunately there is a double-synergy: Inspection of learners' representations and simulation actions can provide a tutor with valuable information about what students are thinking and thus what kind of guidance is needed.

We believe that the potential to routinely synthesize new intelligent learning environments from off-the-shelf components that combine multiple functionalities is sufficient justification for moving to a component-based development approach. The potential advantages of component-based systems must, of course, be weighed against their costs. Creating composable software components requires exposing enough of their internal representations, through carefully designed protocols, so that effective communication is possible. Doing this in ways that minimize communication overhead while maximizing reuse is a subtle design problem that can require substantial extra work. Communication between multiple development teams in different locations also provides a substantial challenge.

A FEASIBILITY DEMONSTRATION OF A SCIENCE LEARNING SPACE

In this section we describe the Science Learning Space demonstration that we undertook. We begin with the learning activity that motivates our particular combination of tools; then we describe the underlying architecture and step through an example interaction scenario.

The Learning Activity: Scientific Inquiry

There is no point in combining components unless the learner benefits. In particular, the functionality provided by each component must contribute to the facilitation of effective learning interactions in some way. Consider scientific inquiry. Students have difficulty with

8 For a review of research on the features and benefits of simulations see de Jong & van Joolingen (1998).

the basic distinction between empirical observations and theoretical claims and particularly in coordinating the two (Kuhn, Amsel, O'Loughlin, 1988; Ranney, *et al.* 1994). They need to learn that theories are posed to explain and predict occurrences and that theories are evaluated with respect to how consistent they are with all of the relevant observed data. They need to seek relevant evidence, both confirming and disconfirming, perform observations, and conduct experiments to test hypotheses or to resolve theoretical arguments between hypotheses. Experimentation requires certain process skills, such as the strategy of varying one feature at a time. Evaluation of the results of experiments requires scientific argumentation skills. Thus, this is a learning problem that could benefit from (1) experimentation in simulation environments, aided by coaching based on a process model of effective experimentation; and (2) articulation of and reflection upon one's analysis of the relationships between hypotheses and evidence, aided by coaching based on principles of scientific argumentation.

The Implementation Architecture

We describe the abstract implementation architecture (see Figure 1) behind our demonstration as one illustration of how several technologies enable the construction of component-based systems. Our collaboration began with a Science Learning Space demonstration involving an Experimentation Tutor Agent and Active Illustration [Forbus, 1997] communicating through a Lambda-MOO derivative using the "MOO Communications Protocol" [O'Day, et al., 1998]. Forbus had already made use of the MOO for communication between the Active Illustration simulation engine and a simulation user interface (upper right of Figure 1). A MOO was chosen as the infrastructure because its notion of persistent objects and multi-user design made it easy for participants in experiments (both human and software) to be in a shared environment despite being on different machines, often in different parts of the country. The open, ASCII-based MOO Communications Protocol made it easy to add a Tutor Agent to monitor student performance as the basis for providing context-sensitive assistance. Koedinger used the plug-in tutor agent architecture [Ritter & Koedinger, 1997] which employs a simple Translator component (box #1, lower right of Figure 1) to manage the communication between tools and tutor agents. The Translator watched for messages between the Simulation Interface and the Active Illustration server, extracted messages indicating relevant student actions, and translated these student actions into the "selection-action-input" form appropriate for semantic processing by the Tutor Agent's "model-tracing" engine. Model-tracing is a plan recognition technique used in Cognitive Tutors to parse student action sequences in terms of a goal-based production rule sequence that could have generated such actions [Anderson, Corbett, Koedinger, & Pelletier, 1995].

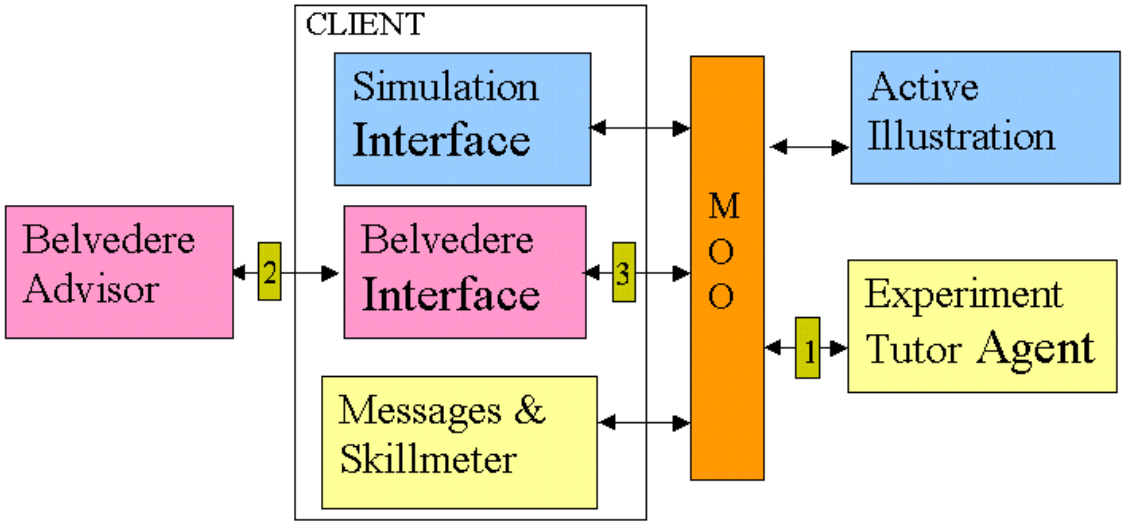


Figure 1. Communication architecture used in the demonstration**.**

In the first of two Science Learning Space demonstrations we created, the science domain was evaporation processes, a typical middle school science topic. To demonstrate the domain generality of Active Illustrations and the Experimentation Tutor Agent, we changed the science domain in our second Science Learning Space demonstration to atmospheric phenomena, of particular interest because of its relationship to global warming. More significantly, in this second demonstration we added a Representation Construction tool, the Belvedere system, and a second tutor agent, the Argumentation Advisor (see the left side of Figure 1). Belvedere itself provides a communication architecture, described in Suthers & Jones [1997], which is shown in Figure 1 as box #2, between the Belvedere Advisor and Interface. Integration of the Belvedere subsystem into the MOO required the addition of one translator component (box #3 in the figure). No modification to Belvedere itself was required. The translator watched the MOO for Hypothesis and Simulation Run objects sent by the Simulation Interface. When seen, these were converted to Belvedere Hypothesis and Data objects, and sent into Belvedere for consideration.

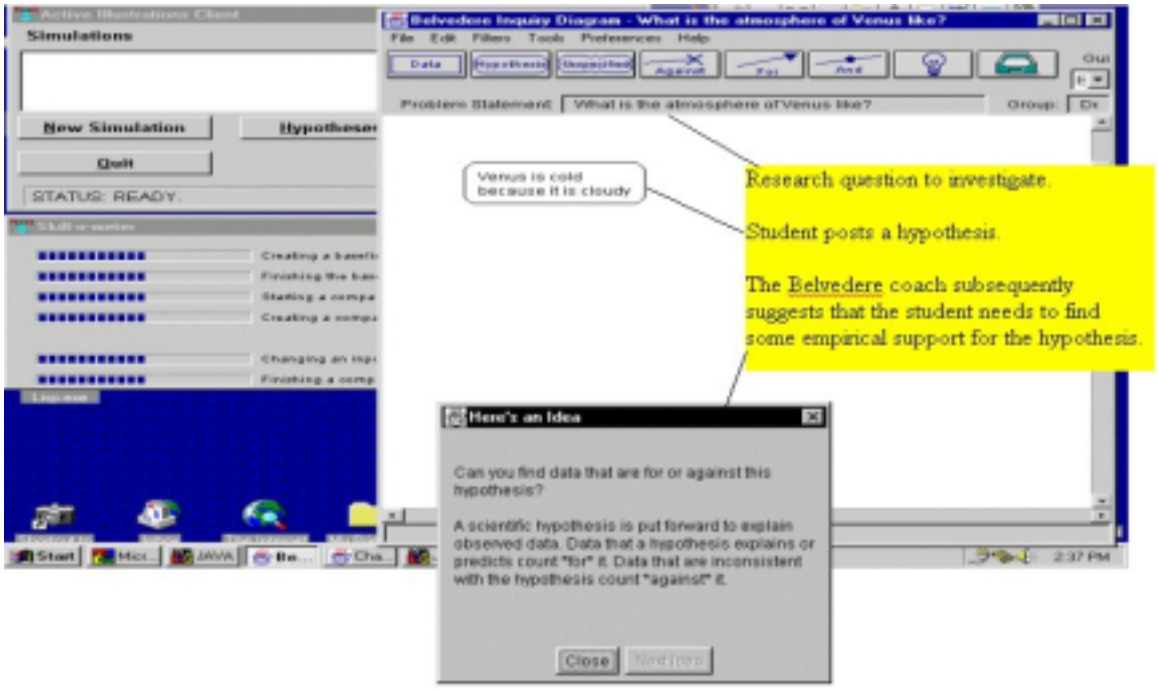


Figure 2. Opening situation: An initial hypothesis is posed in Belvedere.

Learning Scenario

In our demonstration scenario, we imagine a student engaging in an investigation of the climate of Venus. She starts by posing a plausible hypothesis that Venus is cold because its excessive cloud cover makes it so. Next, she uses the multiple tools and intelligent assistance of our multi-application Science Learning Space to record, empirically test, formally evaluate and revise this hypothesis.[9]

In the opening situation (Figure 2), the student has recorded in Belvedere's evidence-mapping facility the hypothesis that Venus is cold because it is cloudy. On the left are the Active Illustration simulation interface and the Tutor Agent's Skillometer showing initial estimates of the student's level of skill on several subskills. Next, the student invokes Belvedere's Argumentation Advisor, which applies its "Hypothesis Lacks Empirical Evidence"

9 The full example is available at http://advlearn.lrdc.pitt.edu/advlearn/examples/caeti97demo.html

strategy (Figure 9) to suggest that the student find some evidence for this hypothesis (bottom of Figure 2). Since the student can't go to Venus to experiment, she decides to use an Active Illustration simulation of the Earth's atmosphere as an analog instead.

In the Active Illustration interface, the student runs a baseline simulation using normal cloud cover parameters for Earth (Figure 3). The Tutor Agent observes these events in the MOO and updates estimates of baseline experimentation skills (lower portion of Figure 3 and 4). Then the student constructs a comparison simulation by increasing the Earth's cloud cover to 95%, much more like Venus (Figure 4). This action is model-traced by the change-one-variable production in the Experimentation Tutor Agent (see Table 2) and the "Finishing a comparison experiment" skill (bottom of Figure 4) will update once the student clicks Okay and then Simulate without changing any other variables.

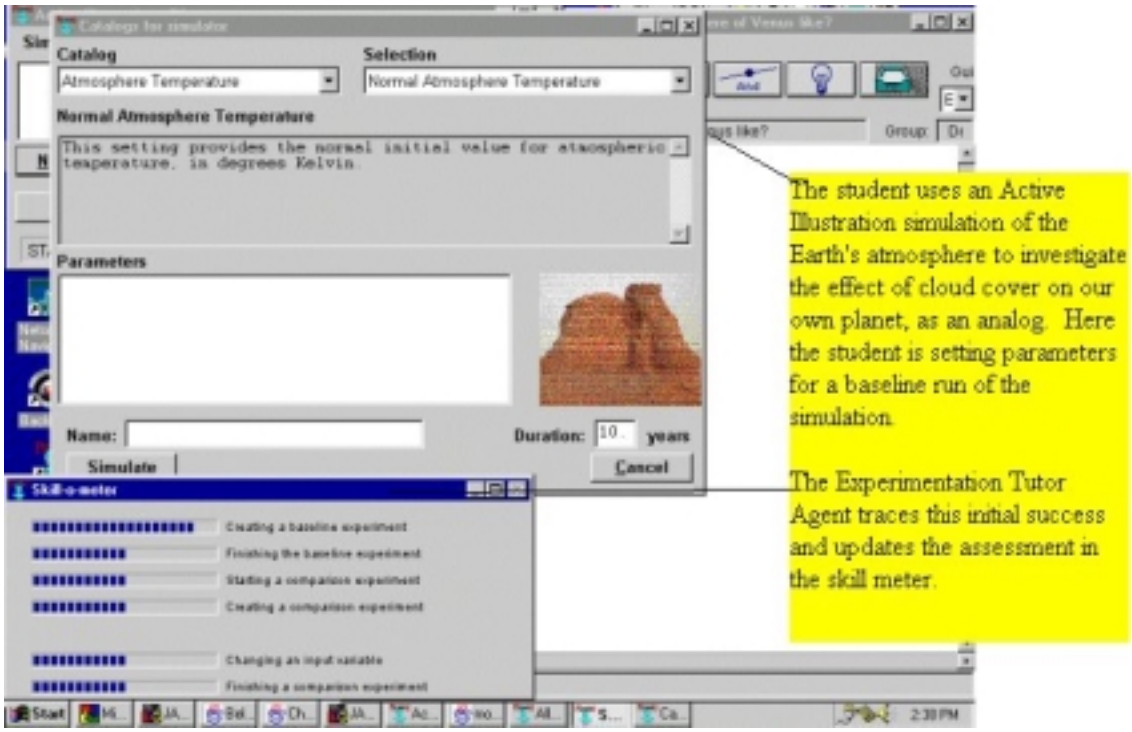


Figure3. The student setups up a baseline experiment in Active Illustrations to test the hypothesis posed in Belvedere.

Table 2. Domain-Independent Productions for Experimentation and Argument

```
Change-more-than-one-variable-bug (Pre-conception)
IF the goal is to discover a hypothesis
   and you have a first experiment
THEN change some variable values to create a second experiment
Change-one-variable
IF the goal is to discover a hypothesis
   and you have a baseline experiment
THEN change one variable value to create a comparison experiment
One-trial-generalization-bug (Pre-conception)
IF the goal is argue for hypothesis "The greater the <cloud cover>
     the higher the <atmospheric temperature>"
   and you did an experiment where <cloud cover> was high
   and the resulting <atmospheric temperature> was high
THEN argue the hypothesis is true by citing this experiment
Argue-from-controlled-comparison
IF the goal is argue for hypothesis "The greater the <cloud cover>
     the higher the <atmospheric temperature>"
   and you did two experiments, a baseline and comparison
   and in one <cloud cover> was low and <temp> was low
   and in the other <cloud cover> was higher and <temp> was higher
THEN argue the hypothesis is true by citing these two experiments
```

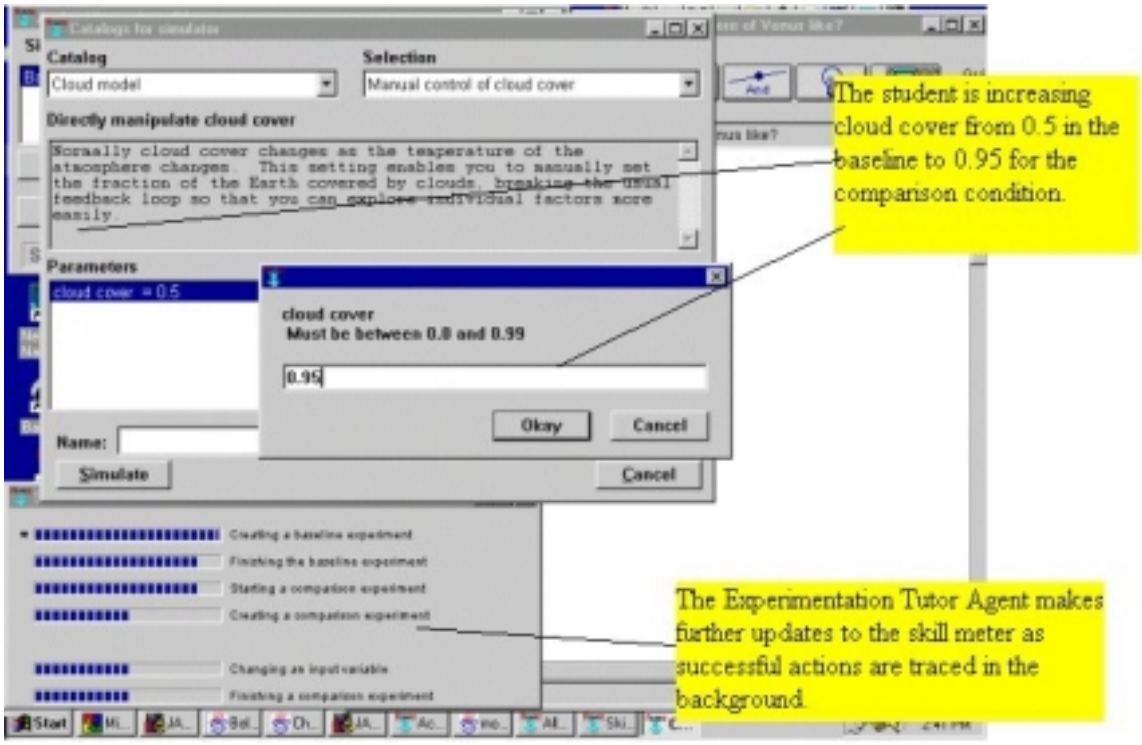


Figure 4. Changing one variable at a time.

In contrast to this desirable experimentation strategy of changing one variable at a time, students will typically change multiple variables when first encountering such experimental situations (Case, 1974; Chen & Klahr, 1999; Kuhn, Amsel, O'Loughlin, 1988). For instance, instead of just changing the cloud cover, the student may have also changed the atmospheric pressure. In this case, the Experimentation Tutor Agent would trace this action as an instance of the change-more-than-one-variable bug (see Table 2). It would generate a feedback message: "If you change both the cloud cover and the atmospheric pressure, you will not know which of these factors affects the temperature." Chen and Klahr (1999) have experimentally demonstrated not only that most students do not spontaneously use this strategy, but that through instruction and practice with feedback students can learn it and do so in a fairly general form that transfers from one science domain to another.

Returning to the scenario, after the student has created baseline and comparison experiments, she uses Active Illustration's plotting facility to show the results (top of Figure 5). To her surprise, Earth gets *hotter* when cloud cover increases. The student realizes that a new hypothesis is required. Using Active Illustration's hypothesis recording facility, the student creates the hypothesis that the temperature of the atmosphere is affected positively by cloud cover and marks it as confirmed (bottom of Figure 5).

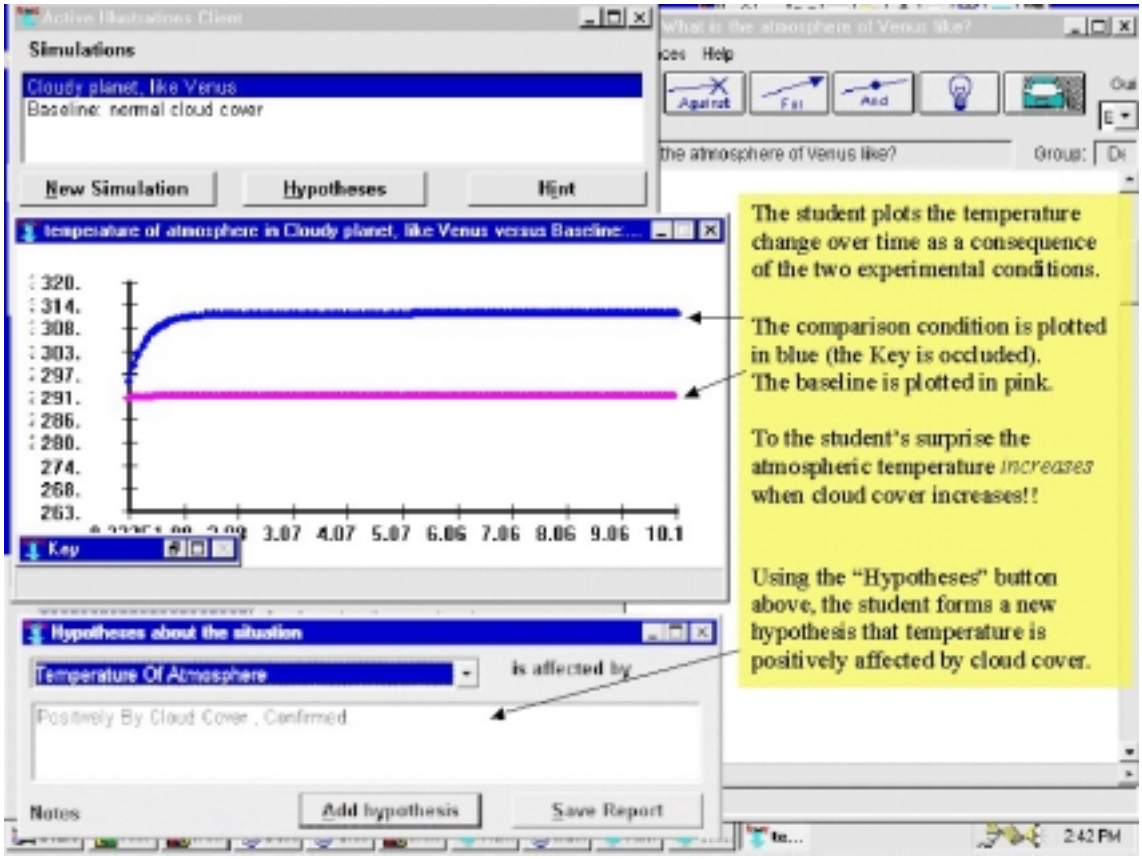


Figure 5. Plotting the results of the baseline and comparison experiments.

The student then decides to send this hypothesis along with the results of the two experiments back to the Belvedere application. She does this by selecting these objects and clicking a "send to other tools" button. The objects are sent out on the MOO, where they are observed by the Belvedere-MOO Translator. This Translator component filters messages, translates hypothesis and data objects from the MOO into Belvedere objects, and then places these objects in Belvedere's "in-box" (see the box icon in the upper right corner of Figures 2-7). The in-box is a place where new information is kept until the student is ready to integrate it into the evidence map. The student selects information objects from the in-box and places them in her evidence map. At this point she uses Belvedere to construct the argument shown in Figures 6 and 7. She starts by using an evidential "against" link to indicate the new hypothesis "temperature of atmosphere is positively affected by cloud cover" contradicts the original hypothesis "Venus is cold because it is cloudy" (Figure 6). Not sure how to proceed, she clicks on the "idea generator" button and the Belvedere argumentation advisor responds (upper left corner of Figure 6): "In science, we try to explain observed data with hypotheses that say why the data are so." The student responds by successfully completing the argument (Figure 7). Because both experiments are needed to support the hypothesis that temperature increases with cloud cover, she connects them together with an "and" link. Finally, she completes the argument by connecting the conjoined experiments to the hypothesis with an evidential "for" link.

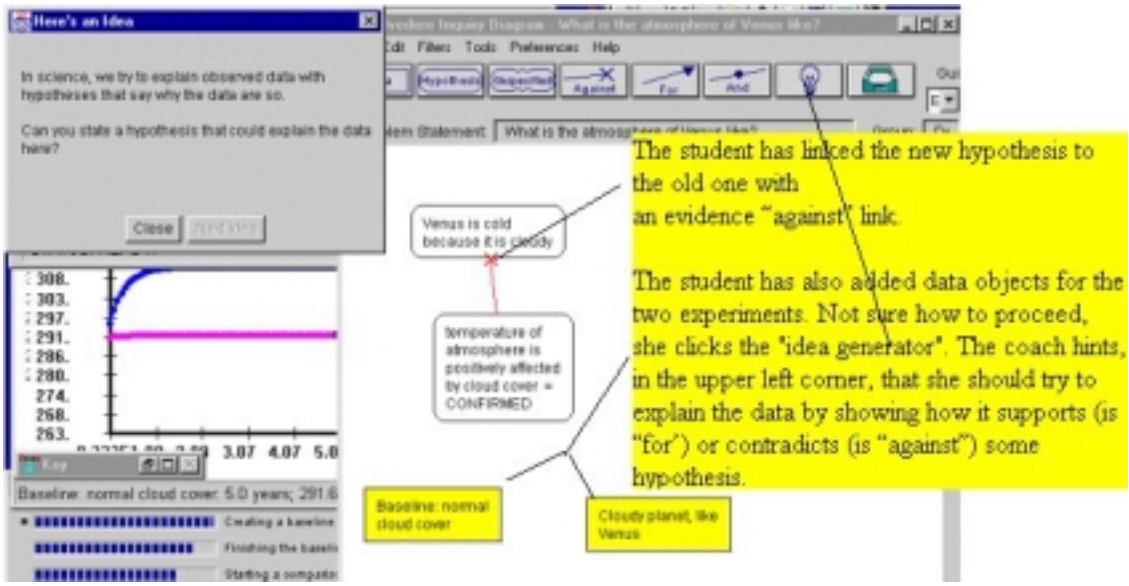


Figure 6. Constructing the argument with Belvedere advice.

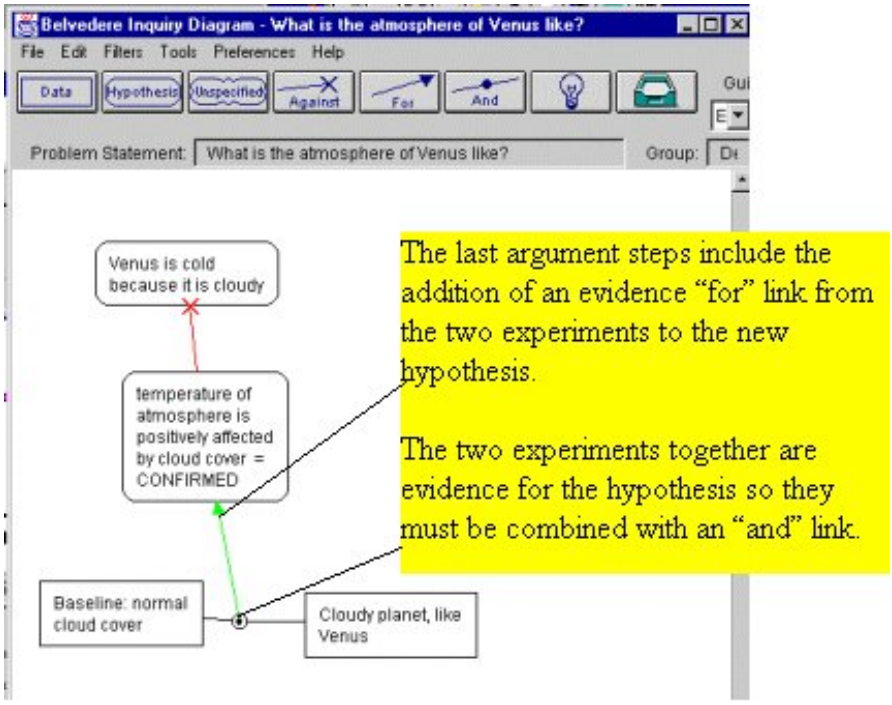


Figure 7. Final evidence map showing how both experiments are needed to support the new hypothesis which, in turn, contradicts the original hypothesis.

The Experimentation Tutor Agent is able to understand the semantics of the experiment and hypothesis objects because it observed the history of the creation of these objects in the Simulation. Specifically, the Agent was designed to understand the semantics of objects created in the Simulation. In composing our tools, we arranged for the object identifier that originated in the Simulation to persist as the object moves to Belvedere. Hence, for the Tutor Agent, the semantics also transfer to Belvedere.

As a consequence, either this agent or the Argumentation Advisor can provide the student with advice on the semantics of the argument. If, for instance, the student were to try to use only the comparison experiment as evidence for the hypothesis, the Tutor Agent can remediate

this argumentation bug with advice that relies on the experimental context (see Table 2): “Just because the temperature was high with high cloud cover is not enough to argue for your hypothesis; you must also cite the contrasting baseline experiment where the temperature was lower with normal cloud cover.”

Recognizing the need to use both experiments is a critical reflection step for students. Although the thinking process here is similar to the change-one-variable step in experimentation, the explication in the argument requires students to reflect on and redescribe in words what they did earlier in action. It is possible for students to learn change-one-variable as a procedure and not have an understanding of why it is a good idea (Case, 1974; Kuhn, Amsel, O’Loughlin, 1988), namely, because it helps discriminate between alternative hypotheses. The argument process provides an opportunity for students to further develop such an understanding.

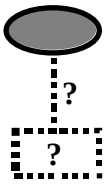
Other Forms of Pedagogical Advice

We want to emphasize that the Experimentation Tutor Agent was not the only source of explicit advice and instruction in our Science Learning Space demonstration. Active Illustrations and Belvedere have capabilities for providing explanations and critic-based advice, respectively.

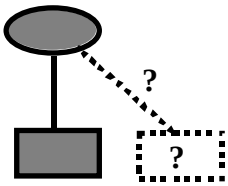
The explanations provided by Active Illustrations concern the qualitative, causal structure of the system being simulated. Explaining the qualitative mathematical relationships between parameters (e.g., "the temperature of the atmosphere is affected by the incoming solar radiation") provides an appropriate level of description for science learners since it does not presume a sophisticated understanding of mathematics. Such explanations are an English rendering of an *influence* (e.g., between solar radiation and temperature) from QP theory [Forbus, 1984]. These explanations both reify physical processes that are acting in the simulated situation and identify qualitatively interesting events (e.g., water completely evaporating away). Thus, they help students organize causal knowledge and link it into real-world events.

In Active Illustrations, the explanation system is an intrinsic part of the simulator. That is, the conceptual explanations the compiler generated to write the simulator code are also used to generate the runtime explanation system. This means that the usual problems of grafting an explanation system onto a simulator post hoc are avoided. These problems include doing the signal processing and recognition necessary to identify interesting patterns in the behavior, and verifying that the explanation system and simulator do not get out of synch.

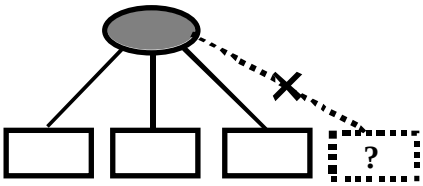
Like the Experimentation Tutor Agent, the Belvedere Argument Advisor can also provide advice to the student. In contrast to the Experimentation Tutor Agent, it gives advice in response to situations that can be defined on a purely syntactic basis, using only the structural and categorical features of the students’ argument graphs. The students’ text, which is the content of the nodes in the argument graph, is not interpreted. Principles of scientific inquiry are instantiated as patterns to be matched to the diagram and textual advice to be given if there is a match. Example advice patterns from the Lisp-based Belvedere 2.0 implementation are given in Figure 8. When the solid-lined portions are present and the dashed portions are missing, the corresponding advice can be given. Typically, several advice patterns will match an evidence map, sometimes with multiple matches per pattern. A student could not effectively absorb all this advice if it were given at one time. Thus, it is necessary to be selective. Selection is performed by a preference-based quick-sort algorithm, described in Suthers et al. (in press). Preferences compare advice on factors such as (a) whether the advice has already been given, (b) whether it addresses representations recently manipulated by the present user, or (c) whether it addresses critical cognitive biases versus more mundane issues.



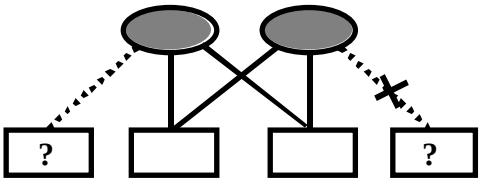

```
(def-advice 'HYPOTHESIS-LACKS-EMPIRICAL-EVIDENCE
  :query '(retrieve (?h) (and (hypothesis ?h) (No-Evidencep
?h)))
  :advice ("Can you find data that are for or against this
hypothesis? A scientific hypothesis is put forward to explain
observed data. Data that a hypothesis explains or predicts count
*for* it. Data that are inconsistent with the hypothesis count
*against* it.")
  :subsequent-advice ("Can you find some data for or against
this hypothesis?")
  :advice-types '(incompleteness))
```




```
(def-advice 'ONE-SHOT-HYPOTHESIS
  :query '(retrieve (?d ?h)
             (and (data ?d) (hypothesis ?h)
                  (Consistent-HypoP ?d ?h)
                  (fail (Exists-Multiple-Consistent-DataP ?h))
                  (fail (Exists-Inconsistent-DataP ?h))))
  :advice ("Strong hypotheses and theories usually have a lot of
data to support them. However, this hypothesis has only one
consistent data item. It looks rather weak. Can you find more
data for this hypothesis? Can you find data that is against
it?")
  :subsequent-advice ("This hypothesis has only one consistent
data item. Could you find more data for (or against) this
hypothesis?")
  :advice-types '(evaluative incompleteness))
```




```
(def-advice 'CONFIRMATION-BIAS
  :query '(retrieve (?h)
             (and  (hypothesis ?h)
                   (Exists-Multiple-Consistent-DataP
?h)
                   (Multiply-LinkedP ?h)
                   (fail (Exists-Inconsistent-DataP
?h))))
  :advice ("You've done a nice job of finding data
that is consistent with this hypothesis. However, in
science we must consider whether there is any
evidence *against* our hypothesis as well as
evidence for it. Otherwise we risk fooling ourselves
into believing a false hypothesis. Is there any
evidence against this hypothesis?")
  :subsequent-advice ("Don't forget to look for
evidence against this hypothesis!")
  :advice-types '(cognitive-bias))
```




```
(def-advice 'DISCRIMINATING-EVIDENCE-NEEDED
  :query '(retrieve (?h1 ?h2)
            (and (hypothesis ?h1) (hypothesis
?h2)
                 (not (same-as ?h1 ?h2))
                 (Exists-Consistent-DataP ?h1)
                 (Exists-Consistent-DataP ?h2)
                 (fail (Consistent-HypoP ?h1
?h2))
                 (Identical-EvidenceP ?h1 ?h2)))
  :advice ("These hypotheses are supported by
the same data. When this happens, scientists
look for more data as a \"tie breaker\" --
especially data that is *against* one
hypothesis. Can you produce some data that
would \"rule out\" one of the hypotheses?")
  :subsequent-advice ("Can you produce some
data that might support just one of the
hypotheses?")
  :advice-types '(incompleteness evaluative))
```

Figure 8. Examples of Belvedere's domain independent argument advisor patterns.

The Belvedere Argument Advisor is complementary with the Experimentation Tutor Agent in two ways. First, the Argument Advisor assesses on the general syntactic form of the argument and ignores the content, while the Tutor Agent assesses the content, in particular, to determine whether particular hypotheses are adequately supported by particular pieces of experimental evidence. The systems are also complementary in a second broader way in that they take different approaches to student modeling. The Argument Advisor takes a critic

approach, evaluating the state of learner-constructed representations. The Tutor Agent, in contrast, employs a model tracing approach that evaluates learner actions (or changes in state).

Our Science Learning Space demonstration did not explicitly address the issue of coordinating multiple sources of advice from different agents [cf., Ritter, 1997]. Our implicit solution, which may work in many cases, was to allow the agents to act independently. Any agent might offer up a piece of advice at any time.

DISCUSSION

In the following sections we highlight key features and lessons that emerged from our Science Learning Space demonstration. The discussion proceeds from more general software development issues to more particular science pedagogy issues. We start with a discussion of issues of component-based system construction and, in particular, how semantic interoperability can be achieved among multiple components. The lessons described here are not only relevant to Science Learning Spaces, but also to other component-based development efforts.

Next, we move to a discussion of the potential of domain independence in the components that fill the three roles in the Science Learning Space architecture: simulation, representation constructor, and tutor agent. The components we used, Active Illustrations, Belvedere, and the Experimentation Tutor Agent, are not limited to the domain of atmospheric phenomena: they can be adapted to create learning environments for other domains of scientific experimentation and reasoning.

Finally, we discuss science pedagogy issues that emerge from interactions between Science Learning Space components. In particular, we illustrate how pedagogical challenges that arise within a specific type of educational software system (simulation, representation tool, or intelligent tutor) can sometimes be resolved through the composition of these systems.

Semantic Interoperability for Constructive Learning Interactions

Since the three systems we composed made reference to the same set of objects (e.g., experiments, data, hypotheses), it was critical that a shared semantics was achieved. Below we discuss some alternate solutions we considered, and their roles.

Shared Ontologies

One possible approach to achieving semantic interoperability is to have a common ontology of educational objects that each system accesses (Mizoguchi *et al.* 1997). Significant portions of the communications between developers were, in effect, a process of negotiating an informal shared ontology. The process may have been more efficient and involved fewer misunderstandings if a standard ontology or reference vocabulary were available and known to all.

While shared ontologies may be worthy goals in the long term, they require a high level of community consensus and standardization that is still well out of reach (if not in defining the standards, certainly in having them take hold). Standardization of representations used internally by the component applications should not be the goal. There is good reason to believe that multiple alternative representations of the same objects or concepts are not only inevitable, but useful. Different representations afford different kinds of processing. For example, the representation of an experiment in the simulation stores numerous simulation-related details whereas the Tutor Agent's representation of an experiment is at a more abstract level appropriate for reasoning about experimental design. The Belvedere abstracts further to a level appropriate for reasoning with the results of, rather than about, experiments.

Translators to Preserve Advantages of Alternative Representations

We recommend the use of Translator components because they allow developers of component systems to make their own representational decisions. Once these decisions have been made, developers can get together to identify the shared semantics and specify translators to implement them. It is not necessary to work out ahead of time the precise meaning and structure of all representations. The Translator components were critical to the relative ease in which we composed the three systems. Our composition task would not have been as easy, however, if Active Illustrations and Belvedere had not built from the start in an open communication architecture. These tools were "recordable" and "scriptable" [Ritter & Koedinger, 1997] implying that the Tutor Agent could observe or "record" all user interactions in the tool and, potentially, manipulate or "script" the tool interface. Unfortunately, too few office applications or educational objects are currently built with such an open architecture.

Translators provide a solution for learning environments built upon communication between component tools. Alternatively, component-based learning environments can be based on the embedding of one tool – or set of active representations – inside another. Such environments are exemplified by the ESCOT project, which uses the term "Wrappers" to describe code added to implement connections between components [10]. Whereas we have used Translators to connect components that are complete applications in their own right [see also Ritter & Koedinger, 1997], Wrappers typically connect smaller components and may add functionality. Despite this variation, a major common role of both Translators and Wrappers is to translate between differences in representation of the same or similar object in the two connected components.

Granularity and Persistence of Identity

The Active Illustrations simulation and Simulation Interface used the MOO to communicate in terms of the multiple parameter settings that define a simulation run or experimental trial. However, we wanted experimental trials to appear in Belvedere as single nodes in the evidence map. We needed a way to bridge this difference in granularity while preserving the essential semantic identity of object representations as they are moved from tool to tool. This design problem ultimately led us to better understand how the software's need for persistence of identity can sometimes be solved by addressing the learner's same need with a stronger coupling between the component interfaces.

We initially considered solving this problem by using the Belvedere-MOO Translator to aggregate individual parameter setting and simulation events into "data" objects that record the results of a particular trial. These data objects would then appear automatically in Belvedere's in-box. However, focusing on the needs of the learner, we elected to follow a different approach for three major reasons. (1) Not all simulation runs will be informative enough to use. We wanted to avoid cluttering Belvedere's representations with many not so useful objects. (2) We wanted to encourage the learner to reflect on which simulation runs were worth recording, by requiring that the learner make the decision of which to record. (3) The learner needs to make the connection between her experiences in the simulation environment and the representational objects that she manipulates in Belvedere. Hence the aggregated objects representing simulation runs should be created while still in the simulation environment and given visual identities recognizable to the learner, preferably by the learner herself. Representations that appeared suddenly in Belvedere would not have the richer contextual semantics provided by objects that she herself created and imported from the simulator.

The Simulation Interface already enabled the user to provide textual labels for simulation runs, and we took advantage of that. We modified the Simulation Interface to provide a facility for broadcasting labeled simulation summary objects to the MOO (and hence to the Belvedere

10 Jeremy Roschelle, personal communication. For ESCOT, see http://wise.sri.com/escot/.

in-box via the Belvedere-MOO Translator) thereby enabling the learner to select relevant results without leaving the simulation context.

The Role of the MOO

The MOO and MCP proved to be a convenient infrastructure for our experiments, providing a common "bus" into which we could plug our components without needing to design and implement low level communication mechanisms. However our experience suggests that more special purpose protocols and communication hubs will be necessary for robust and scalable Learning Spaces. A MOO has to multitask on many chores; adding the handling a large number of low-level interactions to its other tasks seems

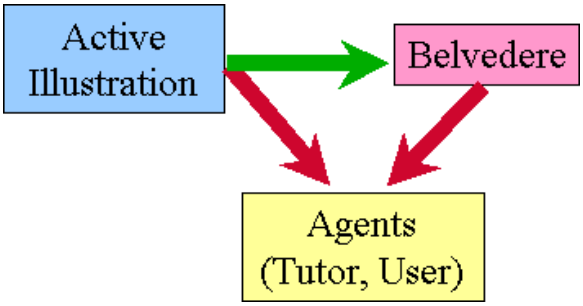


Figure 9. Retaining formal semantics versus accumulating contextual semantics.

suboptimal design choice. Also, the generic bus architecture approach burdens the components and their translators with the need to filter out irrelevant messages. Instead, special purpose communication hubs or "Resource Exchanges" where simulation, modeling, and coaching components can find each other as needed should be designed. Similarly, the MCP protocol was designed primarily for relatively low-bandwidth purposes, and so rich data streams (i.e., bitmaps) had to be communicated by other means. As more Learning Spaces are designed, the constraints on better protocols and resource exchanges should become clearer.

Summary of Component Communication Lessons Learned

Efforts towards achieving "plug and play" component-based software tend to address primarily the need for a common communication protocol or API. Where the semantic issues are addressed, the focus is usually on solving the problem of sharing information between the component applications. In other words, the goal is *to retain formal semantics as the interpreter changes.* In our demonstration, this was exemplified by the horizontal arrow of Figure 9, showing how simulation run and hypothesis events created in Active Illustration were moved to Belvedere, which also understood them as empirical and hypothetical statements

Standardized shared ontologies have been suggested as a solution. In this approach, information is represented in the same manner at both ends of the horizontal arrow of Figure 9: there is *one representation and two interpreters.* However, fundamental mismatches in components' internal representational needs will frustrate attempts to achieve "plug and play" interoperability between components by relying on standard ontologies alone. This is exemplified by Active Illustration's and Belvedere's needs to represent simulation-based experiments at entirely different granularities. With human intervention during the "plugging" or composition of a learning environment from components, *translators* can play a significant role in bridging between different representational requirements. The translation approach is greatly facilitated if components are designed to be inspectable, making semantically significant aspects of their state and internal communications visible to other components via an open architecture. In this approach, information is represented differently at the two ends of the horizontal arrow of Figure 9: there are *two representations and two interpreters.* The

builders of the learning environment identify the information that is made visible by one application and is needed in the other, and write lightweight software to mediate between the two. Standardized *shared vocabularies* may play a useful role in expediting the negotiation of semantics between developers of two particular educational software components.

However, there is another semantic issue that is not fully addressed by the solutions summarized above: the need for semantic entities to retain their identity for observers as they move between representations and/or components. Here the goal is to *accumulate contextual semantics for the same interpreter*. As an entity move between contexts, possibly changing its representational form but retaining its identity, it accumulates semantics gained by manipulating it in those contexts.

This need is depicted by the two diagonal arrows of Figure 9: there are *multiple representations (different contexts) and one interpreter.* It is exemplified in our demonstration in two ways: (1) by the Tutor Agent's need to know that an object being manipulated in Belvedere corresponds to activities and objects in Active Illustration; and (2) by the human user's need to know that an external (visual) representation that appears in Belvedere stands for concepts and experiments from Active Illustrations. Automatic translation (the horizontal arrow) alone could not solve this problem. We had to modify the components themselves, to ensure that (1) internal identity was preserved through the use of a common identifier for each object; and that (2) external identity (for the user) was preserved through the use of common textual labels for each object. In brief, accumulation of contextual semantics requires persistent identity, and this requires coordinated design of components.

Thus this example reveals limitations of a pure "plug and play" approach to component-based systems. Communication protocols cannot anticipate all future needs. Furthermore, they are not even sufficient as a solution for all presently known needs. The components themselves may need to be modified in a coordinated manner to achieve the proper educational experience.

Domain Generality of Science Learning Space Components

Active Illustrations, Belvedere and the Experimentation Tutor Agent were all designed to be relatively domain independent. Active Illustrations rely on self-explanatory simulators [Forbus & Falkenhainer, 1990, 1995], which can be automatically compiled for any domain for which there is a domain theory that can be expressed in Qualitative Process theory [Forbus, 1984]. In practice, this means domains that can be described in terms of systems of ordinary differential equations without simultaneities, where the set of relevant equations can change over time[11]. Belvedere can be used to represent any argument that involves the relationship of evidence with alternative hypotheses. As illustrated in Figure 8, Belvedere's rules are largely domain independent, relying only on basic concepts of hypothesis, data, and evidential relations. The Experimentation Tutor Agent can be used to monitor and guide students in experimentation involving selecting values of multiple independent variables, monitoring experimental results and forming hypotheses, making arguments related experimental results and hypotheses. As illustrated in the example working memory elements in Table 3, the Experimentation Tutor Agent's representation is not particular to any science domain.

11 Covering the full range of phenomena that need to be simulated in, for example, an entire K-12 science curriculum, will take additional technologies in addition to the kind of self-explanatory simulators used in our Active Illustrations architecture. Self-explanatory simulator interpreters for systems with simultaneities have been created [c.f. Amador et al, 1993; Iwasaki & Low, 1993], although they do require sophisticated on-line symbolic algebra capabilities and have not to our knowledge been shown to scale to large-scale simulators.

Table 3. General working memory representation in the Experimentation Tutor Agent instantiated for the atmosphere domain.

```
atmosphere-problem
   isa> experimentation-problem
   experiment-space> atmosphere-world
   understand-effects-on> "atmosphere temperature"
   consider-variables> ("manual control of cloud cover" "co2 in atmosphere"
      "cloud model" "earth's surface" "solar radiation" "atmosphere
      temperature")
   trials-performed> (trial-001 trial-002)
   current-trial> trial-002
   prob-status> ready-to-simulate-trial
trial-002
   isa> experimental-trial
   trial-number> 2
   independent-variable-values> (("manual control of cloud cover" 0.95)
      ("co2 in atmosphere" "current earth co2" ) ("cloud model"
      "default cloud model")  ("earth's surface" "current earth surface
      properties") ("solar radiation" "normal solar radiation") ("atmosphere
      temperature" "normal atmosphere temperature")
   dependent-variable> "atmosphere temperature"
   base-experiment> trial-001
   changes-to-base> ((“manual control of cloud cover” 0.95))
```

The slot names (e.g., trials-performed or independent-variable-values) are general to any science domain. It is only in the slot values that the particulars of a domain are encoded. These slot values are filled by the Tutor Agent's Translator component as it observes the messages sent between the simulation interface and the simulation engine.

As evidence of the mutual domain generality of Active Illustrations and the Experimentation Tutor Agent, we changed the science domain between our first Science Learning Space demonstration and the second one illustrated above. In contrast to the atmospheric domain illustrated above, the science content in the first demonstration concerned evaporation processes. In going from the first to second demonstration, the only major change was in the input to the Active Illustrations and the Experimentation Tutor Agent. Domain rules for atmospheric phenomena were expressed in Qualitative Process Theory and input to Active Illustrations. The simulator was generated automatically and all the interface and communication code was reused. In the case of the Experimentation Tutor Agent, except for some enhancements that applied to both domains, the same working memory and production rule representation was used in both demonstrations to model trace simulation activities. (Of course, to trace argumentation activities in Belvedere in the second demonstration, new working memory elements and productions were added.) A further demonstration of the domain generality of the Experimentation Tutor Agent was performed in a student project in which it was connected with a simulation of the factors contributing to pitch of guitar string.

Resolving Pedagogical Challenges through the Science Learning Space Architecture

The scenario presented above oversimplified some difficult pedagogical issues that must be addressed to scale up this approach. One such issue is how the Experimentation Tutor Agent infers students' intentions in constructing further experiments. In particular, when a student performs a third experiment we cannot know for sure whether her intention is to compare this experiment with the first, the second, or both prior experiments. Consider the student in our scenario performing a third experiment in which she now modifies both cloud cover and atmospheric pressure. In reference to the second experiment, this looks like the change-more-than-one-variable-bug. However, if the student changes the cloud cover back to 0.5 as it was in the first experiment, then this third experiment will only differ from the first in atmospheric pressure. If the student's intention was to compare this third experiment with the first, she has made a good move. However, if her intention was to compare it with the second experiment, she has not. This creates a problem for the Experimentation Tutor Agent, which has no information about the student's intentions.

Our solution to this problem involves making use of the third, Representation Constructor, component of a Science Learning Space. Essentially we postpone the ambiguity resolution to the argument stage. During the tracing of simulation activities, the Experimentation Tutor Agent gives the student the benefit of the doubt. In particular, the change-one-variable production rule was written so that the baseline experiment can match against any prior experiment. If a new experiment is created that is one variable different from *any* previous experiment, then the model tracer will accept the experiment as reasonable (i.e., change-one-variable will fire).

Of course it is quite possible that a student may create such an experiment by accident – as would be the case, above, if the student's intention was to compare the third experiment to the second. The possibility of such ambiguities is a serious problem in the context of the usual coached simulation ITS. An approach to ambiguities sometimes taken in Cognitive Tutors (e.g., the LISP tutor) is to query the student when a model tracing ambiguity appears (Anderson, Corbett, Koedinger, & Pelletier, 1995). The tutor might provide a menu of possible alternative intentions for the student to choose from, for instance, "I intend this experiment to be a comparison with a) experiment 1, b) experiment 2, or c) neither". Such menus can be disruptive to the student's planning process and may give too much away (on the other hand, properly designed, such menus may provide a useful form of instructional scaffolding for metacognitive reflection on the planning process).

The inclusion of a Representation Constructor component in a Science Learning Space, Belevedere in this case, can provide an alternative solution to this ambiguity problem. The argument construction task in Belvedere provides a good complement to the simulation environment in that it provides a way to assess and reveal any confusions students might have had in their intentions during experimentation in Active Illustrations. Consider the example above where the student's third experiment differs from her second by two variables, but is only one variable different from her first experiment. If the student's intention was to compare this third experiment to the first, this good intention will be revealed in the argument phase when the student argues for a hypothesis about the effects of atmospheric pressure (the one changed variable) using the first and third experiments. If the student does otherwise, for instance, by arguing for this hypothesis using the second and third experiments, then the tutor can assess this misunderstanding and remediate: "Since experiments 2 and 3 differ by both atmospheric pressure and cloud cover, you cannot be sure which caused the change in temperature."

Summary of Synergies among Science Learning Space Components

The Science Learning Space vision is an inclusionary vision. Going beyond the exclusionary mode of academic argumentation (i.e., approach X is wrong, approach Y is right), the Science Learning Space vision acknowledges that AI-based educational approaches that are often pitted as opposites, in fact, usually have complementary features. In combination such features are more powerful than any alone. The challenge is to make progress on component-based architectures and practices so as to lower the current barriers to taking advantage of complementary approaches.

We do not advocate arbitrary collections of components as if more is always better. Rather, we believe that specific kinds of components fit well together. In particular, the three categories of components of a Science Learning Space, namely simulations, representation constructors, and tutor agents, are particularly well suited for combination. As summarized in Table 4, each component category has strengths that compensate for weaknesses of the others.

In addition to these general characteristics of these component categories, we also identified a few specific synergies between the components in our feasibility demonstrations. Certain problems that would have been daunting given any two of the three components were made much simpler by the use of all three.

Table 4. Pros and Cons of Contrasting Approaches to Educational Software

	Pros	**Cons**
Simulations	Experience is more accessible (in time, place, and cost); idealizations guide interpretation of experience.	No explicit assistance or learning agenda. Limited representational support and expression of one's own ideas.
Representation Construction Tools	Affords powerful patterns of thinking; supports learners' expression and reflection.	No explicit assistance or learning agenda. Lack of richness of presented activities
Intelligent Tutor Agents	Provides just-in-time, context-sensitive assistance; assesses & selects appropriate activities	Limited representational support and expression of one's own ideas. Lack of richness of presented activities

One example of a daunting problem is creating Tutor Agent capabilities to interpret students' representation constructions. For instance, in typical uses of Belvedere (a representation constructor), students are allowed to enter hypothesis and evidence nodes in free-from English. Without sophisticated natural language understanding capabilities, the Belvedere Argumentation Advisor does not know the semantic content of these nodes. The addition of the simulation component, Active Illustrations in this case, provides an alternative to natural language understanding. As described above, the meaning is given to a node, evidence or hypothesis, through the context of its creation within the simulation environment. This contextually determined or indexical semantics can then be carried over into the Belvedere environment.

A second daunting problem is creating Tutor Agent capabilities to infer student intentions during simulation actions. For instance, as students create multiple experimental trials in Active Illustrations it becomes more difficult to infer student intentions with any level of certainty. Without interrupting the student with questions about their intentions, the Tutor Agent is left to just make best guesses. The addition of the representation constructor component, in this case Belvedere, provides an alternative solution. Although the Experimentation Tutor Agent still cannot be sure of students' intentions during the experimentation stage, in the process of argument construction students must make those intentions clear. Or, at least, students must make clear their reflections on their intentions since it is possible that a student realizes only at the argument stage what their intention should have been. In any case, such reflection and articulation provides an arguably powerful learning opportunity for students. At the same time, it provides the Tutor Agent with a more reliable view on student intentions that is not possible without all three components.

CONCLUSIONS

In this paper, we described a case study of component-based construction of a *Science Learning Space*, consisting of a simulation tool (Active Illustrations), a representational tool (Belvedere), and tutoring agents (the Experimentation Tutor Agent and Argumentation Advisor). We discussed several approaches to reducing the effort required to "hook up" diverse components

and demonstrated the value of sharing semantics between applications. Information objects created with particular semantic identities in Active Illustrations retained their identity in their treatment in Belvedere and its Argumentation Advisor. Furthermore, the Experimentation Tutor Agent treated these objects as having the same semantics in both situations.

The Science Learning Space vision is to combine the pedagogical benefits of simulations, representational tools, and intelligent assistance to support students in cycles of inquiry -- questioning, hypothesizing, modeling, reflecting and revising -- to both acquire new scientific content and to improve reasoning and learning skills. A major obstacle at this point is that too few developers are creating open components that are recordable and scriptable by other applications. Although *media* interoperability – copying and pasting between applications, and embedding of one medium inside another – is widely available between "office" tools in current software environments, our vision is of a *semantic* interoperability between knowledge-based software for learning. In this vision, learner-constructed objects will maintain their meaning (though not necessarily the same underlying representation) when moved from one tool to another. They will remain meaningful not only to the human user, but also to the software agents that interact with each tool. Consistent treatment of the learner's constructions in different contexts by different software agents reinforces the deep semantics that we want learners to extract and generalize from specific experiences. At the same time, the contextual semantics of these objects accumulate as they are used. In a Science Learning Space, students experience the concept of experimental trial, for instance, by first *thinking about it*, designing discriminating trials in the simulation, and then *thinking with it*, using these trials to construct an argument. Tutor agents support students in properly encoding these learning experiences and in engaging in effective scientific reasoning processes. We hope our initial efforts to integrate components in a Science Learning Space point the way to future, more complete efforts.

Acknowledgments

The DARPA CAETI program funded the Active Illustrations, Belvedere, and Tutor Agents projects. Thanks to Kirstie Bellman for providing inspiration for this collaboration, to Danny Bobrow and Mark Shirley for providing the MOO infrastructure, and to programming staff Ray Pelletier, Dan Jones, Leo Ureel, and Tamar Gutman. Thanks also to Bob Balzer, Brant Cheikes, Steve Ritter, and Jeremy Roschelle for pertinent discussions that influenced our thinking.

References

Amador, F., Finkelstein, A. and Weld, D. (1993). Real-time self-explanatory simulation. *Proceedings of AAAI-93*.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences, 4* (2) 167-207.

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

Case, R. (1974). Structures and strictures: Some functional limitations on the course of cognitive growth. *Cognitive Psychology, 6*, 544-573.

Chen, Z., & Klahr, D. (1999). All other things being equal: Children's acquisition of the Control of Variables Strategy. *Child Development. 70*, 1098-1120.

Collins, A. & Ferguson, W. (1993). Epistemic forms and epistemic games: Structures and strategies to guide inquiry. *Educational Psychologist,28(1)*,25-42.

de Jong, T., & van Joolingen, W.R. (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research, 68*, 179-202.

Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence, 24*, 85-168

Forbus, K. (1997). Using qualitative physics to create articulate educational software. *IEEE Expert*, 12(3).

Forbus, K. and Falkenhainer, B. (1990.) Self-explanatory simulations: An integration of qualitative and quantitative knowledge, Proceedings of AAAI-90.

Forbus, K. and Falkenhainer, B. (1995.) Scaling up self-explanatory simulators: Polynomial-time compilation. *Proceedings of IJCAI-95*, Montreal, Canada.

Goldenberg, E. P. (1995). Multiple representations: A vehicle for understanding understanding. In *D. Perkins, J. Schwartz, M. West, & M. Wiske (Eds) Software Goes to School: Teaching for Understanding with New Technologies,* pp. 155-171. New York: Oxford University Press.

Iwasaki, Y. & Low, C. (1993). Model generation and simulation of device behavior with continuous and discrete changes. *Intelligent Systems Engineering, 1*(2).

Kaput, J. (1995). Creating cybernetic and psychological ramps from the concrete to the abstract: Examples from multiplicative structures. In *D. Perkins, J. Schwartz, M. West, & M. Wiske (Eds) Software Goes to School: Teaching for Understanding with New Technologies,* pp. 130-154. New York: Oxford University Press.

Koedinger, K.R. (1991). On the design of novel notations and actions to facilitate thinking and learning. In *Proceedings of the International Conference on the Learning Sciences*, (pp. 266-273). Charlottesville, VA: Association for the Advancement of Computing in Education.

Koedinger, K. R., Anderson, J.R., Hadley, W.H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education, 8*, 30-43.

Kuhn, D., Amsel, E. D., & O'Loughlin, M. (1988). *The development of scientific thinking skills*. Orlando, Fl: Academic Press.

Larkin, J. H. & Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science 11(1):* 65-99.

O'Day, V., Bobrow, D.G., Bobrow, K., Shirley, M., Hughes, B, & Walters, J. (1998). Moving practice: From classrooms to MOO rooms. *CSCW Journal, 7 (1-2)*, 9-45.

Paolucci, M., Suthers, D., & Weiner, A. (1996). Automated advice-giving strategies for scientific inquiry. *Intelligent Tutoring Systems, 3rd International Conference,* Montreal, June 12-14, 1996.

Ranney, M., Schank, P., Hoadley, C., & Neff, J. (1994, October 16). *"I Know One When I See One": How Much Do Hypotheses Differ from Evidence?* Paper presented at the 5th ASIS SIG/CR Classification Research Workshop, Alexandria, VA.

Reiser, B. J., Beekelaar, R., Tyle, A., & Merrill, D. C. (1991). GIL: Scaffolding learning to program with reasoning-congruent representations. In *Proceedings of the International Conference on the Learning Sciences*, (pp. 382-388). Charlottesville, VA: Association for the Advancement of Computing in Education.

Reusser, K. (1993). Tutoring systems and pedagogical theory: Representational tools for understanding, planning, and reflection in problem solving. In S. P. Lajoie & S. J. Derry (Eds.), *Computers as Cognitive Tools* (pp. 143-177). Hillsdale, NJ: Lawrence Erlbaum Associates.

Ritter, S. (1997). Communication, cooperation and competition among multiple tutor agents. In du Boulay, B. and Mizoguchi, R. (Eds). Artificial Intelligence in Education: Knowledge and media in learning systems (pp. 31-38). Amsterdam: IOS Press.

Ritter, S. & Koedinger, K. R. (1997). An architecture for plug-in tutoring agents. In *Journal of Artificial Intelligence in Education, 7* (3/4), 315-347. Charlottesville, VA: Association for the Advancement of Computing in Education.

Roschelle, J. & Kaput, J. (1995). Educational software architecture and systemic impact: The promise of component software. Presented at *AERA Annual Meeting,* San Francisco, April 19.

Roschelle, J. (1996). Designing for cognitive communication: Epistemic fidelity or mediating collaborating inquiry. In D. L. Day & D. K. Kovacs (Eds.), *Computers, Communication & Mental Models* (Vol. 13-25,). London: Taylor & Francis.

Roschelle, J., Kaput, J., Stroup, W., & Kahn, T. M. (1998). Scaleable integration of educational software: Exploring the promise of component architectures. *Journal of Interactive Media in Education, 1998*(6). [http://www-jime.open.ac.uk/98/6]

Suthers, D. D. (1999). Representational Support for Collaborative Inquiry. *Proceedings of the 32nd Hawai'i International Conference on the System Sciences (HICSS-32)*, January 5-8, 1999, Maui, Hawai'i (CD-ROM), Institute of Electrical and Electronics Engineers, Inc. (IEEE).

Suthers, D., Connelly, J., Lesgold, A., Paolucci, M., Toth, E.E., Toth, J., & Weiner, A. (in press). Representational and advisory guidance for students learning scientific inquiry. Submitted to AAAI Press for a book on Artificial Intelligence and Education, K. Forbus and P. Feltovich, Eds.

Suthers, D. & Jones, D. (1997). An architecture for intelligent collaborative educational systems. *AI-Ed 97, the 8th World Conference on Artificial Intelligence in Education,* Kobe Japan, August 20-22, 1997.

Suthers, D., Toth, E., and Weiner, A. (1997). An Integrated Approach to Implementing Collaborative Inquiry in the Classroom. *Computer Supported Collaborative Learning (CSCL'97)*, Toronto, December, 1997.