Component-Based Construction of a Science Learning Space

Kenneth R. Koedinger, Daniel D. Suthers,[1] and Kenneth D. Forbus

Human-Computer Interaction Institute, Carnegie Mellon University
Information and Computer Sciences, University of Hawaii
Institute for the Learning Sciences, Northwestern University
E-mail: koedinger@cs.cmu.edu, suthers@hawaii.edu, 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) lab/field simulations in which experiments are run and data is collected, 2) modeling/construction tools in which data representations are created, analyzed and presented, and 3) tutor agents that provide just-in-time assistance in higher order skills like experimental strategy, representational tool choice, conjecturing, and argument. We believe that achieving this ambitious 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 lab simulation environment, 2) the Belvedere argumentation environment, and 3) a model-tracing Experimentation Tutor Agent. We illustrate student interaction in this 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 are 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, focus on scientific process
Activity	Inquiry process hampered by mundane procedure & long waits	Simulations speed time, leave technique lessons for later
Tools	Paper and pencil	Data representation & argument construction
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

1 Work performed while at Learning Research and Development Center, University of Pittsburgh

A Science Learning Space can be created by coordinating software components of three types: 1) *lab/field simulations* in which experiments are run and data is collected, 2) *modeling/construction tools* in which data representations are created, analyzed and presented, and 3) *tutor agents* that provide just-in-time assistance in 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 authoring 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 is for research labs 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 [Ritter & Koedinger, 1997; Roschelle & Kaput, 1995; Suthers & Jones, 1997]. This has led to a number of successful workshops on the topic (e.g., http://advlearn.lrdc.pitt.edu/its-arch/), the emergence of several standards efforts specifically targeted to advanced educational technology (e.g., www.manta.ieee.org/p1484/), and new repositories for educational object components (e.g., trp.research.apple.com). These efforts gain leverage from the rise of interactive Web technology and its associated emphasis on standards-based interoperability. Solutions for component-based systems are arriving, in the form of shared communication protocols, markup languages, and metadata formats. 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 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 Coach [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 ARPA’s Computer Aided Education and Training Initiative program contractors. Using a MOO as a communication infrastructure, 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 lab/field simulation component, Belvedere as a modeling/construction tool, and the Experimentation Tutor Agent and Argumentation Coach 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 purported economic and engineering benefits. Component-based systems are 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 their own knowledge through experience. 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, modeling of expert behavior, 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, tutoring systems are themselves subject to the criticism. Emphasis on knowledge engineering usually leaves little time for careful design of performance tools to enhance pedagogical goals. Thus, there is a third need for representational tools for manipulating data, searching for patterns, or articulating and testing new knowledge. Spreadsheets, outliners, graphers, and other such tools provide representational guidance that help learners see certain patterns, express certain abstractions in concrete form, and discover new relationships. Representational tools can be designed based on cognitive analysis to address particular learning objectives [Koedinger, 1991; Reiser *et al.*, 1991] and can function as "epistemic forms" [Collins & Ferguson, 1993] that afford desirable knowledge-building interactions. 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 kind of guidance is needed.

We believe that the ability to routinely synthesize new intelligent learning environments from off-the-shelf components that combine multiple functionalities rarely found today 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 which can require substantial extra work.

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 the basic distinction between empirical observations and theoretical statements. 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.

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 the Science Learning Space to record, test, revise and argue for this hypothesis.

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 Learning Space demonstration involving an Experimentation Tutor Agent and Active Illustration [Forbus, 1997] communicating through a Lambda-MOO derivative using the "MOO Communications Protocol". Forbus had already made use of the MOO for communication between the Active Illustration simulation engine and a simulation user interface (bottom 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 (small box upper 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.

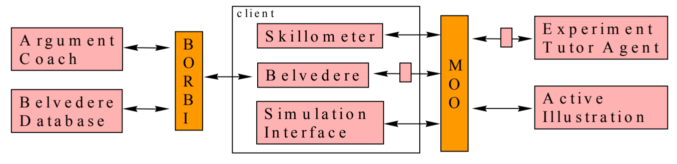


Figure 1. Communication architecture used in the demonstration.

Subsequently, the Belvedere system and Argumentation Coach were added (left side of Figure 1). Belvedere itself provides a communication architecture, described in Suthers & Jones [1997] and abstracted as "BORBI" in Figure 1. Integration of the Belvedere subsystem into the MOO required the addition of one translator component (the other small box 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 placed in the user's "in-box" for consideration.

Learning Scenario

Returning to our student who is thinking about planetary climate, we illustrate a learning interaction scenario supported by our multi-application Learning Space. 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, Belvedere's Argumentation Coach suggests that the student find some evidence for this hypothesis (Figure 3). 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.

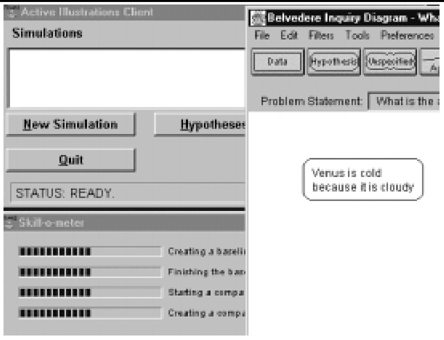
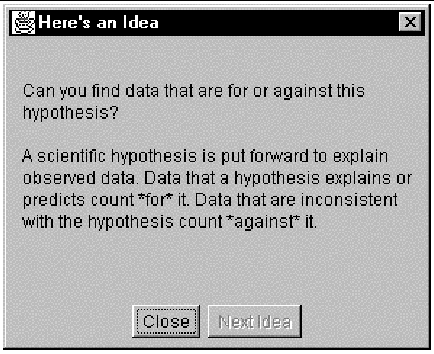


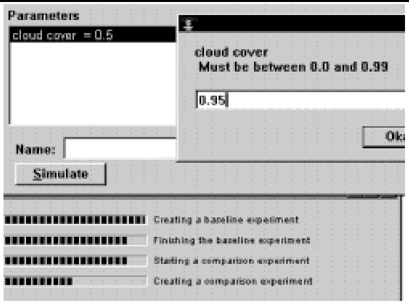
Figure 2. Opening situation: An initial hypothesis.


Here's an Idea

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.

Figure 3. Coaching scientific argumentation


cloud cover
Must be between 0.0 and 0.99

Parameters
cloud cover = 0.5
0.95
Ok
Name:
Simulate
Creating a baseline experiment
Finishing the baseline experiment
Starting a comparison experiment
Creating a comparison experiment

Figure 4. Creating an experimental run of the simulation.

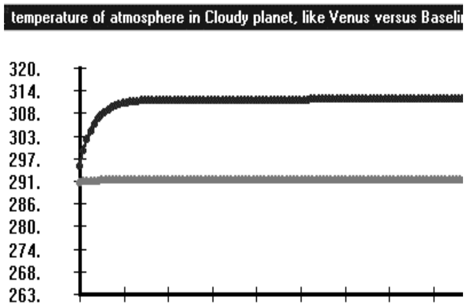
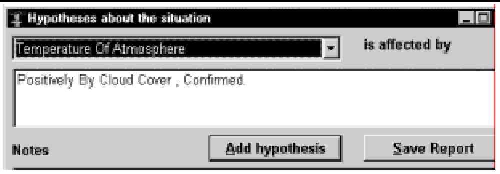


Figure 5. Plotting the results of the experiment.



Positively By Cloud Cover , Confirmed.

Figure 6. Making a new hypothesis in Active Illustration.

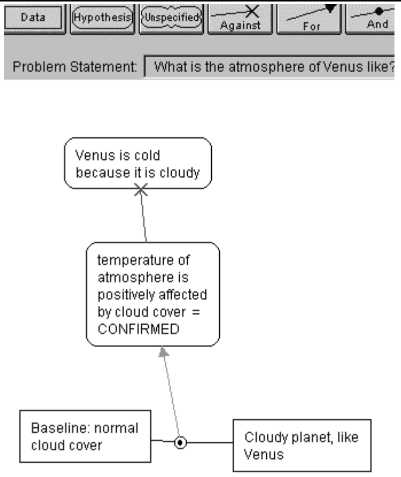

Data
Hypothesis
Unspecified
Against
For
And
Problem Statement:
What is the atmosphere of Venus like?
Venus is cold because it is cloudy
temperature of atmosphere is positively affected by cloud cover = CONFIRMED
Baseline: normal cloud cover
Cloudy planet, like Venus

Figure 7. Evidence map with experimental results.

In the Active Illustration interface, the student runs a baseline simulation using normal cloud cover parameters for Earth. The Tutor Agent observes these events in the MOO and updates estimates of baseline experimentation skills (lower portion of Figure 4). Then the student constructs a comparison simulation by increasing the Earth's cloud cover to 95%, much more like Venus (Figure 4). If the student had attempted to change more than one parameter of the simulation, the Experimentation Tutor Agent would recognize this action as an instance of the change-more-than-one-variable bug (see Table 2), and would have generated a feedback message to prompt the student toward designing a more discriminating experiment. Having created baseline and comparison experiments, the student uses Active Illustration's plotting facility to show the results (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 (Figure 6).

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. It filters messages and translates hypothesis and data objects into Belvedere's representations and places them in Belvedere's "in-box". 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 Figure 7. The two experiments are connected with an "and" link and then connected to the hypothesis that temperature increases with cloud cover using an evidential "for" link. Finally, this experimentally derived hypothesis is used to argue against the original idea that Venus is cold because it is cloudy by making an evidential "against" link.

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. As a consequence, this agent 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 agent can remediate this argumentation bug (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."

Table 2. Domain-Independent Productions for Experiments 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
```

Semantic Interoperability for Constructive Learning Interactions

Since all three systems 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 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. Significant portions of our communications 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 even reference vocabulary were available and known to all.

However, 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). Furthermore, 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 with, rather than about, experiments.

Translators to Preserve Advantages of Alternative Representations

The use of Translator components 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 symbol structures in a shared ontology. 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 client-server architecture. These tools were “recordable” and “scriptable” by the Tutor Agent [Ritter & Koedinger, 1997] and by each other. Unfortunately, too few office applications or educational objects are currently built in this open architecture.

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 coordinate 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 need for persistence of identity can sometimes be solved by addressing the learner's same need.

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 the in-box with many not so useful objects. (2) We wanted to encourage the learner to reflect on which 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 learner herself.

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 in-box) thereby enabling the learner to select relevant results without leaving the simulation context. This example reveals one limitation of a pure “plug and play” approach to component based systems: Communication protocols cannot anticipate all future needs.

Conclusions

We described a case study of component-based construction of a *Science Learning Space*, consisting of a simulation tool (Active Illustrations), a modeling tool (Belvedere), and tutoring agents (the Experimentation Tutor Agent and Argumentation Coach). 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 Coach. 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, modeling 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 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.

References

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.

Collins, A. & Ferguson, W. (1993). Epistemic Forms and Epistemic Games: Structures and Strategies to Guide Inquiry. *Educational Psychologist,*28(1),25-42.

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

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

Forbus, K. & Falkenhainer, B. (1995.) Scaling up Self-Explanatory Simulators: Polynomial-time Compilation. Proceedings of IJCAI-95, Montreal, Canada.

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: AACE.

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.

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.

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: AACE.

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, 1995.

Suthers, D. & Jones, D. (1997). An architecture for intelligent collaborative educational systems. *AI-Ed 97, the 8th World Conf. 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.