The Apprenticeship Learning Object Toolkit: A Generalized Architecture for a Family of Computer Tutoring Systems

Michael C. Dorneich
Patricia M. Jones

Department of Mechanical and Industrial Engineering
University of Illinois at Urbana – Champaign
1206 W. Green St.; Urbana, IL 61801

ABSTRACT

The emergence of computer technology has had profound implications for instruction. Recent efforts in the field have shifted the focus from knowledge–as–a–product to learning–as–a–process. Computer–supported collaborative learning (CSCL) focuses on learning as a collaborative activity that is situated in its environment. This paper describes the Apprenticeship Learning Object Toolkit (ALOT), a generalized architecture, that forms the basis of a family of computer tutoring systems. The system, based on previous work[2]–[6], utilizes the apprenticeship model of learning to create a learning environment for the study of operational procedures of experiments in the physical sciences. Examples include nuclear magnetic resonance (NMR) spectroscopy or x-ray diffractometry (XRD). The development of a learning environment is driven by the goal of linking theoretical knowledge with practical operational experience. Active, exploratory, apprentice–style learning is supported via modes of operation within the system. The student can flexibly choose to "observe the expert" perform and explain operational steps, or "act as an apprentice" and carry out the steps autonomously. The student can switch between these modes at their discretion, giving the student control of the level of intervention by the system. In addition, the student can explore and reflect on an "information space" of object designations, procedures, and related concepts. Additionally, the BAUEN (Basic AUthoring ENvironment) is introduced. BAUEN is a graphical user interface system that is intended to help the designer author a computer tutorial system with maximum efficiency, focusing the developer's efforts on domain content, and not on system implementation. The tutoring systems created via this architecture are part of on–going work in the development of a design framework for a learning collaboratory (LUCIDIFY: Learning Collaboratory Design Framework) and the instantiation of that framework in a testbed system (CLEOS: Collaboratory Learning Environment for Operational Systems)[3].

I. INTRODUCTION

In an educational setting, there are several reasons to support distributed instruction. First and foremost, the issue of allowing easier and more widely distributed access to instructional facilities is seen as a desirable way to increase the learning opportunities of a wide range of students. As financial resources become scarce, providing access to educational resources in a cheaper and more direct way becomes important to maintaining a high level of education. In the case of a learning community that has as its members not only students and teachers, but practitioners and researchers, increases in the interaction possibilities of all parties is potentially of mutual benefit.

In the physical and engineering sciences, students experience both classroom and laboratory instruction. Typically, classroom instruction focuses on theoretical principles, while laboratory instruction focuses on practical tasks such as running experiments using equipment. It is important that the links between theory and practice are strong and clearly articulated.

There are two parts to any instructional intervention: what is meant to be learned, and how it is taught. In this work, we are focused on teaching the theory and practice of a physical science experiment. We wish to create a distributed, Web-based tutorial system as a means to accomplishing this goal, and employ the teaching pedagogy of apprenticeship learning. A related family of experimental procedures can be taught in this way. Instead of starting from scratch for every tutorial system of this type, it is instructive to see if there are any common elements to such a family of systems. This notion of re-using the structural (pedagogical) aspects of an apprenticeship-learning based tutorial system forms the motivation of the creation of the Apprenticeship Learning Object Toolkit (ALOT). The ALOT system is a a library of Java objects from which one can build Java–based tutoring systems, where this library forms the basis of a generalized architecture for a family of computer tutoring systems.

The ALOT architecture's learning methodology is based on the theories of cognitive apprenticeship[1][7] and legitimate peripheral participation[8]. These theories emphasize learning–by–doing in the context of the activity (i.e. locating cognitive activity in context[11]) where the novice learner interacts with an expert.

A guiding principle in the design of a virtual environment to educate learners is to provide resources for the learners to engage in meaningful activity in a domain of authentic practice. Tutoring systems of virtual instrumentation provide a meaningful context in which to learn operational procedures.

Theory and practice are linked by situating the practice of procedures within a system that demonstrates the theory underlying the procedures. Multiple resources for exploratory learning and interaction with experts (be it a "virtual expert" embodied in the system, or a human expert on-line) are a cornerstone of a system that attempts to support all levels of expertise in the user.

II. SITUATED COMPUTER–SUPPORTED COLLABORATIVE LEARNING

Situated action theory emphasizes the local management of activity as mediated by relevant environmental cues[13][9]. The implications for learning are that appropriate actions are generated from a recognition of appropriate opportunities given the context. Apprenticeship learning is a means through which situated learning can occur, where apprentices are active participants in an activity, usually with an expert. Apprentices' process of learning moves from peripheral to full participation in the activities of a community of practice, as the expert "fades" from engagement of the activity. The support for active contextualized learning[10] lends itself well for the teaching of procedural knowledge and operational skills. Vygotskian theories of learning stress that individuals gain skills by engaging in tasks with an "adult or more capable peer"[14]. In general, four overlapping stages of pedagogy can be identified[1]: (1) Modeling, through the observation of expert performances, (2) Coaching, with expert guidance and help, (3) Fading, where expert assistance is gradually withdrawn, and (4) Reflecting, student self–monitoring and reflecting upon past performances.

Computer–Supported Collaborative Learning (CSCL) argues that learning is generally seen as a collaborative activity that is situated in its environment. Collaboration promotes convergence of a shared relational meaning[12]. Knowledge is constructed incrementally through a process of mutual contributions via interaction. For a more in–depth review of situated learning and the use of computer technology in education, the reader is referred to the Technical Report associated with this project[5].

III. ALOT: THE APPRENTICESHIP LEARNING OBJECT TOOLKIT

This paper describes a generalized architecture, derived from work done on the UIUC Virtual Spectrometer (UIUCO–VS)[4], that forms the basis of a family of computer tutoring systems. That family includes systems based on an apprenticeship learning pedagogy and is well suited for the teaching of the theory and practice behind physical science experiments. The Apprenticeship Learning Object Toolkit (ALOT) is a a set of libraries of Java objects from which one can build Java–based tutoring systems. The system utilizes the apprenticeship model of learning to create a learning environment for the study of operational procedures of experiments in the physical sciences. Examples include nuclear magnetic resonance (NMR) spectroscopy or x–ray diffractometry (XRD).

III.1 Design Goals of ALOT

The design goals of the ALOT object library is two–fold. First, we wish to capture the conceptual "objects" that comprise the pedagogical structure of a apprenticeship learning based tutorial system. Secondly, we wish to create a set of Java objects that can be re–used in creating multiple tutorial systems. This will require that the ALOT library provide Java classes that can be readily picked up and used by a system designer. The ALOT library is written in Java, as our purpose is to write systems that can be deployed onto the Web, and may form the basis of a shared simulation environment.

Using the experience of the design and construction of UIUC–VS as a guide, the first order of business is to identify the pedagogical structure of the ALOT libraries.

The ALOT Java library is a collection of Java objects. These objects can be used (re-used) to build a tutoring software program that encapsulate the apprenticeship learning pedagogy in its structure and approach to teaching operational procedures. The structure and pedagogy of such a tutoring system is exemplified in the LEMRS and UIUC–VS systems. The purpose of the ALOT library is to create a "generic" set of Java objects to allow a developer to quickly reuse the structure of the system to implement a different tutoring system. The goal then is to "abstract out" the components that provide the structure of UIUC–VS, and encapsulate it in Java libraries that can be re-used by a designer in constructing a new tutoring system. The ALOT Java Library allow a designer to quickly add the content and particular domain information to build a tutoring system.

The challenge in developing such a "generic" library is to provide enough structure to allow for the rapid development of a new tutoring system, without providing too much structure that constrains the designer's creativity and flexibility. If the object definitions are too inflexible, or poorly designed, then they will not be used. Towards achieving the right balance between structure and flexibility, we have attempted to create a set of objects that can be used in more than one way. More will be said on this later.

III.2 ALOT Library Structure

The ALOT library is comprised of several packages, where each package contains a set of Java objects. The packages that make up the ALOT system are: 1) alot.lang, containing objects that represent the pedagogical concepts of an apprenticeship learning style tutorial system, 2) alot.gui, containing definitions and standard graphical user interface (GUI) components to construct the tutorial system, and 3) alot.util, containing classes that support the functioning of the other two libraries.

The alot.lang package contains classes pertaining to pedagogical concepts and data organization. The alot.gui package is subdivided into three sub-packages. The alot.gui.basic contains some basic graphical widgets. The alot.gui.definition package contains interfaces[1] that define the methods the standard graphical panels should have to work with the rest of the classes. The alot.gui.standard package contains default graphical components that implement the definitions from the alot.gui.definition interfaces, which together comprise a partially completed graphical interface. The designer can use the standard component classes, or use their own custom classes. The alot.util package contains utility classes used by the other alot packages. Included in this listing are an examples of the packages written for a specific tutorial application (e.g.. The Virtual X-Ray Diffractometer).

III.3 The Pedagogical Objects of an Apprentices Learning Tutorial System

The structure of the semantic content of the tutorial system is expressed in the classes found in the alot.lang package. The key concepts and their relation to one another is discussed.

Lesson. An experiment can usually be broken down into several general steps, each of which form one lesson. A lesson itself may have several specific procedural steps for it's successful completion. Also, in the context of a tutorial system, each lesson usually has a graphical component associated with it.

Lesson Plan. A series of lessons can be grouped into a lesson plan. One lesson plan describes all the lessons that comprise a teaching scenario. It is possible to have multiple lesson plans in the same tutorial system. One lesson plan, for instance, may contain all the lessons for an experiment, where another lesson plan may contain only a subset of the defined lessons. Thus lessons can be used in multiple lesson plans in an attempt to provide a varied simulation environment to teach the concepts and procedures behind an experiment.

Procedural Step. Within a lesson, there may be various steps required for its successful completion. Each step has requirements for its completion. Successfully completing all the procedural steps within a lesson means that you have successfully completed that lesson.

Demonstration Step. In the "Show Me" or "Observe Expert" mode, the system takes control of the Lesson Window and guides the user through a series of pre-planned demonstration steps. These steps are designed to teach not only the actions necessary to successfully complete the lesson, but to explicate the theory behind those actions.

Domain Concept. In the course of interacting with the system, the user has the opportunity to explore relevant concepts. These domain concepts, and their elaborations, are presented to the user in a context-relevant information hierarchy.

"What If" Question. In addition to teaching the user how to successfully perform the procedural steps within a lesson, the system also provides some answers to "what if" questions. These questions are meant to capture the most common errors or misconceptions, and to allow the user to discover the ramifications of certain (not always correct) actions.

Learning Goals. Each lesson plan has associated with it a series of learning goals. It is often helpful to the student to be able to view these learning goals at the outset of a tutorial.

III.4 The Functional Objects of an Apprentices Learning Tutorial System

Having decided to create a Web-based system utilizing Java, the next decision is the conceptual layout of the program. The domain of the system is split into three primary pieces: (1) main lessons, (2) context-dependent information index and (3) expert knowledge. Each piece is encapsulated by a window (the Lesson Window, the Information Window, and the Expert Window, respectively). Additionally, the Menu Window serves to inform the student of where in the lesson plan they reside (inside the Lesson Window) currently. The Information Window is the area that responds to student requests for more information and feedback to their progress within a specific portion of a lesson. The information includes context-dependent checklists of procedures and also a hierarchy of information and concepts that the student can explore. Finally, the Expert Window, located at the bottom of the screen, displays the expert knowledge of the system in the form of answers to queries, what-if scenarios, and demonstration tutorials. This information is conveyed in text and voice form. Figure 1 illustrates a candidate structure of our system.

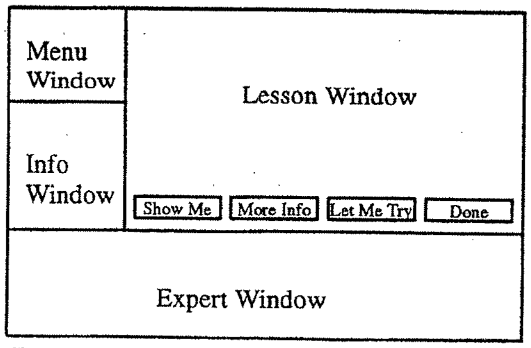

Figure 1. A candidate layout structure of an apprenticeship learning based tutorial system.

1. An *interface* is a Java class that declares abstract methods. Any Java class "implementing" this interface is thus required to implement the methods defined in the interface.

There are two modes in our architecture: 1) "Observe Expert" and 2) "Act as an Apprentice".

Observe Expert. Upon selecting this mode, the focus of control shifts to the Expert Window. Using both voice and text, the student is taken on a step–by–step tour of the entire experimental procedure. The student is able to disable the audio portion of the tutorial. The tutorial starts with a statement of the goals of the experiment, and takes the student through the operational procedures. The student simply keeps pressing the "Next Step" button to continue to the next step of the tutorial after reading (or listening to) the description of the present step. The Lesson Window is automatically updated throughout the demonstration. In fact, all the controls on the Lesson Window are disabled as the student proceeds through the demonstration. In this mode, the student is there to observe as the Expert (i.e. system) manipulates the experiment. The student may stop the demonstration at any time, and attempt to work though the experiment him– or her–self, via the Act as an Apprentice mode.

Act as an Apprentice. Selection of the Act as an Apprentice mode means that the student is ready to attempt to try to do the experiment him or herself. The student is presented with the menu of lessons from the current lesson plan. As the student enters a specific lesson page, the name of that page in the Menu Window is highlighted. As the student completes a portion of the lesson, the corresponding box in the Menu Window is checked. Since the Menu Window and its contents are always visible, the student always knows where in the lesson they are (via the highlighted label) and the status of their progress through the lesson (via the checked boxes).

Learning Goals. This non–required portion of the lesson explicitly states the learning goals of the lessons to follow.

Show Me. If the student presses an available "Show Me" button they will start the demonstration mode for the present lesson of the experiment. The Expert Window takes over control of the interface again, and the student is lead through a step–by–step tutorial of the procedures necessary to successfully complete this lesson of the experiment. The student can stop the demonstration at any time, and control of the Lesson Window passes back to the student, with the settings prior to the invocation of the Show Me demonstration mode restored.

Let Me Try. Upon pressing this button, the student is presented, in the Information Window, with a step–by–step checklist of the procedure for completing this portion of the lesson. As each step is successfully completed, the corresponding box is checked. The student can press Reset Lesson which will reset this portion of the lesson to its starting values.

More Info. The student can request more information on the present portion of the lesson by pressing the "More Info" button. A selection list of topics appears in the Information Window. After selecting a subject, the student presses the "What is it and Why is it Important?" button, and a textual answer appears in the Expert Window. Additionally, the student can explore what if scenarios by pressing the "What If...?" button in the Information Window. A selection list of "what if" scenarios appears in the Expert Windows, and after selecting a scenario the answer appears in the same window.

Throughout each portion of the lesson, the student always has the ability to return to the Lesson Plan Menu, and so can investigate previous portions of the lesson that they have already successfully completed, as well as quitting the system altogether.

III.5 Development Strategies and Issues

The development of the ALOT library is being driven from two directions. In essence, the first task is to look at The Virtual Spectrometer (UIUC–VS) and try to conceptualize how the code would have been if there was to be a decoupling of the basic pedagogical structure (as realized in the code) from the domain-specific lessons that comprise the tutorial. Conversely, the development of a second, new tutorial system, the Virtual X–Ray Diffractometer (VXRD) provides a design experience from the bottom up. The VXRD design will be based on the ALOT Library. So as that design takes shape, it will helpfully be clear what parts of the system code belong in the ALOT library, and what parts of the code are domain-specific to VXRD. By attempting to develop ALOT from both the perspective of the re-design of UIUC–VS and the perspective of the design of VXRD, we hope to create a robust set of generic Java objects that will aid in the design of any apprenticeship-style tutoring system.

The alot.gui package is of particular note because here is where decisions must be made as to how much structure the ALOT library will provide (or impose?). The final decision is twofold. One one hand, a partially completed graphical interface is provided, one that has as much structure as could be generated without knowledge of the actual lessons to be used in the system. This "standard interface" is intended to be as easy to pick up and use as possible. Easy-to-use methods to add "content" to the system are provided. On the other hand, however, the "standard interface" is written in such a way that if a system designer wanted to start from scratch and write their own interface from the bottom up, then they would still find objects that they could use within the alot.gui library. In addition, a designer using these libraries will be able to "swap out" and the standard component for a custom one of their own design. Each default class implements an interface, which requires certain methods to be associated with that class. A designer writing their own custom graphical component must also implement this interface, and so they will know specifically which methods their custom class must have in order to work with the other default classes within the system. In this way, the designer has the freedom to pick up an use only those standard GUI components as they so choose.

The default interface already has the structure to support two modes of operation (Observe Expert and Act as an Apprentice), access to context-dependent declarative and theoretical knowledge (More Info), and access to procedural knowledge (Let Me Try).

Information to "fill in" the domain content are encapsulated in alot.lang objects and the tutorial GUI object representing the lessons and instrumentation, which reside in the lesson panel, would be part of the specific application package. So in effect, an application is built that makes use of a library of objects that, at its maximum re-use, would provide the conceptual and GUI underpinnings of a tutorial system.

The designer creates two classes of the type SemanticSpecification and GUISpecification, where they instantiate the semantic and GUI classes, respectively, necessary to populate the system. The semantic information consists of alot.lang objects like LessonPlan(s), Lesson(s), DomainConcepts(s), etc. The GUISpecification class creates all the GUI components that constitute the tutorial interface. These GUI components can be those found in alot.gui.standard, or can be created by the designer.

The design of the ALOT library is based on the new windowing toolkit offered by Sun, Swing. Swing will become part of the core Java language upon release of JDK 1.2, expected Fall, 1998. The Swing toolkit is a powerful graphics package that is written in 100% Java (no native peers) and so the developer had much more control over the look and feel of the application. In addition, Swing is fully Beans-compliant. This will enable all the Java objects in the alot.gui directory to be cast all as Java Beans. A Java Bean can be directly plugged into a graphical GUI-builder and this again will support rapid development of tutoring systems, a goal mentioned above.

IV. BAUEN: A BASIC AUTHORING ENVIRONMENT

The ALOT Java library is intended to facilitate the design and development of a family of tutorial systems. The difficult part of the design of the tutorial system remains to the designer however. Studying the domain, understanding and deciding on the methods of instruction, devising lesson plans and demonstrations are all work that falls outside the scope of this paper, but are vital to the success of a tutorial intervention. However, once the semantic domain knowledge has been acquired, demonstrations designed, relevant domain concepts identified, etc. this information must be coded into the software. In an attempt to facilitate this process, a "helper" application has been designed. The BAUEN (Basic AUthoring ENvironment) system is a graphical user interface system that is intended to help the designer author a computer tutorial system with maximum efficiency.

The inputs to the BAUEN system are the semantic data and a specification of which GUI components will be used (custom or standard). The output of the BAUEN system are three ready-to-compile class files. Using the BAUEN system, the user inputs the information to define the contents of alot.lang objects like LessonPlan(s), Lesson(s), DomainConcepts(s), etc. BAUEN will output a [MyApp]SemanticSpecification.java file that builds these objects. Likewise, BAUEN allows the designer to select standard GUI components, or input the class name of their own custom components. The system then outputs a file of type [MyApp]GUISpecification.java. These two files define the bulk of the semantic domain and GUI component information for the system. Once finished with BAUEN, the user can simply compile its resulting output files to create a complete working tutorial system.

V. ON-GOING AND FUTURE WORK

The ALOT system will be expanded to incorporate mechanisms to handle synchronous communications between instances of the tutorial running simultaneously on different machines. Initial trials using Java Remote Method invocation (RMI) have been successful. The ALOT architecture effort is part of a larger project whose goal is to develop a framework and demonstration testbed for the design of a learning collaboratory. The focus is on supporting effective collaborative learning via system design. The Learning Collaboratory Design Framework (**LUCIDIFY**) is a framework for design that combines methods to structure domain knowledge, represent navigational strategies, characterize expertise, and support collaborative learning and work. The Collaboratory Learning Environment for Operational Systems (CLEOS) will serve as a testbed for LUCIDIFY. CLEOS is envisioned as a virtual learning environment where students can collaboratively learn the theory and practice of operational systems in the context of doing projects and experiments using simulated instrumentation. Project-based learning will be supported via a project management tool that helps instructors create and manage multiple student projects.

VI. References

[1] A. Collins, J. S. Brown, and S. E. Newman, "Cognitive Apprenticeship: Teaching the craft of reading, writing, and arithmetic", Cognition and Instruction: Issues and Agendas, Resnik (ed.), Hillsdale, NJ: Erlbaum, 1987.

[2] Michael Dorneich and Patricia Jones, "Supporting Apprenticeship Learning of NMR Spectroscopy in a Collaborative Web–Based Learning Environment", *Technical Report HCCPS–97–01*, The Human Computer Cooperative Problem Solving Laboratory, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana–Champaign, 1997.

[3] Michael Dorneich, "Towards a System Design Framework for a Collaborative Learning Environ-

ment of Operational Procedures", *Technical Report HCCPS−97−03*, The Human Computer Cooperative Problem Solving Laboratory, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana−Champaign, 1997.

[4] Michael C. Dorneich and Patricia M. Jones, "The Virtual Spectrometer: A Java−based Implementation of a Learning Environment", *1997 IEEE International Conference on Systems, Man, and Cybernetics*, Orlando, Florida, October 12−15, 1997.

[5] Michael Dorneich, "The Apprenticeship Learning Object Toolkit: A Generalized Architecture for a Family of Computer Tutoring Systems", *Technical Report HCCPS−97−04*, The Human Computer Cooperative Problem Solving Laboratory, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana−Champaign, December 1997.

[6] Patricia M. Jones and Kenneth J. Schneider, "Learning environment for Magnetic Resonance Spectroscopy (LEMRS): Supporting Apprenticeship Learning in Operational Environments", *Journal of Educational Multimedia and Hypermedia*, Vol 5, no. 2, pp. 151−177, 1996.

[7] S. Lajoie and A. Lesgold, "Apprenticeship training in the workplace: Computer–coached practice environment as a new form of apprenticeship". In M. J. Farr and J. Psotka (Eds.), Intelligent instruction by computer: Theory and practice (15–36). New York: Taylor and Francis, 1992.

[8] Jean Lave and Etienne Wenger, Situated Learning: Legitimate Peripheral Participation, Cambridge University Press, 1991.

[9] Agre, P. E. and Chapman, D., "Pengi: An implementation of a theory of activity", *Proceedings of AAAI−87*, Los Altos, CA, 1987, pp. 196−201.

[10] Hoppe, H. U., "Cognitive apprenticeship: The emperor's new method? A polemical reaction to the debate on situated cognition and cognitive apprenticeship", *Journal of Artificial Intelligence in Education*, vol.4 no. 1, 1993, pp. 49−54.

[11] Edwin Hutchins, *Cognition in the Wild*, MIT Press, 1995.

[12] Jeremy Roschelle, "Learning by Collaborating: Convergent Conceptual Change", CSCL: Theory and Practice of an Emerging Paradigm, Lawrence Erlbaum Associates, New Jersey, pp. 209–247, 1996.

[13] Suchman, Lucy, *Plans and Situated Action: The problem of Human −Machine Communication*, Cambridge University Press, 1987.

[14] Vygotsky, L., *Mind and Society*, Cambridge, MA: Harvard University Press, 1978.