Special Purpose Ontologies and the Representation of Pedagogical Knowledge

Tom Murray

Center for Knowledge Communication
University of Massachusetts
Amherst, MA, USA
tmurray@cs.umass.edu, www.cs.umass.edu/~ckc/

Abstract: In intelligent learning/tutoring environments which deal with several types of knowledge or with knowledge in complex domains, some form of pedagogical and curriculum knowledge must be represented in order for the ITS to offer guidance and structure in the learning process. In this paper we illustrate how pedagogical knowledge is represented by various instructional theories and ITS systems, discuss a number of key issues involved in authoring pedagogical knowledge, and show how these issues are addressed in the representational framework of the Eon ITS authoring tools. Unique among the methods we use are Ontology objects, which allow for the creation of representational frameworks tailored to classes of domains or tasks, and "topic levels," which provide an additional level of sophistication beyond network representations, while increasing cognitive clarity.

Introduction

In this paper we discuss how pedagogical knowledge can be represented in intelligent computer tutors. We start by arguing for the need for pedagogical and curriculum knowledge in Intelligent Tutoring Systems (ITSs), then review how the subject is addressed in various instructional theories and ITSs. Then we highlight important issues, problems, and tradeoffs in representing pedagogical knowledge, showing how we have addressed these issues in our suite of ITS authoring tools. Along the way we propose a powerful method for achieving both scope and depth of knowledge representation, which we call an Ontology object.

Before proceeding any further, we need to clarify what we mean by pedagogical knowledge. We distinguish two categories of teaching knowledge: procedural knowledge, which we call teaching strategies, and declarative knowledge, which we call pedagogical knowledge (see Figure 1). Pedagogical knowledge is the part of domain knowledge which is relevant to teaching or learning about the domain, as opposed to domain knowledge that represents performance expertise in the domain. It is descriptive, as opposed to prescriptive, in nature. Pedagogical knowledge includes curriculum knowledge, which deals with how the knowledge of the domain is carved up and organized to facilitate instructional decisions.

The Need for Explicit Representations of Curricular and Pedagogical Knowledge

It is generally acknowledged that the effectiveness of computer-based instruction depends heavily on two criterion: 1) that learning environments simulate the contexts in which the target knowledge will be used (or do so with reasonable fidelity), and 2) that appropriate guidance be given to assist student learning. Learning is more motivated if it is relevant to meaningful situations, and is more effective if students "learn by doing." Instructional systems, since they are designed artifacts, will always have instructional goals (even if the goals are vague, implicit, or open ended), and all but the most motivated, advanced, and prepared students will require guidance in achieving these goals. This guidance can come in many forms, from giving coaching hints to the selection of problem situations tailored to student needs.

In order to 1) bias a learning environment and structure the sequence of what is presented for efficient learning (a top down approach), and 2) respond to student behaviors and queries, (a bottom up or opportunistic approach), intelligent tutoring systems must contain expertise in the subject being taught, and in how to teach that material. Perhaps because representing expertise is so challenging, much effort has been put into building ITSs that have deep or expert system representations of subject matter, but relatively little has been put into exploring the knowledge that is needed for teaching.

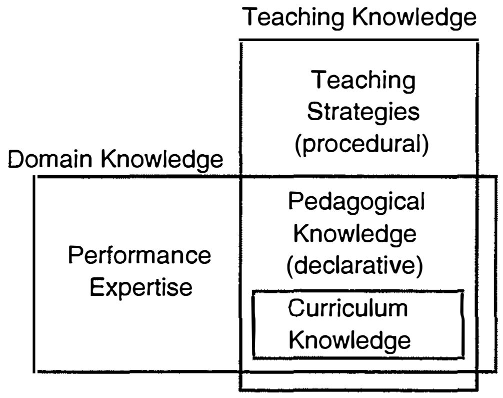
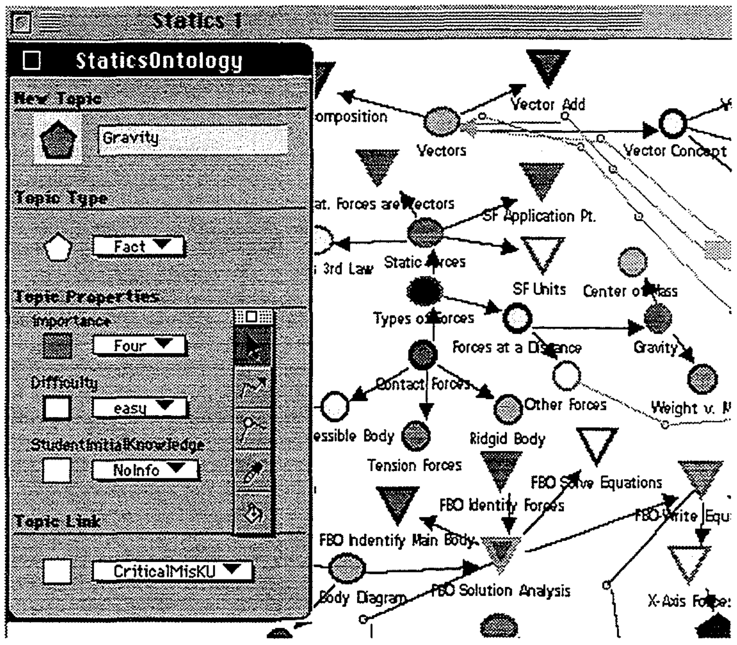


Figure 1: Teaching and Domain Knowledge



New Topic: Gravity
Topic Type: Fact
Topic Properties:
Importance: Four
Difficulty: easy
StudentInitialKnowledge: NoInfo
Topic Link: CriticalMisKU

Figure 2: Topic Network Editor

Clearly, the learning or teaching of different things is done best in different contexts using different methods [Gagne 1985], and learning any non-trivial skill involves learning a number of component pieces of knowledge, some of which are dissimilar in their pedagogical requirements. Therefore, ITSs need to have some representation of all of these content pieces, and some way to distinguish and use their pedagogically relevant characteristics. We call this representing the *curriculum*. The question about how small and precise these curriculum knowledge components need to be is open, but some form of abstract representation must exist for the system to reason about instruction or guidance. In our ITS authoring tool development efforts we are trying to design reusable representations for pedagogical knowledge which are general (and therefore relatively minimal) yet can be easily extended in a principled manner. First we describe related work from the literature.

Previous Work in Representing Pedagogical Knowledge

Much of what has been written about instruction combines ideas about how to teach with ideas about how to represent knowledge for instructional purposes. In fact, since all representational schemes are designed for some purpose, each feature of a scheme presupposes some pedagogical assumptions. For example, a representational scheme which has an object called a "hint" presupposes some uses for hints, and those uses actually define the meaning of "hint" in that system. Though we acknowledge this confounding between representation and use, we will ignore it and focus our discussion on pedagogical knowledge representation, mentioning how this knowledge is used (i.e. in tutoring strategies) only in passing. Below we summarize key learning theories related to representing pedagogical knowledge.

Hierarchies and concept learning. Ausubel's "subsumption theory" of learning [Joyce & Weil 1986] focuses on the hierarchical organization of concepts in disciplines, and thus is amenable to computational modeling. He proposes that abstract knowledge (further up in the hierarchy) is more meaningful and useful, and preferred to more specific or rote learning. His Advanced Organizer model prescribes that new information must relate to previous information, and that effective learning paths through the hierarchy of knowledge will differ for each student. Web teaching [Halff 1988] similarly requires that knowledge networks be annotated with information about the relatedness of topics (prefer more closely related topics) and generality (give generalities before specifics).

Procedural knowledge. The subsumption relationship is valid for conceptual learning, but pedagogical knowledge for procedural or skill learning requires a different treatment. Burton and Brown's BUGGY tutor [1982] uses a skill lattice to represent subtraction subskills. The NEOMYCIN system [Clancey

1982] uses an and/or lattice to represent medical diagnostic procedures. The BIP-II programming tutor [Westcourt et. al 1977] uses a network of subskills related by four links: analogous, harder than, same difficulty, prerequisite.

Lessons and instructional goals. [Lesgold 1988] points out that the concept of prerequisite is often inadequate, since whether one topic is a prerequisite of another may be a function of the learning goal of a particular session, rather than a static relationship between topics. He proposes a goal lattice structure which captures the different "viewpoints" of a curriculum structure which result from different instructional goals (or perspectives).

Lehnhardt and Greeno [1986] distinguish lesson structure and subject matter as the two fundamental systems of knowledge needed for teaching, where subject matter knowledge is used by the lesson structure, the later being in charge of tailoring a session for an individual student.

Beyond hierarchies and lattices. Domain knowledge is usually messier than can be represented in a simple hierarchy or lattice. Goldstein's [1982] Genetic Graph includes relationships among procedural rules which represent the way knowledge evolves while a student learns how to master a maze exploration game. The relationships include explanation, generalization, analogy, and refinement, and show how learning can follow knowledge pathways from abstract (simple) to more refined, from deviation to correction, and from specialization to generalization.

Knowledge attributes. Bruner's [1966] theory of learning focuses on how we form new concepts, categories, and rules by induction from examples or cases along with the analysis of key features. This indicates that not only knowledge chunks and their relationships, but also their pedagogically relevant properties, need to be represented for instruction. Case-based tutors (such as some of the Goal-Based Scenarios described in [Schank et al. 1994]) which use knowledge bases of example objects or situations search the knowledge base for appropriate cases based on case attributes.

Types of Knowledge. All of the work mentioned thus far deals with organizing units of knowledge which are basically of one type (usually concepts or procedural skills). But, as attested to by the diversity of the above instructional approaches, there are many types of knowledge.

Bloom [1956] and Gagne [1985] were among the first to develop clear classifications of knowledge and learned behavior, and assert that different types of knowledge require different types of learning or instructional methods. Other knowledge typing schemes were later developed which are better grounded in modern cognitive theory and are more operational and concrete for the purposes of computational representation. For example, Merrill's Component Display Theory [Merrill 1983] organizes knowledge in a matrix with *content type*, e.g. fact, concept, or procedure on one axes and *performance level*, e.g. remember, apply, and create, on the other. This matrix scheme is more expressive and intuitive than hierarchical representations. Kyllonen and Shute [1988] propose a more complex multidimensional model which distinguishes knowledge types in a hierarchy which illustrates cognitive complexity, and organizes these types in relation to the level of autonomy of learning and the processing speed needed to perform the task.

Representing buggy knowledge. Many of the theories and instructional systems mentioned above include some representation of buggy knowledge and a method for remediating it. Buggy knowledge, such as misconceptions and buggy skills or rules, is usually represented in a form similar to its corresponding performance knowledge, but with additional properties and relationships that allow the buggy knowledge to be diagnosed and remedied.

Summary. The above theories lead to a number of recommendations about generic pedagogical knowledge representation formalisms for ITS. The use of network formalisms with directed links allows for the creation of hierarchies and lattices and less canonical frameworks, and it is clear that a number of different types of nodes and links may be needed for an ITS. Learning goals for a tutorial session should be represented separately from instructional content. ITSs should be able to distinguish among different types of knowledge, and also represent buggy knowledge, so that teaching strategies can be predicated on knowledge classes. There is also an indication that exclusive use of a network formalism does not provide enough knowledge structuring, and that more complex data structures will often be needed.

Special Purpose Ontologies and ITS Authoring Tools

We are in the process of developing a suite of ITS authoring tools. Therefore, our goal is not to determine which of the above methods for representing pedagogical knowledge are "best," but to look for commonalties and underlying principles. The suite of tools, called Eon, includes a conceptual vocabulary and representational structure for describing content, teaching strategies, student models, and interface design, and a user interface for authoring this knowledge. The tools are designed for users with only a little experience in instructional design and knowledge representation (i.e. not every teacher or trainer would be able to use them, but every school or company could easily have at least one person trained to use them, who could work with content experts and graphic artists to build ITSs).

A number of generic frameworks or shells have been proposed for ITSs, each with its own architecture for knowledge representation (many are discussed in [Murray 1996a]). Each architecture is well suited for certain domains and poorly suited for others. Special purpose shells, which are used to build tutors for specific task types [Jona 1995, Dooley et al. 1995], can build tutors with more fidelity and depth than general purpose shells. To address this generality/power tradeoff we propose a *meta-shell* approach. Though Eon is an authoring system for creating ITSs, we also see it as an authoring shell for creating special purpose authoring shells. Our goal is to be able to provide special purpose ITS shells tailored to domain or task types. For example, ITS authoring shells could be produced for science concepts, human service/customer contact skills, language arts, and equipment maintenance procedures. Instructional designers using special purpose authoring tools do not have to start from scratch, but can immediately start constructing a tutor in an environment that supports and helps structure the knowledge acquisition process. They would be provided with default teaching strategies, default student modeling rules, default student interactions, and a topic structure (see "Ontology" below) which is tailored to a specific type of domain and/or task, which they could use as is, or modify.

Key to our concept of meta-authoring is the Ontology object. This object defines the conceptual vocabulary and underlying structure for the domain knowledge for each special purpose ITS shell (or for a particular ITS).[1] Eon uses a semantic net representation of the tutor's knowledge called the "Topic Network". The Ontology defines the types of topics and topic links allowed in the network. For example, our Statics Tutor [Murray & Woolf 1991a] has topic types Fact, Concept, Procedure, and Misconception, and topic links Prerequisite, Generalization, and Sub_Concept; while our Manufacturing Equipment tutor has topic types Safety, Maintenance, Operation, Theory, and Common Failures, and topic links Sub_Part and Similar_Part (both of these tutors are prototypes, and their ontologies are still in flux). Both of these Ontologies are not specific to the domain, but appropriate for a class of domains. Eon Ontologies specify a number of other things, such as topic properties (e.g. difficulty, importance), and allowed student model values.[2]

A Curriculum Object Framework

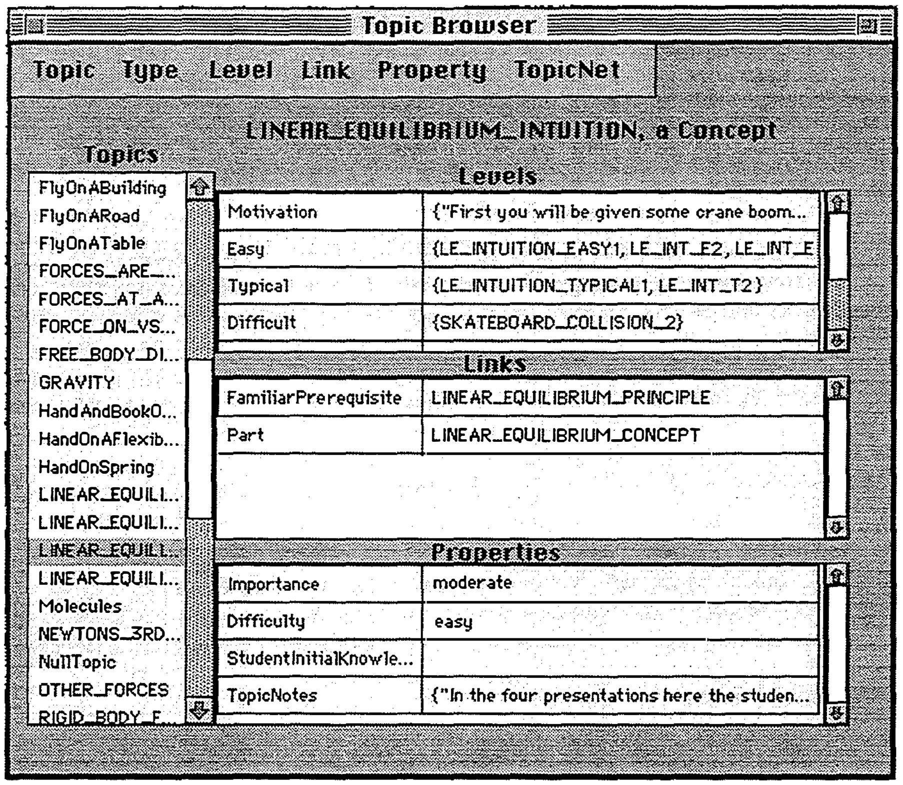
As one could guess from the variety of instructional approaches described above, trying to design the most general or powerful representational framework for an ITS will, at the least, result in a very complex and obscure system. In contrast, we have tried to identify a minimum underlying object-oriented framework on which to build Ontologies, which is neutral regarding domain or instructional theory, and let the Ontology designer specify the remaining complexity.

From our overview of previous work, we conclude that a minimum framework must have the following capabilities: 1. A method for distinguishing the abstract representations of "knowledge" from the concrete media which the student will see, read, hear, and manipulate; 2. A method for referring to discrete chunks of abstract knowledge and the relationships between the chunks; 3. A method for distinguishing the goals and needs of a particular tutorial session from the general domain pedagogical knowledge; and 4. A method (or methods) for assigning instructionally relevant attributes, categories, or purposes to chunks or knowledge or media, e.g. "explanation," "summary," "hint," "difficulty," so that teaching strategies can effectively use the information. To address these minimal requirements, the Eon system uses the following objects or mechanisms, which we describe along with the authoring tools used to create them.[3]

[1] The ARPA Knowledge Sharing Effort (KSE) described in Gruber (1993) is exploring the use of standardized ontologies for sharing knowledge in knowledge-based systems. Our work is related, but currently we are focussing on ontologies that support knowledge authoring rather than sharing, and we focus on pedagogical knowledge, where the KSE deals with performance knowledge. Also, our purposes are, for the time being, more practical and less rigorous, in that *instructional* systems need to accomodate a wide range of often very complex target knowledge, for which rigourous analysis is beyond the state of the art, and simplified or ad-hoc representational schemes are employed to make representation tractable.

[2]In Murray 1996A we describe the Eon ITS authoring tools and the design tradeoffs involved in balancing the needs for power/flexibility, and usability, and how we provide tools for two levels of users, one of which designs Ontologies and special purpose shells, and the other which uses special purpose shells to build ITSs.

[3]One tool we will not describe in this paper is the Strategy Editor, which is used to visually author the tutoring strategies which specify how and when to teach topics, give presentations, and respond to student behavior (see [Murray 1996a]).



LINEAR_EQUILIBRIUM_INTUITION, a Concept

Topics

FlyOnABuilding
FlyOnARoad
FlyOnATable
FORCES_ARE_...
FORCES_AT_A...
FORCE_ON_VS...
FREE_BODY_DI...
GRAVITY
HandAndBookO...
HandOnAFlexib...
HandOnSpring
LINEAR_EQUILI...
LINEAR_EQUILI...
LINEAR_EQUILI...
LINEAR_EQUILI...
Molecules
NEWTONS_3RD...
NullTopic
OTHER_FORCES
RIGID_BODY_F...

Levels

Motivation	{"First you will be given some crane boom...
Easy	{LE_INTUITION_EASY1, LE_INT_E2, LE_INT_E
Typical	{LE_INTUITION_TYPICAL1, LE_INT_T2}
Difficult	{SKATEBOARD_COLLISION_2}

Links

FamiliarPrerequisite	LINEAR_EQUILIBRIUM_PRINCIPLE
Part	LINEAR_EQUILIBRIUM_CONCEPT

Properties

Importance	moderate
Difficulty	easy
StudentInitialKnowle...	
TopicNotes	{"In the four presentations here the studen...

Figure 3: Topic Browser

• **Topics**. Knowledge elements of any size are represented by Topics. Since one can define specific types of topics (see below) and create hierarchical links between topics, this simple object suffices for many representational schemes. Topics can have any number of Topic Properties, such as difficulty and importance.

The Ontology defines these properties and their allowed values.

• **Topic Links**. The Ontology defines the types of links (relationships) allowed between topics. Not constraining the link types allows for the representation of a wide variety of topic networks, including component hierarchies, instructional goal lattices, concept networks, etc. Figure 2 shows the Topic Network Editor and the Node Pallet which is used to create and edit topic nodes and links. The pallet reflects the Ontology begin used for the Statics tutor domain. Pop-up menus on the Pallet show the allowed values for topic type (shape), topic properties (color, border width, and border color), and topic links (link color).

• **Presentations**. Whereas Topics are *abstract* objects that refer to modular chunks of the knowledge to be taught, Presentations contain the *specific* contents (media) that the student will see and manipulate (text, graphics, etc., or the templates or algorithms for generating this content). Presentations are expository or inquisitory interactions, usually associated with a specific interactive screen templates which the author designs using the Interaction Editor (see [Murray 1996a]).

• **Topic Levels**. As explained later, having only the semantic net to represent all aspects of curriculum structure was found inadequate. Topic Levels allow for distinguishing multiple levels of performance (e.g. memorizing vs. using knowledge), mastery (novice to expert), and pedagogical purpose (summary, motivation, example, evaluation, etc.) for each topic. The Ontology defines what levels are available. Presentations are associated with Topics via Topic Levels, as specified in the Topic Browser (see Figure 3). For example, the Motivate level of the topic Gravity will specify the list of Presentations used to prime the student's interest in that topic. For buggy knowledge, we create Diagnose and Remediate Topic Levels.

• **Topic Types**. As mentioned, there is general agreement that there are different types of knowledge, each type having its own properties and each type requiring a different instructional method. Rather than have knowledge type be simply a property of Topics, Topic Types are first class objects which all Topics inherit from. Since there are numerous theories of knowledge types, we leave knowledge type definition to the Ontology. Each knowledge type defined in the Ontology has its allowed properties, allowed link types it can connect to, and allowed topic levels. Buggy Knowledge is represented as topics of type Misconception, Procedure_Bug, etc., depending on the type of knowledge that is in error. Near the top of Figure 3 "a Concept" following the current topic's name indicates the Topic Type.

• **Lessons**. Topic networks, which define pedagogically relevant relationships between topics, do not specify any ordering or starting point for learning sessions, and are independent of the instructional purpose of sessions. Lesson objects are used to specify instructional goals and learning/tutoring styles for a particular group of students. Nominally, the Lesson lists a small number of starting or goal topics, and specifies the default teaching strategy.

These basic elements were included in Eon because some form of all of them (with the exception of Topic Levels) seem necessary for any pedagogical representation. Now that we have described our underlying representational framework, and how Ontology objects are used to create more complex representational frameworks for specific purposes, we will consider several important issues encountered when we represent the pedagogical knowledge of a domain in an AI tutor, and our approaches for dealing with these issues. Though our discussion will be phrased in terms of Eon's representational scheme, the issues are universal to all ITSs.

Problems and Solutions in Representing Pedagogical Knowledge

Human knowledge does not exist in neatly defined, clearly named packages---it is inherently complex, densely connected, fuzzy, and ambiguous. Yet to use knowledge in AI systems we try to represent it in individual units with clear structure. The tension between the organic nature of knowledge and our need to modularize it leads to a number of issues for ITS knowledge representation, which we discuss below.

Knowledge Structure and Complexity

Our overview of past work illustrates that there are many ways to structure knowledge, and that these can be represented computationally using common data structures such as hierarchies, lists, networks, arrays, and frames, as well as multi-dimensional, multi-level combinations of nested data structures. Though, theoretically, a knowledge space of arbitrary complexity can be represented with just knowledge objects (topics) with their properties and interrelationships (links), very complex structures are unwieldy, esoteric, and difficult to use and maintain. Knowledge authoring must be supported with tools that allow clear visualization of the knowledge structures, and ITS authoring tools must commit to and support (via visualization tools) an underlying structure.

Knowledge is structured in Eon using several mechanisms. First is the layering of basic object classes: Lessons, Topics, Topic Levels, and Presentations, mentioned above. The second method is in allowing arbitrary classifications of Topics and Topic Links in the topic network. Given the right Ontology, all of the hierarchies, lattices, and networks mentioned in the Previous Work Section can be represented in topic networks. The third mechanism is the Topic Levels themselves.

Topic Levels, though a simple construct, are unique to the Eon framework, and allow a significant level of representational sophistication without loss of clarity. For example, we have simulated Merrill's Performance Content Matrix by assigning content types to Topic Types (fact, procedure, skill, etc.), and performance levels to Topic Levels within each topic (remember, use, apply, create, and meta-knowledge). Without Topic Levels, there would be a confusing proliferation of related Topics, one for each level of each Topic, i.e. we would need topics called Gravity-memorize, Gravity-use, etc. As mentioned, levels of mastery can also be encoded, as can different pedagogical functions for a topic. For example, a teaching strategy can tell a topic to "teach" itself, "summarize" itself, "define" itself, and "test" for its knowledge if the Ontology defines these topic levels [Murray & Woolf 1991b].

Modularity vs. Interdependence

When the knowledge in a domain is organized into modular units, which are then sequenced flexibly according to instructional strategies, a number of unavoidable problems arise. First, it is difficult to encode the knowledge "between" the topics, which can be about how they are related to each other, or emergent knowledge that comes from understanding topics together. Reigeluth [1983] and Lesgold [1988] refer to this as the "glue" in a curriculum.

In Eon we deal with this glue in several ways. First, Topics can have Topic Levels such as "Introduction" and "Conclusion," which address how the topic relates to other topics that are likely to precede or follow it in most curriculum paths. Second, Topic Types called Composites and Synthesizers can be used. A

Composite Topic is one that represents the whole which is more than the sum of its parts. For example, on our Static Tutor's Linear Equilibrium (LE) topic had "sub-part" links to LE Intuition, LE Concept, and LE Principle. Knowing Linear Equilibrium involves knowing each of these parts, and also how the parts fit together in an understanding of static situations. "Synthesizer" is a term used by Reigeluth (1983) for instructional components that interrelate and integrate instructional units. Synthesizer topic types in Eon can point to two or more other topics and compare or contrast them.

Eon has one other mechanism for dealing with curricular "glue:" Lesson objects. Since Lesson objects can specify a sequence of goal topics, they can also be used to insure that certain information is presented between topics, to compare and contrast them.

The second modularity problem is that topics are often interdependent. For example, in our Statics tutor, the student has to know something about Gravity to fully understand Linear Equilibrium, yet some understanding of Linear Equilibrium is prerequisite to learning about Gravity. Topic Levels organized by mastery, combined with levels of prerequisites, allow us to deal with this in Eon. We can assign content to topics at various levels (e.g. easy, normal, and difficult) and allow prerequisite links such as simple-understanding, and full-understanding. We can specify that the easy level of Linear Equilibrium should precede learning the difficult level of Gravity, and that the easy level of Gravity should precede the difficult level of Linear Equilibrium. Thus we can simulate spiral teaching, in which the same topics are taught from successively more difficult perspectives (see [Murray & Woolf 1992] for more discussion of spiral teaching).

The third modularity problem is that there is a tradeoff between the modular integrity of the units and smooth tutorial transition between the units. It is hard to design a unit whose text makes sense regardless of what comes before and after. In practice, this is dealt with by providing tools which allow the designer to easily step though many potential paths, check for dialogue consistency, and make adjustments to the knowledge base. When feasible, dynamic text generation can ameliorate this problem. In Eon we can assign an arbitrary function to any chunk of text that appears on the student screen. This mechanism could in theory be used to employ natural language generation techniques, but in practice we use it for template-based text generation.

Conceptual Vocabularies

Though special purpose ITS shells with customized Ontologies will make tutorial construction much easier, it will still take at least a small degree of instructional design and knowledge engineering skill to use an established representational framework to build a tutor. Our experience with users will advise us as to the types of documented design guidelines that will be needed. The field needs more in the way of semi-standard terminologies for describing domain and teaching knowledge, in order to better compare systems and share knowledge bases. We are in the process of designing a conceptual vocabulary of primitive tutorial actions, pedagogical parameters, and a knowledge classification scheme that authors can use to organize and codify domain knowledge and teaching knowledge. The goal is to design a conceptual vocabulary for describing the objects, actions, and parameters of instruction which will serve as building blocks or conceptual primitives of representation. We are culling these terms from the literature in instructional design, cognitive psychology, and intelligent tutoring systems, to build a loose taxonomy which, in its first incarnation, will serve more as a paper-based knowledge acquisition tool.

The goal is not to build a complete prescriptive model for instruction, it is simply to offer a kitchen-sink style loose taxonomy of terms (with definitions) intended for generality and completeness, not coherence and consistency, which a designer can choose from to define a subset of these terms which will form a coherent model for a particular domain or class of domains. The taxonomy includes lists of topic types, topic links, and topic properties, which can be used to develop Ontology objects. For example, the list of topic links is: a-kind-of, sub-procedure, sub-component; causes, allows, requires, justification, purpose, supports; concrete/abstract, specialization/generalization; prerequisite, critical-prerequisite, familiarity-required, mastery-required; critical-misconception-of; deviation/bug; analogy, bridging analogy, anchoring analogy; synthesizes; comparison.

Also included are taxonomies for domain types, task types, primitive tutorial actions (for instruction, coaching, and hinting), tutorial decision parameters, student modeling parameters, tutoring styles, and explanation and question types.

Conclusion

We have illustrated how pedagogical knowledge is represented by various instructional theories and ITS systems, discussed a number of key issues involved in authoring pedagogical knowledge, and shown how the representational framework underlying the Eon authoring tools addresses these issues. Of particular note are our use of Ontology objects for the defining knowledge structures and primitives for domain classes , and Topic Levels for organizing a topic's contents around pedagogical purpose. We also have a parallel effort to develop a semi-canonical conceptual vocabulary of primitive tutorial actions and decision parameters, which will be used in part as a source for Ontology terms.

Our ITS authoring tools are usable prototypes, and we have begun using them to build several tutors in domains such as introductory Statics, solvency and chemical reactions, a part of Japanese language called "honorifics," and the thermodynamic principles underlying refrigeration. These first tutors are intended to be in diverse domains in an attempt to flesh out the representational power of the framework and the usability of the tools. Since we have not yet built tutors with considerable similarity in their pedagogical structure, our experimentation with reusable Ontologies and the meta-shell approach is just beginning. Our near term goals include trying to represent a number of documented instructional strategies in Eon, and trying to identify Ontology objects which will be useful for multiple domains with similar pedagogical characteristics.

Acknowledgments

This material is based upon work supported by the National Science Foundation and the Advance Research Projects Agency under Cooperative Agreement No. CDA-940860.

References

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

Bloom, B. (1956). Taxonomy of Educational Objectives, Vol. 1. David McKay Co., New York.

Bruner, J. (1966). Toward a Theory of Instruction, Harvard Univ. Press, Cambridge, MA.

Burton, R. R., & Brown, J. S. (1982). An Investigation of Computer Coaching for Informal Learning Activities. In Sleeman & Brown (Eds.), Intelligent Tutoring Systems. New York, NY: Academic Press.

Clancey, W. (1982). Tutoring rules for guiding a case method dialogue. In Intelligent Tutoring Systems, D. Sleeman & J. Brown (Eds.), Academic Press 1982, pp. 201-225.

Dooley, S., Meiskey, L., Blumenthal, R., & Sparks, R. (1995). Developing reusable intelligent tutoring system shells. In AIED-95 workshop papers for Authoring Shells for Intelligent Tutoring Systems.

Gagne, R. (1985). *The Conditions of Learning and Theory of Instruction.* Holt, Rinehard, and Winston. New York.

Goldstein, I. P. (1982). The Genetic Graph: A Representation of the Evolution of Procedural Knowledge. In Sleeman & Brown (Eds.), *Intelligent Tutoring Systems.* New York, NY: Academic Press.

Gruber, T. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation* , Guarino & Poli (Eds.). Kluwer Academic Publishers.

Halff, H. (1988). Curriculum and Instruction in Automated Tutors. In *Foundations of Intelligent Tutoring Systems,* Polson & Richardson (Eds.). Lawrence Erlbaum Assoc., Hillsdale, NJ.

Joyce, B. & Weil, M. (1986). *Models of Teaching,* Prentice-Hall, Inc., Englewood Cliffs, NJ.

Jona, M. (1995). Representing and re-using general teaching strategies: A knowledge-rich approach to building authoring tools for tutoring systems. In AIED-95 workshop papers for Authoring Shells for Intelligent Tutoring Systems.

Kyllonen & Shute (1988). "A Taxonomy of Learning Skills." Brooks Air Force Base, TX: AFHRL Report No. TP-87-39.

Leinhardt, G., & Greeno, J. (1986). The Cognitive Skill of Teaching. In *Journal of Educational Psychology*, Vol. 78 No. 2, 75-95.

Lesgold, A. (1988). Toward a Theory of Curriculum for Use in Designing Instructional Systems. In Mandl & Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems,* Springer-Verlag, New York.

Merrill, M.D. (1983) Component Display Theory. In *Instructional-design theories and models: An overview of their current status,* pp. 279 - 333. C.M. Reigeluth. (Ed), Lawrence Erlbaum Associates, London.

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

Murray, T. (1996a). Having It All, Maybe: Design Tradeoffs in ITS Authoring Tools. *Proceedings of the ITS-96 Conference,* Montreal, Canada, June, 1996.

Murray, T. (1996b). From Story Boards to Knowledge Bases: The First Paradigm Shift in Making CAI "Intelligent.". *Proceedings of the ED-Media 96 Conference,* Boston, MA, June 1996.

Schank, R., Fano, A. Bell, B. & Jona, M. (1994). The Design of Goal-Based Scenarios. *Journal of the Learning Sciences,* Vol. 3 No. 4.

Reigeluth, C. (1983). The Elaboration Theory of Instruction. In Reigeluth (Ed.), *Instructional Design Theories and Models,.* Lawrence Erlbaum Assoc., Hillsdale, NJ.

Westcourt, K., Beard, M. & Gould, L. (1977). Knowledge-based adaptive curriculum sequencing for CAI: application of a network representation. *Proceedings of the National ACM Conference,* Seattle, Washington, pp. 234-240.