A Fully-Integrated Approach to Authoring Learning Environments: Case Studies and Lessons Learned

Menachem "Kemi" Jona & Alex Kass

The Institute for the Learning Sciences
Northwestern University
1890 Maple Avenue
Evanston, IL 60201
jona@ils.nwu.edu, kass@ils.nwu.edu

Abstract

While the idea of using modular, interchangeable components in building authoring tools for Intelligent Tutoring Systems sounds quite appealing from a software engineering perspective, we're skeptical about the educational validity of this idea, and of the implicit model of learning which underlies it. The mix and match approach is not theoretically neutral with regard to the questions of what really drives learning, and what are the central features of a learning environment. In this paper, we present an alternative approach to authoring learning environments, present 10 case studies of applications built using our authoring tools, and conclude with some reflections on lessons learned using our approach to date.

Problems with a component approach to building ITS tools

Traditional Intelligent Tutoring Systems (ITSs) are typically constructed out of four primary components or modules: the user interface, expert model, student model, and instructional module. It is not surprising, then, that authoring tools for constructing these tutoring systems tend to consist of specialized editors for building each component (*i.e.*, a user interface builder, expert model editor, etc.). Recently, researchers have begun to investigate whether these component editors could be developed into stand-alone tools, with the goal of then being able to mix-and-match the best tools for building each component of an ITS.

However, while the idea of modular, interchangeable components sounds quite appealing from a software engineering perspective, we're skeptical about the educational validity of this idea, and of the implicit model of learning which underlies it. The mix and match approach is not theoretically neutral with regard to the questions of what really drives learning, and what are the central features of a learning environment. For example, the approach generally assumes that the central component of a learning environment is the tutor, and that the critical learning events are interactions between the learner and the tutor. This view is not compatible with what many who study education and human learning have found. For example, many progressive educators would argue that the most important aspect of a learning environment is a complex, realistic activity in which the learner becomes engaged, and *not* the tutoring received. Under this view there may still be an important role for computer-based tutors in computer-based learning environments, but it is a secondary role; the crucial interaction is between the student and some sort of simulation, or task environment, rather than the interaction with the tutor. Furthermore, the structure of interaction with tutoring is fundamentally driven by the structure of the task environment or simulation.

For this reason we feel it is misguided to focus narrowly tutoring, student modeling, or user interface *per se* when thinking about tools for authoring learning environments. It is arguable whether it is even possible to do a needs a analysis for, say, a tutor, or a tutor-building tool, absent a broader picture of the context in which the tutor will be operating. For example, here are some examples of the crucial questions which contribute to the effectiveness of a learning environment, which aren't part of the design of a tutoring module, *per se*, the answers to which help place key constraints on what the tutor should look like:

- What type of goal will the student be pursuing?
- What kinds of activities will the student be engaged in?
- What special types of hard issues will the activities present, which will make achieving the goal a difficult challenge?
- What types of failures will the student who does not fully understand the subject matter experience?
- Why kinds of questions will the activities in the learning environment raise in the student's mind?

An integrated approach to tool building

We have adopted an alternative methodology for building computer-based tutoring systems that entails creating a fully designed and implemented teaching architecture along with a special-purpose tool for instantiating that architecture in a variety of domains. A teaching architecture (Schank 1994), in the sense used here, means a system that includes:

- a task for the student to perform and a simulation or

other environment in which to perform that task;
- a representation of the student's task that the system can use to understand what the student is doing;
- a set of teaching strategies or interventions tailored to the student's task; and
- a user interface that allows the student to engage in the task, access whatever informational resources are required during the task, and receive guidance.

In other words, the architecture is a complete tutoring system, but one in which all the specifics of the domain to be taught are isolated and thus can be replaced by other domain content. What makes this approach different than the traditional approach described above, is that the structure of the learner's task within each architecture is fixed and cannot be changed. The predetermined nature of the learner's task allows for the creation of special-purpose authoring tools for specifying the details of the task and customizing the user interface. In order to preserve the pedagogical integrity of the system, the tools permit only limited modifications; in essence, the author is instantiating a framework (or teaching architecture) with particular subject matter.

This approach has a number of advantages, argued for in previous work (Jona & Korcuska 1996). One disadvantage, relative to more general purpose authoring tools is a lack of flexibility in creating a wide range of teaching systems. In recognition of this, a goal of our research program at the Institute for the Learning Sciences (ILS) is to create a suite of eight such teaching architectures (Schank & Korcuska 1996) that will provide a variety of approaches to teaching suitable for a wide range of subject matter. We now have four of the eight planned architectures operational and in use, with a fifth currently in active development. The five architectures are:

1. **Advise**. The student plays the role of an advisor to an important decision maker and must evaluate a set of plans and recommend one that does best on a specified set of goals.
2. **Investigate & Decide (INDIE).** The student must investigate a phenomenon in detail, performing and interpreting lab tests, and make a decision based on the results of his or her investigation.
3. **Run**. The student must manage a complex system or group and respond to unforeseen events that threaten the stability of the system.
4. **Script.** The student must learn to correctly perform a procedural task under a variety of conditions, and to recover appropriately from any errors made.
5. **Persuade** (in development). The student plays a role in a social simulation and must understand the other characters' goals and agendas in order to persuade them change their positions or reach consensus on an issue.

In the remainder of this paper we seek to illustrate this approach to learning environment development tools by briefly describe each tool we are actively using, along with a couple example applications we have built with that tool. We conclude with some reflections on the lessons we have learned about our approach to authoring tool development, and about developing learning environments in general.

Ten example applications built using the ILS tool set

This section is divided into five subsections—one for each tool—and provides a short description of two of the applications built using each tool. In total, we describe ten applications across a wide range of topics, that have been built using our approach.

The Advise Architecture

In an Advise application the student plays the role of an advisor to an important decision maker and is asked to create a report evaluating several alternative solutions to a given problem. For example, the student may be placed in the role of an advisor to President Kennedy during the Cuban missile crisis and be asked to evaluate several options for responding to the Soviet threat. To help complete the task, the student is given a large hypermedia database of information and expert opinions about the issue, a skeletal report, and a panel of opinionated experts who represent conflicting perspectives about the issue. Upon submitting the report for review, the student receives a critique which focuses on the quality of the evidence she used to support their conclusions. The student can then revise and resubmit the report. The Advise architecture is described in more detail in (Korcuska forthcoming); (Korcuska, Herman, & Jona 1996); and (Korcuska, Kass, & Jona 1996).

Crisis in Krasnovia

In this system, the student plays a top advisor to the President of the United States, who asks the student to review and evaluate possible U.S. responses to the situation in Krasnovia, a fictional country based on the former Yugoslavia. The student can call on a panel of advisors to help with the evaluation. These advisors—whose policy preferences run the gamut from military intervention to diplomacy—offer their opinions buttressed with evidence from a video database that includes interviews with real experts in history and foreign policy as well as information and footage about past foreign policy problems. The student evaluates the options by asking questions of the advisors, and consulting with the experts in the video database. Finally, the student constructs a report to the President outlining his or her conclusions, with supporting evidence from the video database. The President critiques the report based on how thoroughly the student has documented his or her opinions, how consistent the report is, and how well it articulates the administration's foreign policy goals.

Emerging Economies

Emerging Economies, developing in collaboration with the Kellogg School of Management, is a learning environment for business school students and executives seeking to master the intricacies of doing business in an emerging economy. Emerging economies are those countries around the world that seem poised for explosive growth--for example, China, Brazil, some areas in Eastern Europe, and countries in South Asia. These are the places where every business that wants to grow wants to be.

The problem is that business success in these unevenly developed, changeable, culturally distant markets is by no means assured. Without experience it is all too easy to make fatal miss-step, and people with experience in these markets are in short supply. The Emerging Economies system provides an environment in which students are set the task of advising a fictional company's CEO on how best to take the company into a fictional emerging economy (specifically, a baby-food company that is plotting its strategy for entering Eastern Europe). In the role of consultant, the only way to prepare a complete report for this CEO is to learn from the experts about how other companies have fared in other markets.

The Investigate & Decide (INDIE) Architecture

In applications built using the Investigate and Decide (INDIE) architecture, students must make a decision about some issue, and in order to make this decision are given a set of investigative tools to collect the necessary evidence to back up their decision. To represent their decision, students' typically prepare a structured report and must back up whatever decision they've reached with evidence collected from running tests or doing other types of investigative actions. (Bell 1996) discusses the INDIE architecture at length.

Immunology Consultant

Built in conjunction with Northwestern University's medical school, Immunology Consultant is a goal-based scenario (GBS) designed to teach the basic physiology and pathophysiology of the immune system to second year medical students. A student plays the role of a consultant to a generalist physician, who is dealing with a puzzling case that might have an immunological etiology. The student's task is not simply to provide a diagnosis, but rather to explain the underlying causes of the patient's illness to the physician. During an exploratory process that may take several hours, the student may ask questions of the patient, request laboratory tests, and consult a large, well-structured database of expert knowledge in the form of video clips, text, and graphics. We have just completed a prototype version of the system that contains a single case. After testing the system with students, we hope both to use it in classes at Northwestern's medical school and to seek additional funding to expand the content to create a complete, computer-based course in immunology.

Is it a Rembrandt?

Recently, the art community has come to question the authenticity of many works previously ascribed to Rembrandt. Attribution is no easy task. Aside from the obvious issue of forgeries and unsigned works, there are also issues related to Rembrandt's workshop model of painting: for a painting done by both student and master, is it a Rembrandt? To answer this question, scholars examine features of a painting such as subject, composition, and palette; they review conservation records; and they perform scientific tests such as X-ray analysis, dendrochronology, autoradiography, and pigment analysis. The "Is it a Rembrandt?" GBS, built in collaboration with Northwestern University's Art History Department, introduces the students to these issues by having the student determine the authenticity of three Rembrandtesque paintings. The student learns how to examine a painting to determine attribution, using evidence from both connoisseurship and technical analysis. In this process, the student encounters general problems about attribution, and learns more about both painting in general and Rembrandt in particular.

The Run Architecture

The Run architecture places the student in the role of managing or controlling a system, process, organization, or team to achieve a desired outcome or end state. The architecture provides a mechanism for simulating the dynamic interrelationships that exist in a particular domain, and it is through exploring these interrelationships that students' come to learn the relevant aspects of the domain. Throughout the course of the simulation, students are confronted with a variety of challenges and unexpected situations that they must address or resolve. At each decision point the student has the opportunity to hear from experts in the domain, and can use their advice to inform his or her decision about what course of action to pursue.

Fire Commander

Fire Commander is a educational software program designed to teach 6th-8th graders about fire safety and firefighting by putting them in the role of incident commander at the scene of a house fire. The student is responsible for directing the firefighting teams to put out the fires and save the lives of the people at the house. After the student gives firefighters instructions, he will see video of the firefighters carrying out those instructions, as well as video of everything else that happens in the world at that time. At any time, the student can ask questions of firefighters about what just happened, what to do, or the current state of the world.

Zookeeper

The goal of the system is to teach 3rd to 5th graders about zoo diet/nutrition, zoo habitat, and zoo care for a number of different animals. The student plays the role of a zookeeper who must complete everyday tasks as well as assess and solve unexpected zoo-related problems. The student is presented with problems which need to be solved for various animals at the zoo. These problems are things that go wrong at the zoo (*e.g.* the gorillas are

fighting). The student needs to prioritize the problems and decide in which order the problems need to be addressed. Each problem has 3-4 actions which can be chosen to solve that problem. For every problem and the associated actions, the student can ask questions as an aid in determining which is the best solution for a given problem. As the student ignores problems or chooses incorrect actions, existing problems will worsen and/or new problems will be created.

The Script Architecture

The Script architecture is a descendant of the Moped tool developed by (Ohmaye 1992) and extended by (Jona 1995) and (Guralnick 1996). In applications built with the Script architecture, the student engages in a simulated task that is highly scripted or proceduralized. A tutor observes the student's progress through the procedural task and intervenes to point out incorrect or missed steps. The student can also query the tutor at any point on what to do next or why a particular step is correct or incorrect.

ChemLab

To truly understand chemistry, a student must work in the laboratory. Unfortunately, lab time is a precious commodity, and much too often, students become overwhelmed by the procedure and fail to understand the underlying theory. When complete, the "Chemlab" GBS will allow students to spend unlimited time learning both procedure and theory in a virtual laboratory. The student will learn how to perform experiments such as Thin-Layer Chromotography in simulation. Expert chemists will be available to help, and the student will be permitted to try things which in the real laboratory would mean rushing to the emergency shower (or worse). Built in collaboration with Northwestern University's Chemistry Department, this GBS will serve as auxiliary material for organic chemistry students who will first become familiar with the procedures in simulation, and then truly explore *chemistry* in the lab.

411

Built for the purpose of training directory assistance operators, *411* provides 50 different situations that operators would likely encounter on the job. Through practice, trainees learn how to answer many different types of calls with speed and accuracy. *411* provides just-in-time tutoring; that is, key information and instruction are always available, just when trainees need them. Other advantages of *411* are that employees become productive faster and that the company does not need as many instructors, both of which reduce training costs. The company anticipates that the software will reduce a training session from five days to two days.

The Persuade Architecture

Applications built using the Persuade architecture put students in a social simulation in which they interact with simulated characters and seek to persuade them to change their positions on an issue, or build consensus among them on a set of issues. The student has the option to explore any approach they may wish to take by consulting a richly indexed database of experts.

French Revolution

French Revolution is a computer-based learning environment for undergraduate French majors studying the cultural history of France. The goal of the system is to familiarize these students with the social situation prevailing in France just prior to their revolution. The approach this system takes is to cast the student as a member of the 3rd estate delegation to the Estates General, taking on the mission of recruiting other delegates to join the nascent National Assembly. Transported to a computer-simulated Versailles garden, the student converses with members of many different social strata of French society. In order to succeed at convincing these simulated characters to join the Assembly, the student must learn about the characters' background and interests. By the time students have met and talked with all the available characters, they will be conversant in the issues current on the eve of France's revolution.

EPA Community Partnering

In collaboration with the U.S. Environmental Protection Agency (EPA) we are developing a computer-based learning environment called Community Partnering for Environmental Results. The goal of the project is to build a simulation-based system that can be used by a broad range of EPA staff to practice and refine public outreach and community relations skills. In the system the learner must field calls, run meetings, seek input, and manage the community relations (including relations with state and local agencies) for various simulated EPA initiatives or projects.

In the first scenario, for example, the learner plays the role of an EPA coordinator for the fictional community of Evans Bay. The learner is introduced to come of the environmental issues be confronted in Evans Bay via news videos and background documents and reports. After reviewing this background material, the learner's first task is to conduct a "Question & Answer" meeting to identify the public's concerns. Conducting this meeting entails interacting with simulated audience members (who represent a range of personality types and interest groups commonly found in public meetings). Through the use of advanced blue-screen video technology, simulated audience members appear to be speaking directly to the learner, heightening the realism of the simulated meeting. The scenario is designed to make the meeting quite challenging; some members of the audience can be quite unpleasant. Moreover, a response that makes one audience member happy may cause another to become upset. If the learner exhibits a lack of preparation, insensitivity to the public's feelings, poor judgment, or other common problems, the audience will become even more hostile. After the simulated meeting concludes, the student can see the results of his or her decisions via simulated public opinion polls and press reports of the meeting.

Lessons Learned

While still relatively new, we have learned some interesting lessons about our approach to authoring tool development. Some of the key successes and "lessons learned" of our approach include:

- Overall development time for typical applications using our existing teaching architectures has been reduced from around 12 months to 6-8 months, depending on the complexity of the application.

- We have dramatically reduced the need for programmers on our project teams. In the past, applications required 2-3 programmers for the duration of the project. Now we need only about 0.25 FTE programmers.

- We have cut development costs for average projects by around 75%.

- Access to subject matter experts is now the rate limiting factor in completing projects. Prior to development of the teaching architectures and tools, programming and interface implementation were the rate limters.

- The tools allow for more rapid prototyping and permit earlier opportunities to "test" the system under development and refine the content in the system. This allows us to notice problems and gaps in the content much earlier than was possible before the tools were developed.

- Given the scaffolding provided by the design embodied in the teaching architectures, the design task has been eased, especially in the early stages of project development.

- However, even despite the design scaffolding afforded by the teaching architectures, skilled design oversight is still required to insure that a high quality learning environment is produced. We had initially hoped that the tools would enable designers with less specialized design training to build consistently high quality software. This hope has not been realized to the extent we had thought it would.

- While we have achieved significant reductions in the time needed to develop new applications, we have not been able to reduce it as much as initially thought. Our goal was to reduce development time for typical projects to around 4 months. The limited availability of subject matter experts with whom we are collaborating and difficulty in coordinating their schedules with that of the development team has been a key factor in keeping the development time longer than we were shooting for.

References

Bell, B. 1996. *A special purpose architecture for building educational software.* Unpublished Ph.D. dissertation, Northwestern University.

Bell, B., & Korcuska, M. 1995. The Goal-Based Scenario Builder: Experiences with Novice Instructional Designers. Paper presented at *American Educational Research Association Meetings*, San Francisco, CA.

Bell, B.L., Kedar, S.T. 1995. When Less is More: Supporting Authoring and Interface Building via Special-Purpose Task Models. In Proceedings *7th World Conference on Artificial Intelligence and Education*, Washington, DC., pp. 533-540.

Guralnick, D. 1996. *Training systems for script-based tasks*. Ph.D. dissertaton, The Institute for the Learning Sciences, Northwestern University.

Jona, M. 1995. *Representing and Applying Teaching Strategies in Computer-Based Learning-By-Doing Tutors.* Ph.D. dissertaton, The Institute for the Learning Sciences, Northwestern University.

Jona, M., & Korcuska, M. 1996. Same Architecture New Domain (SAND): An Alternative Methodology for Building High-Quality Reusable Tutoring Systems. *Proceedings Third Conference on Intelligent Tutoring Systems,* Montreal, CA.

Korcuska, M. (1998). *Knowledge-based Authoring Tools for Educational Software.* Unpublished Ph.D. dissertation, Northwestern University. In preparation.

Korcuska, M., Herman, J., & Jona, M. 1996. Evidence-based Reporting. In P. Carlson & F. Makedon (Ed.), *Educational Multimedia and Hypermedia*, (pp. 783). Boston, Ma: Association for the Advancement of Computing in Education.

Korcuska, M., Kass, A., & Jona, M. 1996. Matching teaching goals to learning-by-doing tasks: When to use EBR. *Proceedings Second International Conference on the Learning Sciences*, June 1996.

Ohmaye, E. 1992. *Role playing and social simulation.* Ph.D. dissertation, The Institute for the Learning Sciences, Northwestern University.

Schank, R. & Korcuska, M. 1996. *Eight GBS Tools.* Technical Report #67. The Institute for the Learning Sciences, Northwestern University.

Schank, R.C. 1994. *Teaching Architectures.* Technical Report #3, The Institute for the Learning Sciences, Northwestern University.