TOM MURRAY

Chapter 15

PRINCIPLES FOR PEDAGOGY-ORIENTED KNOWLEDGE BASED TUTOR AUTHORING SYSTEMS:

Lessons Learned and a Design Meta-Model

Abstract. While intelligent tutoring systems (ITSs), also called knowledge based tutors, are becoming more common and proving to be increasingly effective, each one must still be built from scratch at a significant cost. This paper discusses a number of design issues and design tradeoffs that are involved in building ITS authoring tools, and discuss knowledge acquisition and representation "lesson learned" in our work. A generic framework or "reference model" called KBT-MM (knowledge based tutor meta-model) for knowledge based tutor authoring tools is described. The reference model articulates a minimal but necessary set of features for knowledge based authoring tools that aim for scope, depth, learnability, and productivity.

1. INTRODUCTION

Intelligent tutoring systems. Intelligent Tutoring Systems (ITSs), also called knowledge-based tutors, are computer-based instructional systems that have separate data bases, or knowledge bases, for instructional content (specifying what to teach) and teaching strategies (specifying how to teach), and attempt to use inferences about a student's mastery of topics to dynamically adapt instruction. ITS design is founded on two fundamental assumptions about learning. First, that individualized instruction by a competent tutor is far superior to classroom style learning because both the content and style of the instruction can be continuously adapted to best meet the needs of the situation (Bloom 1956). Second, that students learn better in situations which more closely approximate the situations in which they will use their knowledge, i.e. they "learn by doing," learn via their mistakes, and learn by constructing knowledge in a very individualized way (Bruner 1966, Ginsburg & Opper 1979). Individually paced instruction and frame-based computer aided instruction (CAI) comprised early attempts to provide adaptive instruction, and, though successful for some types of learning, fell short because their learning environments were too contrived and their ability to adapt was limited to branching between static screens. ITSs use techniques that allow automated instruction to

Murray, Ainsworth, & Blessing (eds.), Authoring Tools for Adv. Tech. Learning Env.

come closer to the ideal, by more closely simulating realistic situations, and by incorporating computational models (knowledge bases) of the content, the teaching process, and the student's learning state (Wenger 1987).

The need for ITS authoring tools. In the last decade ITSs have moved out of the lab and into classrooms and workplaces where some have proven to be highly effective as learning aides (see Chapter 17 in this volume). While intelligent tutoring systems are becoming more common and proving to be increasingly effective, each one must still be built from scratch at a significant cost. Little is available in terms of authoring tools for these systems. Authoring systems are commercially available for traditional CAI and multimedia-based training, but these authoring systems lack the sophistication required to build intelligent tutors. Commercial off-the-shelf (COTS) authoring systems excel in giving the instructional designer tools to produce visually appealing and interactive screens, but behind the presentation screens is a shallow representation of content and pedagogy.

Pedagogy-oriented tutors. A fundamental aspect of intelligent tutors is that knowledge about the domain and knowledge about how to teach is stored in modular components that can be combined, visualized and edited in the process of tutor creation. I use the term "knowledge based tutors" to highlight this aspect. In Chapter 17 I claim that intelligent tutors can be categorised into two broad groups, pedagogy-oriented and performance-oriented (though some systems fall in a grey area between these two):

> Pedagogy-oriented systems focus on how to sequence and teach relatively canned content. Most of them pay special attention to the representation of teaching strategies and tactics. Performance-oriented systems focus on providing rich learning environments in which students can learn skills by practicing them and receiving feedback. Most of them pay special attention to the representation of human problem solving skills or domain-specific processes or systems (either man made ones such as electrical components, or natural ones such as the meteorology). In general, performance-oriented systems focus on feedback and guidance at the level of individual skills and procedural steps, while pedagogy-oriented systems pay more attention to guidance and planning at a more global level, looking at the sequence of topics and the need for prerequisite topics.

This paper focuses on pedagogy-oriented tutoring systems. The suggestions presented in this paper apply to performance-oriented tutors also, but will be useful only to the extent that they model the conceptual structure of a domain, represent curriculum, and/or incorporate teaching strategies (as do most pedagogy-oriented systems). In Chapter 11 I describe the Eon knowledge based tutor authoring system. This Chapter summarizes some lessons learned from our experiences in the Eon

project, and describes a generic framework or "reference model[1]" called KBT-MM (knowledge based tutor meta-model) for knowledge based tutor authoring tools.[2] The reference model proposes a minimal but necessary set of features for knowledge based authoring tools that aim for high generality, usability, flexibility, and power (see Chapter 17 for a discussion of these terms). It specifies representational and authoring features that systems should have, but not how these features should be implemented.

The Chapter begins by discussing some general issues that characterize knowledge based tutors in comparison with traditional CAI systems. I characterize this difference as an evolution from a story-board metaphor to a knowledge based metaphor for representing instructional content. I explain that an important characteristic of knowledge based tutors is that they are designed using building blocks at a higher more expressive level of abstraction. I call this "designing at the pedagogical level" (vs. at the media level).

The Chapter then describes the KBT-MM, beginning with a conceptual model of the primary objects needed in the pedagogical domain model (including topics, lessons, and contents), their attributes and how they relate to and interact with each other. This model is called the "five layered decision architecture." Then we describe general features of student models and teaching models within the KBT-MM framework. Next, I discuss how Ontology objects are used to specify the domain-specific details of a tutor and allow for the development of special purpose authoring systems. Finally I will discuss lessons learned and some of the problematic areas of representing pedagogical knowledge in knowledge based tutors.

2. FROM COMPUTER-BASED TO KNOWLEDGE-BASED INSTRUCTION

How an authoring tool is designed depends critically on who the intended user audience is. I begin by clarifying our target authoring audience and design team model, and then describe the types of design paradigm shifts needed to build ITSs.

2.1 Who Are the Users of ITS Authoring Tools?

In Chapter 17 Section 5.3 is a discussion of what skills are needed for using authoring tools of different types. As discussed in that Chapter, many ITS authoring systems simplify the authoring process by either making some aspects of the ITS non-authorable, or by constraining the type of ITSs that can be produced. For this Chapter our goal is support the authoring of all aspects of an ITS: domain model, teaching model, student model, and interface. The skill set needed to do this dictates that an ITS will usually be built by a team rather than an individual. The software

---

[1] Our term meta-model is similar to "reference model/reference architecture" mentioned in Schoening & Weeler (1997) and Betz et al. (1997).

[2] The reader may read either Chapter first, depending on whether one finds it most perspicuous to learn about generalities or a particular example first.

we are picturing has a complexity on the order of magnitude of Photoshop, AutoCAD, or FileMaker, i.e. it would take more skill or training than is usually associated with tools like word processors and PowerPoint. Building an ITS from scratch in this way requires a significant time commitment on the part of a teacher or domain expert. The "master" teachers or trainers who become ITS authors will have to be able to invest significant time building the systems and invest additional start-up time on the learning curve for these sophisticated tools. Our goal here is not to lower the skill threshold so far that the average classroom teacher can author an ITS (but they may be able to customize the ITS, see Section 3.7) but lower it to a point where every company and every school district could have at least one team capable of ITS authoring. These teams could work with teachers, subject matter experts, and graphical artists to rapidly produce ITSs.

By providing visualizations of key concepts and components in the ITS, authoring tools could make ITS authoring accessible to hundreds of thousands of individuals rather than a few hundred. We will call the lead person on an ITSs design team, the one who synthesizes knowledge of the domain, teaching strategies, and the requirements of the authoring system, the "instructional designer." The goal of the KBT-MM framework is to facilitate cost-effective ITS production by these instructional designers, as well as by those who have traditionally built ITSs from scratch (primarily those in academic and industrial research labs).

2.2 From Story Boards To Knowledge Bases

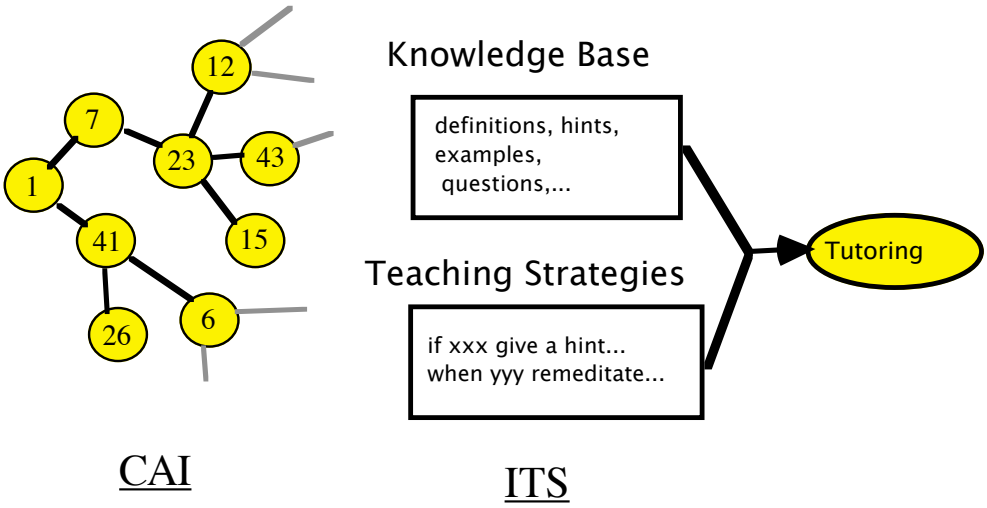

Figure 1: CAI story board vs. ITS knowledge base

Though there are few ITS instructional designers, there are many CAI instructional designers. Empowering these users to build more powerful instructional systems

requires new tools *and* a shift in the way many of them conceptualize instructional systems. Specifically, it is proposed that moving from CAI authoring to ITS authoring involves a fundamental paradigm shift from "story board" representations of instructional material to more powerful and flexible "knowledge based" representations. The basic concept is not new; in fact, it is fundamental to all AI work. Our contribution is in fleshing out how the knowledge based paradigm can be best presented to empower instructional designers.

Commercially available authoring systems assume and support a representation of instructional content and instructional flow that is explicit, non-modular, and fairly linear. At a conceptual level, the instruction is specified like this (see Figure 1): "Bring up screen # 41; If the user clicks on button A, then go to screen # 26." Though branching is allowed, each branch must be explicitly specified. Adding a new topic or question involves explicitly encoding the branches to this content. Designing new content requires a duplication of efforts. I call this paradigm for designing instructional systems "story boarding" because it is based on enumerating all of the screens and the explicit links from each screen to the next. In contrast, in a knowledge based tutor the instructional content is separated from the specifications of how and when that content is presented to the student, so that the content can be used and re-used in multiple ways. Specifying how and when the content is to be presented is done with generic, reusable teaching strategies. For example, a CAI system may be programmed to give two hints for wrong answers to exercises. If the author later realizes that 3 hints are necessary, she has to go back and change every link associated with giving hints. In contrast, in the knowledge based model, there is one strategy specifying how and when hints are given, so that changing from 2 to 3 hints is a matter of making one change.

Designing at the Pedagogical Level. In traditional CAI instructional actions are encoded using building blocks at the level of the media: text, pictures, button clicks, etc. In contrast, knowledge based tutors can facilitate the design of instructional actions using pedagogically relevant building blocks. For example: "give a hint," or "teach the prerequisites". Designing instruction using building blocks such as "hint," "prerequisite," "if-confused," "mastered," "explanation," and "summarize" is much more powerful than designing instruction at the level of "show video," "present picture" or "wait for the button click" The instructor can conceptualize the curriculum at a more appropriate and powerful level of abstraction. An instructional strategy in the intelligent tutor might be: "if the current topic is conceptual and the student is doing poorly, give several examples." Alternate strategies can be created, so that the appropriate strategy can be used according to the needs of the student (e.g. learning style or mastery of the current topic) or the pedagogical characteristics of the content being taught (e.g. whether it is procedural or conceptual information).

Benefits of knowledge based tutoring. Designing tutoring systems in this way has many advantages over the traditional CAI design paradigm:

1) The behavior of the tutor can be **easily modified**. As shown above, to change when hints are given, only a single "give hint" strategy needs to be

changed and this effects how hints are given through the entire curriculum. This is true for the author making this change and also for a "meta-strategy" that sets this parameter during run time.

2) The content of the knowledge base is modular and can be **used for several purposes**. For example, a “topic” object in the knowledge base can contain information about how to teach itself, summarize itself, give examples of itself, introduce itself, and test the student's knowledge of itself. The same topic can then be used in many parts of the tutorial, for example, giving a summary at some point and teaching itself later on.
3) Systems designed in this way can be more **adaptive** to the needs of the student.
4) Instruction can be much more **learner-centered**, since modularization allows students to navigate to the topics they want to learn, and to ask for hints, examples, etc.
5) Because the content is modularised, authoring tools can be built to provide multiple and abstracted views of the instructional content. This facilitates instructors in being able to **easily view, inspect and navigate through the knowledge bases**.

3. A KNOWLEDGE-BASED TUTOR META-MODEL

3.1 Focussing on the Representation of Pedagogical Knowledge

Authoring tools can be described in terms of two aspects: an underlying representational framework (conceptual model), and a user interface (the authoring "tools") that reify this framework for the user, allowing her to create, visualize, and modify the elements of an ITS. Thus, the end usability and power of an authoring tool depends on both the power and fidelity of the underlying conceptual model and also on the usability of the interface tools. The conceptual model is of primary importance because if it is insufficient the best that the interface can do is to skillfully reify a weak model. Therefore, one goal of this Chapter is to define a conceptual model for knowledge based tutor authoring tools that aims for high power, generality, flexibility, and usability, based on what we see in a variety of other ITS authoring systems. The architecture is a "reference architecture" that illustrates the key distinctions necessary for a knowledge based tutor, but is not meant to constrain how these distinctions are implemented in a system. Rather than specify a particular design, KBT-MM prescribes properties that we think all knowledge-based authoring tools should have. It is a base-line model that contains a minimal but necessary set of features and elements, rather than (a baroque) model that tries to include most of the features seen elsewhere. In some instances it describes *innovations* to what is found in most ITS authoring tools, but in most aspects it is a systematic *synthesis* and abstraction of the important elements found in other systems. The Eon authoring system, described in Chapter 11, is one example of many possible implementations that conform fully with our description of the KBT-MM.

As is commonly done, we can describe the knowledge encoded in an ITS as being either domain knowledge or teaching knowledge. However, some of the knowledge in an ITS is in both categories. For example, hints and prerequisite relationships are usually considered part of the domain knowledge base, yet they are relevant only to the teaching of a subject (not to performance in the subject). As shown in Figure 2, I define a domain's "pedagogical knowledge" (sometimes called propeadutic information) as information specific to a domain that is relevant to teaching about the domain. As shown in the figure, some domain knowledge is relevant to performance but not necessarily used in instructional decisions. This "performance expertise" usually comprises the expert system rules (or production rules or procedures) in performance-oriented systems. Teaching knowledge can also be categorized as either teaching strategies, the (usually) domain independent strategies indicating how to teach; and pedagogical knowledge, the domain dependent declarative knowledge mentioned above. Performance-oriented systems tend to have non-existent or degenerating teaching strategies, and pedagogy-oriented systems tend to have non-existent or degenerative performance expertise.

The purpose for clarifying these distinctions is that KBT-MM focuses on the representation of pedagogical knowledge. Pedagogical knowledge is declarative information (whether "canned" or generated) such as examples, hints, explanations, definitions, problem descriptions, and problem answers. It includes the definition of topics and their relationships (e.g. prerequisite, generalization) that are needed to sequence instructional units. It also includes the classification of knowledge (or topics) according to knowledge type (e.g. fact, concept, principle) for the purpose of instruction.

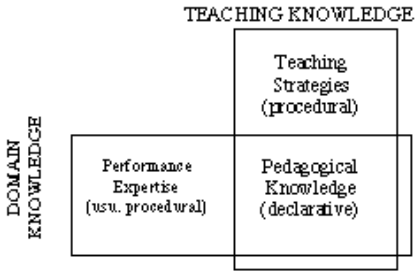

Figure 2. Teaching and Domain Knowledge

Traditionally, ITSs are described as having four major components or functions (Wenger 1987): a domain model, a teaching model, a student model, and an interface. Above I have mentioned the domain and teaching models. KBT-MM has less to say about the representation of the teaching strategies. This is because there is significant overlap and agreement (though nothing approaching consensus) among numerous ITSs, ITS authoring tools, and instructional design theories, on how content and curriculum knowledge should be conceptualised, while there is very little agreement on how to represent teaching strategies. As indicated above, the

representation of expert problem solving knowledge is outside the scope of KBT-MM. KBT-MM does not specify details about a general framework for student modelling. This is because, at the knowledge representation level, student models are a rather straightforward "overlay" assignment of student knowledge values (or user history or performance metrics) to elements in the domain model. The algorithmic method of *inferring* the student state from student behavior is much more complex and idiosyncratic. The representation of the student model follows in a straightforward way from the representation of the domain model, plus a consideration of the data types needed to store the outputs of these inference methods. For example, a system that uses Bayesian inferencing methods will have a student model representational formalism tailored specifically to this method. Inferencing methods whose calculations use the number of hints given in each problem will of course have to store this information for each problem solved.

3.2 Abstract Topic Objects

Intelligent tutors make inferences about what knowledge or skills the student has. As such they differ fundamentally from traditional CAI in that they deal with abstract entities assumed to exist in the minds of students, in addition to the concrete content presented to the student. We call these entities "topics." They are called "instructional units," "domain knowledge elements," "knowledge units," "cognitive rules" etc. in other systems. Expository content, stored as text or media, is "presented," and inquisitory content (such as questions and tests) can be "correct," "answered," "passed" etc. In contrast, topics are "known," "understood," etc. In KBT-MM instructional units of any grain size are represented by Topics. Brusilovsky (in this Volume) notes that in hypermedia authoring there is a parallel distinction between the knowledge space and the hypermedia content space. Topics have a **Topic Type** that allows them to be categorized. Topics can have **Topic Properties**, such as difficulty and importance, as needed. **Topic links** define directed relationships between topics. Since one can define different types of topics and create arbitrary relationships between topics, this single object suffices for many representational schemes. That is, the topic object can be used to represent different types of units (such as concepts and facts) and can be used to represent different hierarchical or containment levels of units--what might be called chapters, sections, or sub-topics.

For a given tutor the vocabulary of topic types and link types is defined in an "Ontology" (see Section 4.5). Customizable link types allow the representation of a wide variety of topic networks, including component hierarchies, skill lattices, concept networks, etc. Each knowledge type defined in the Ontology has its allowed properties, allowed link types it can connect to, and allowed topic levels. Buggy Knowledge can be represented as topics of type Misconception, Procedure Bug, etc., depending on the type of knowledge that is in error.

3.3 A Layered Curriculum Object Framework

As I have mentioned, our goal is for KBT-MM to have a minimal but necessary set of features for KBT-authoring tools that promotes power, usability, generality and flexibility. From an informal analysis of pedagogy-oriented ITSs and ITS authoring tools, I propose a five layered "decision architecture," illustrated in Figure 3, as a general framework or reference architecture that illustrates common pedagogical components of these systems. The architecture specifies, at a conceptual level, the five main types of pedagogical objects and shows how control is mediated among them as a simple and necessary result of their nature. The Topic, Presentation Contents, and Events layers are common to all knowledge based tutors (though they may be called different things and implemented in various ways). The Lesson and Topic Level layers are less common and are included to satisfy the requirements of power and generality. Next I describe the five layers in an order that facilitates explanation of their purpose.

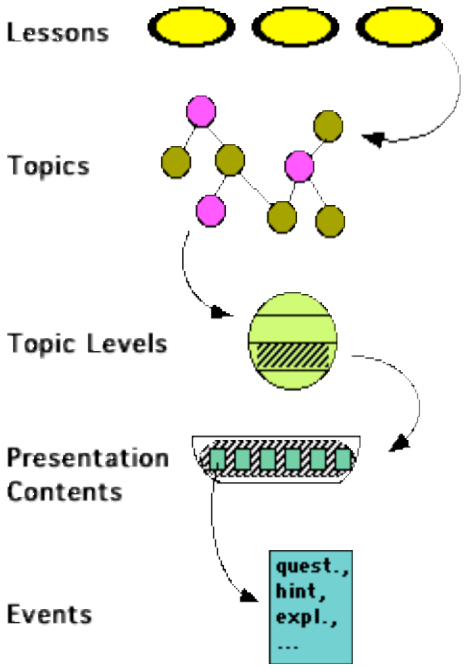

Figure 3: Five-Layer Decision Architecture

Events. Events are the low level interactions between the student and tutor, such as a student clicking a button, or the tutor giving a hint, several of which will occur while a Content is running.

Presentation Contents. The Presentation Contents layer contains the specific concrete contents that the student will see and manipulate (text, graphics,

buttons, interactive screens, etc., or the templates or algorithms for generating this content). Contents are expository or inquisitory interactions.

Topic Levels. As mentioned above, topics are related to each other via topic links to form hierarchies or networks. Having only the semantic network formalism to represent all aspects of curriculum structure was found to be inadequate. In some domains the most perspicuous structure for representing aspects of the content may be a table rather than a network. In our Eon system we found that Levels within topics allowed us to represent 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. This layer of the architecture allows for a structural layer within each topic object. As shown in the Figure, Presentation Contents are referenced from inside the topic levels. (Section 5.4 describes uses of topic levels in more detail.)**Topics**. Topics were defined above. They are specialized according to Topic Types, specified with Topic Properties, and related with Topic Links. Topic networks do not specify any ordering or starting point for learning sessions, and are independent of the instructional purpose of particular sessions.

Lessons. Lesson objects are used to specify instructional goals and learning/tutoring styles for a particular group of students or learning session. Nominally, the Lesson lists a one or more starting or goal topics, and specifies a default teaching strategy. The teaching strategy then determines how the topic network will be traversed, causing other topics to be taught, to satisfy the goal of learning the goal knowledge.

The nature of the objects in the Decision Architecture implies the following general control structure: running a Lesson runs a number of Topics, each of which runs some of its Topic Levels, each of which contains a number of Presentation Contents, each of which leads to a number of Events. In the next Section we discuss how previous research and theory informs the KBT-MM model as described so far.

3.4 Previous Work in Representing Pedagogical Knowledge

This Section focuses on what we have learned about representing pedagogical domain knowledge, including elements from various instructional theories and authoring tools, and use this as evidence for the validity and generality KBT-MM described above. We will make reference to the following authoring tool systems, references to which can be found in Chapter 17: XAIDA, REDEEM, Eon, DNA, Expert CML, Instructional Simulator, IDLE, IRIS, CREAM, SimQuest, and Eon.

A. Modular abstract knowledge units. As mentioned in Section 3.2, intelligent tutors represent units of knowledge to be learned, which we call topics, as modular units separate from content and instructional strategy. All of the principles and theories mentioned below implicitly or explicitly assume that content can be modularized to organize the learning. (However, in Section 6.3 I discuss some problems inherent in knowledge modularization.) Of the authoring tools described

in the Overview Chapter (Chapter 17), every one of them uses modular knowledge units, except for a couple of special purpose authoring systems for tutors that do not reason about what to teach next. Authoring tools use different schemes and use different names for instructional units. REDEEM has pages and sections. Expert CML organizes domain knowledge in a hierarchy of objects including Departments, Programs, Courses, Topics, Subtopics, Modules, SubModules, Objectives, and Activities.

B. Types of Knowledge. Researchers in computer science, psychology and educational theory have developed many schemes for classifying knowledge. VanLehn (1987, pg.60), speaking from an AI perspective, says that the popular procedural/declarative distinction is "notorious...as a fuzzy, seldom useful differentiation." It is recommend that the procedural/declarative distinction be abandoned for classifying knowledge in instructional systems (except in contexts where it has a precise meaning, as in the ACT* theory of cognition (Anderson 1983)) and that more descriptive and precise schemes be used.

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) classifies learning objectives (content types) as facts, concepts, procedures, or principles. In contrast to the hierarchical typing schemes of Bloom and Gagne, Merrill's content types are organized into a matrix along with performance types (explained later). 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. Reigeluth's Elaboration Theory of Instruction (1983) is another complex knowledge typing scheme. It builds upon Merrill's theory for how to teach individual units of knowledge of different types, and goes further to proposes a theory of how these units can be organized and taught within entire domains, which require knowledge of many types.

Topics ("basic learning units") in IRIS use Merrill's typing scheme: facts concepts, procedures, principles. DNA organizes topics ("curriculum elements") into facts (symbolic and episodic knowledge), procedures, and concepts. CREAM uses Gagne's system: verbal information, intellectual skills, cognitive strategies, motor skills, attitudes.

C. Hierarchies. latices, and networks. A number of educational theories mention the hierarchical nature of knowledge. For example, Ausubel's "subsumption theory" of learning (Ausubel 1960) focuses on the hierarchical organization of concepts in disciplines. 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 (e.g. prefer more closely related topics) and generality (e.g. give generalities before specifics). The subsumption relationship is valid for conceptual learning, but pedagogical knowledge for procedural or skill learning can require 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. Knowledge is often messier than can be represented in a simple hierarchy or lattice and network representations are needed. Cognitive science has shown that cognition has network-like aspects (Collins & Loftus 1975). Goldstein's (1982) ITS uses a Genetic Graph with 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.

Most of the authoring systems incorporate hierarchical or network-like topic representations. They differ in the types of topics and topic relationships used. Brusilovsky (this Volume) says that most adaptive hypermedia systems use "is-a" and/or "part-of" relationships. LAT, REDEEM, and RIDES use simple hierarchical representations with one of two relationship types. "Instructional units" in RIDES are in a hierarchy of sub-tasks. XAIDA has associative information (facts) related as "subparts" in the "Physical Characteristics" shell; casual reasoning relationships in its "Theory of Operation" shell, linked procedural steps in its "Procedures" shell, and a discrimination net (fault tree) in its 'Troubleshooting" shell. Within the Physical characteristics shell parts are related using: part-of, function, location, and connected-to. DNA incorporates hybrid network structure that borrows from classic semantic networks and GOMS production rule networks to represent several knowledge types and relationships in a unifying formalism (it also seems to be the only system that has strengths or weights associated with its topic links). Its relationship types include: procedural-part, next-step, conditional-step, causes, and part-of. IRIS uses these relationships between its learning units: prerequisite, procedural (next-step, if/then decision), conceptual (is-a and part-of); theoretical (cause-effect) and precondition. CREAM uses separate networks for three classes of objects. "Capabilities" (similar to our "topics") are related with these relationships: analogy, generalization, abstraction, aggregation, and deviation. "Objectives" are related using: mandatory prerequisite, desirable prerequisite, supporting (or "pretext"), and aggregation. Resources are related using: analogy, abstraction, case, utilization, assistant, and equivalence.

Our goal in KBT-MM is to have a meta-level system that is compatible with all of the above, not a specific system that incorporates all of the topic types and link types.

D. 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.

Relatively few authoring systems seem to incorporate misconceptions and bugs. XAIDA stores "misconception" facts. Ontologies developed for the Eon system use misconceptions, procedural bugs, and erroneous facts. Model Tracing Tutors (see Ritter et al. in the Volume) store buggy procedural rules.

E. Beyond networks to more structured knowledge. Above we argued for the need to include an additional level of (or levels) of structure beyond network representations. We proposed Topic Levels as such a structure, but there are many possibilities (the Topic Level idea is an inclusive and general framework, but some systems use schemes too complex to be incorporated into Topic Levels.). Topic Levels allow the tutorial to distinguish particular levels, methods, or modes of teaching within the topic. We describe two uses for Topic Levels seen in authoring tools: task levels and behavioral objective levels.

To distinguish task levels IDLE breaks a an inquiry topic or problem into these steps: learning the problem context, gathering information from sources, gathering data from instruments, abstracting and manipulating data, drawing conclusions, and communicating conclusions. SimQuest allows for several 'assignment types:" performance, investigation/explaination, specification, and optimization. Instructional Simulator has templates specifying different types of patterned exercises: describe parts, identify and locate parts, demonstrate ("Simon sys" method), practice, and perform. XAIDA has 11 such exercise templates, and RIDES has 25.

Several systems (IRIS, DNA, CREAM, Eon, Instructional Simulator, and XAIDA) incorporate an important principle from instructional design theory: that a subject matter can be learned at several levels corresponding to different types of behavioral objectives. For example consider the procedure for starting a car. We can distinguish at least three types of objectives: the procedural steps can be memorized, the procedural skill can be mastered, and the purpose and causal relationships of its components can be understood. We can distinguish "topics" such as "starting a car" from the types of objectives. For example, Merrill's content matrix (Merrill 1983) incorporates "performance levels" called remember, apply, and create. This is handled in different ways by different authoring systems, but overall it calls for an additional level of structure beyond the network. In CREAM concepts can be identified, recognized, classified, and generalized. IRIS uses the levels defined by Bloom as described above.

Again, our goal is to propose an overall structure (Topic Levels) that is compatible with most (but not all) of the schemes described. In KBT-MM Presentation Contents are assigned to the Topic Levels within each topic, not directly to the topics. This is compatible with the authoring systems mentioned above.

F. Knowledge attributes. Several theories point to the need to assign pedagogically relevant attributes to topics, such as importance and difficulty. 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. 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. Many ITSs incorporate topic attributes into tutoring strategy decisions. Adding attributes to topics gives them internal structure like "frames" or "schemes" in AI knowledge representation.

Brusilovsky (in this Volume) notes how frame-like knowledge representations are used in many adaptive hypermedia systems to represent internal topic structure. The components or parts in Instructional Simulator, RIDES, and XAIDA have properties associated with them. REDEEM and other systems include topic difficulty levels.

G. Lessons and instructional objectives. 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 that captures the different "viewpoints" of a curriculum structure that result from different instructional goals (or perspectives).

Leinhardt 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.

For most systems, including adaptive hypermedia systems, the learning goals are a sub-set of the topic or concept network. RIDES and Eon store goals in lesson objects. IRIS, and CREAM represent learning objectives in terms of behavioral objectives, and described above. Van Marcke's (1992) GTE framework makes a similar distinction between content and instructional goals.

The above principles support the following prescriptions which are used in the KBT-MM framework. Separate "what to teach" into *modular units* independent of how to teach it (item A above). Use *network formalisms* for knowledge units with directed links allows for the creation of hierarchies and lattices and less canonical frameworks (C). Include a number of different *types of nodes and links* (B). Knowledge units should have *pedagogically relevant properties* associated with them (F). *Learning goals* for a tutorial session should be represented separately from instructional content (G). ITSs should be able to distinguish among different *types of knowledge* (B), and also represent *buggy knowledge* (D), so that teaching strategies can be predicated on knowledge classes. There is an indication that exclusive use of a network formalism does not provide enough knowledge structuring complexity, and that *more complex data structures* (such as Topic levels) will often be needed to support multiple levels or modes of instruction for each topic (E).

3.5 Student Model and Teaching Strategies in KBT-MM

Theories from instructional design and cognitive psychology have much to say about the declarative organization of content and knowledge, and this has influenced ITS design. In contrast, though there are many prescriptions for how and when to present content, there is little agreement nor general understanding of how procedural teaching strategies should be formally represented or implemented.

The inference method used by the student model is left unspecified in KBT-MM, but the Decision Architecture does suggest some constraints on what types of information the student model should represent. The student model can include values for objects at all five layers of the architecture (as in Eon's student model). That is, it can store an overlay value for lessons, topics, each level within a topic, presentations, and events. A given implementation can opt out of including some layers in the Student model, for example, the simplest models will include only Topic values (and perhaps will also save the history of all user interactions at the Event layer). For the objects in each layer the student model can have one or more values, for example: mastery, confidence level, was-attempted, was-shown-to-the-student, number-given, etc. KBT-MM specifies the five layers, but leaves the particular vocabulary of value types for each object to the Ontology. The value of objects at each level is a function of the values at the next lower level. For example, a Topic's values are a function of the values of its topic levels.

Teaching strategies are part of the "control structure" of an ITS-- the algorithms or rules that determine how the ITS will behave and respond to the student. Embedded in this control structure will be numerous individual decisions related to deciding what, when, and how to teach. KBT-MM leaves it up to the implementation whether the control structure is, for instance, blackboard-based or production rule based. In any case, at an conceptual level we can say that there are many decisions to be made in the form of "IF X then Y" rules. The antecedents inspect internal states, which must be about the student (e.g. IF the student understands the current topic…), the history of the session (e.g. IF greater than 6 hints have been given for the previous three activities…), or the nature of the current content (e.g. IF the current topic is a difficult fact…). The bulk of the consequents are tutorial actions (e.g. …THEN give a hint, THEN begin the next topic, THEN show the topic summary) or student model inferences (e.g. THEN increase student model confidence for mastery of the current topic level).

Thus, the vocabulary of terms used in the antecedents and consequences of teaching strategies come from the decision architecture. The vocabularies for student model values and domain model details that are defined in the Ontology. It is possible and preferable to design authoring tools such that tutoring strategies reference these and only these pedagogical building blocks. I.E. no new objects or terms need to be introduced beyond the framework described in the Sections above.

3.6 Ontology Objects

As explained in Chapter 17, the goals of usability, flexibility, and power are often at odds with each other, since systems tailored for specific domains can include

powerful non-generic representational and instructional methods. Attempting to create a single ITS framework that includes the capabilities of many existing systems would, at the least, result in a very complex and obscure system. In contrast, we have tried to identify a minimum underlying framework that is neutral regarding domain or instructional theory. KBT-MM includes a highly generic, yet underspecified, Layered Decision Architecture that has many aspects left open to particular implementations. Ontology objects specify the remaining complexity for each implementation. The decision architecture specifies the structure of objects and control. Ontologies in KBT-MM are simply sets of terms that specify the types, levels, or attributes that are allowed for the objects in a particular incarnation of the KBT-MM.

In general, an ontology is a particular way of describing the world (or some domain); it is a scheme for conceptualizing the objects and relationships in a domain (Gruber 1993).[3] I use the term "ontology object" for a data object which defines a conceptual vocabulary for the system. Relating this to the Decision Architecture above, an ontology could specify the following: lesson properties, topic types, topic link types, topic properties, and topic levels. The ontology could also specify the types of values used by the student model, as described above. As an example of Ontology use, consider the ontology that was created for the Eon Statics Tutor. It defined topic types Fact, Concept, Procedure, and Misconception, and topic links Prerequisite, Generalization, and SubConcept. A (fictitious) tutor for Manufacturing Equipment might have topic types Safety, Maintenance, Operation, Theory, and Common Failures, and topic links SubPart and SimilarPart. Ontologies can be generic and reusable, for example, an ontology developed for one science tutor should be usable (perhaps with slight modifications) for other tutors with similar pedagogical characteristics (e.g. instruction at a predominantly conceptual level).

Most tutoring systems fall into one of a number of loose classes, each addressing specific types of cognitive skills or knowledge types. Example domain classes include: conceptual information, factual information, problem solving skills, design skills, procedural skills (such as maintenance), inquiry and experimentation skills, equipment diagnostic skills, customer contact (and other interpersonal) skills, sensory-motor skills, association and pattern recognition skills, and argumentation/hypothesis generation skills. Ontology objects may be reused across tutors within a domain class. Reigeluth (1983) prescribes that each domain be assigned an "organizing content type:" conceptual, theoretical (principle-like), or procedural, that best fits the characteristics of the domain and the instructional goals. His "elaboration theory of instruction" specifies methods for selecting and sequencing content according to the organizing content type. Others have categorized domains according to whether their structure is predominantly

[3]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 focusing on ontologies that support knowledge authoring rather than sharing, and we focus on pedagogical knowledge, where the KSE deals with performance knowledge.

procedural, historical, structural, causal, teleological, inferential, etc. Domain types have characteristic links between topics, for example analogy, physical-part, a-kind-of, etc.[4] Classifying domains according to organizing content type, and creating ontology objects for each organizing content type, would help bootstrap ITS construction.

Several other ITS projects address ontologies. Some of these, for example (e.g. (Mizoguchi et a. 1996) and (Van Marcke 1992)) have a goal to define a generic conceptual vocabulary that can be used do describe describing all of the objects, attributes, and methods needed to design a tutor. The scheme usually includes hierarchical sets of ontologies, with a core ontology defining terms generic to all instruction, and set of domain-specific ontologies (for example, for mathematics or automobile maintenance) to be loaded on top of the core ontology. Some projects also include authoring tools for developing and extending these ontologies, Our use of ontologies is different than, yet compatible with, these efforts. In KBT-MM the Ontology is an open structure that allows the designer to define a vocabulary for a particular tutor. The goal of these other projects is to define a complete vocabulary that is independent of any specific implementation--it is almost a conversational vocabulary defining the concepts needed to define or describe an ITS and its behavior. In the case of KBT-MM the Ontology framework is tied closely to the architectural framework, as its purpose is to allow the author to specify implementation-specific attributes of the primary objects defined by the architecture (topics, presentations, student models, etc.). The two methods are compatible because, once other research teams successfully define vocabularies, elements of these vocabularies can and should be used for KBT-MM Ontologies (especially if the vocabularies are standardized or in common use).

Moving the discussion from representational frameworks to authoring tools, an authoring tool that conforms to KBT-MM needs a tool for defining the ontology. In authoring an ITS an ontology must be defined first, and then the authoring tools must adapt to the specifics of the ontology. For example, if the ontology specifies that the allowed knowledge types for the tutor are Facts and Principles, then the tool for creating topics should only allow these two types of topics to be created.

3.7 Meta-Authoring Special Purpose Authoring Systems

Even with tools that are relatively generic, powerful, and usable, the process of starting from scratch to build an ITS is still a formidable one. Not surprisingly, in our research with the Eon system we consistently ran up against the classic knowledge acquisition bottleneck (Hoffman 1987). Domain experts are usually good at sketching out student interactions, lessons to teach particular topics, and responses to specific student behaviors, but articulating knowledge at a more abstract or generic level is difficult. The following ITS knowledge representation tasks were inherently difficult for most subject matter experts:

---

[4](Wenger 1987, pg. 331) describes several "justification types," such as structure, functionality, and constraints, that could be used as topic links.

1. **Ontology design**. Defining the types of topics, topic links, and topic levels for the Topic Ontology, and also defining the ontology of allowable values for the student model.

2. **Curriculum representation.** Breaking the instructional material and goals up into discrete components (topics) and providing relationships between these components (topic links);

3. **Strategy and diagnostic procedure representations.** Representing teaching strategies and procedures in a general way (e.g. how do we recognize that a student is confused, and what is a reasonable general response to student confusion?);

4. **Student model definition.** Defining rules that express when a student knows a topic, and labeling or characterizing the student's level of knowledge.

In our experience, it takes a knowledge engineer, a person skilled in the elicitation and representation of these types of knowledge, to work with the subject matter expert (the teacher) to build these aspects of the system. Highly usable authoring tools help, since they help the teacher visualize the information and participate more intimately in design process, once the knowledge engineer has broken the ice and explained things once or twice.

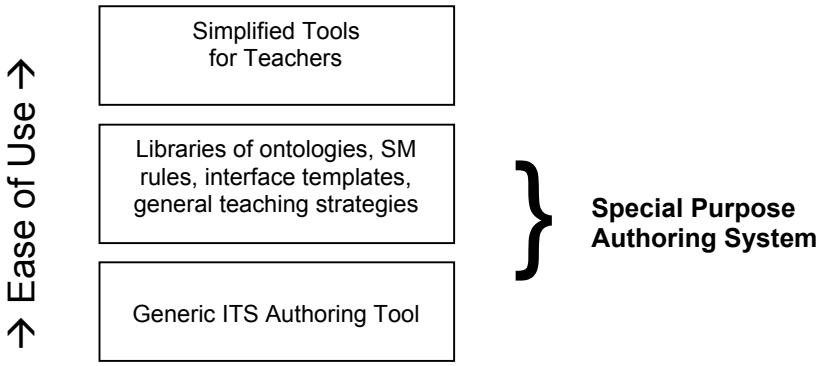

Figure 4: Three Tiered Suite of Authoring Tools

One solution to the knowledge acquisition problem is creating special purpose authoring tools, for example authoring tools for building ITSs that teach anatomy, foreign policy, or verb conjugation. Authoring shells which are used to build tutors for specific task types (Jona 1995, Sparks et al. in this Volume, and Bell in this Volume) can, in principle, build tutors with more fidelity and depth than general purpose tools. The depth vs. breadth tradeoff seems to imply that you can have one but not the other, that 1) ITS authoring tools that can build powerful tutors that closely match the pedagogical needs of a domain must have a narrow scope, and that 2) an all-purpose ITS shell, by necessity, must have a shallow knowledge representation and the learning environment it creates will have little conceptual fidelity (in comparison to special purpose tools).

Our approach is to build special purpose authoring systems on top of the generic authoring system, so that the authoring tool becomes a "meta-authoring tool." This would involve constructing libraries of pre-built components such as ontologies, student model rules, and interface screens. Since some teaching knowledge is general in nature (Van Marcke 1992, Jona 1995), default teaching strategies and meta-strategies could also be provided. We could then provide sets of these components tailored to facilitate building tutors for classes of domains or tasks. For example, ITS authoring shells could be produced for science concepts, human service/customer contact skills, language arts, and equipment maintenance procedures. Instructional designers could immediately start constructing tutors in an environment that supports and helps structure the knowledge acquisition process.

By using a meta-authoring approach, we hope to achieve a fair degree of fidelity and depth, while maintaining usability and generality. Figure 4 illustrates the potential for a three-tiered suite of authoring tools, using the Eon system as the example base authoring system. At the first tier is a general purpose ITS authoring system that requires moderate knowledge engineering and instructional design expertise to use. At the second tier are special purpose ITS authoring systems (built on top of the first tier system) that require minimal knowledge engineering and instructional design expertise. The third tier involves tools for the average teacher using an ITS in her class. At the third tier are a simplified subset of the authoring tools, so that once an ITS is built, *any* teacher can customize it for a particular class or group. For example, by modifying a hint's text, replacing a picture with a more recent version, making a teaching strategy more verbose, or by changing a prerequisite relationship between topics. This is important because some teachers will be reluctant to use instructional systems that they can't understand or adapt.

4. SUMMARY OF KBT-MM

Below I will summarize our recommendations for the design and use of knowledge based authoring tools using the KBT-MM conceptual model.

4.1 Domain and Curriculum Model

The fundamental conceptual entities in an ITS are the curriculum objects. The representational framework should allow for, and the authoring tools should reify, these objects in accordance with the following principles:

- Represent instructional (teaching) **strategies separately** from domain knowledge (i.e. use a "knowledge based paradigm").
- Design for **modularity** and re-usability of content. I.E. whenever possible content should be independent of the way it is used, and be able to be used in multiple contexts.
- Provide a representational formalism that is **customizable and extensible**. The vocabulary of properties and building blocks used to conceptualize and

build a tutor should reflect the structures and assumptions inherent to the particular domain and/or instructional style. KBT-MM does this through the **Ontology** object. Below when the specification calls for customizability, the tutor-specific aspects are intended to be specified in the Ontology of that tutor.

- Explicitly represent abstract pedagogical entities-- referring the knowledge to be learned--from the concrete media which the student will see, read, hear, and manipulate. In KBT-MM we call the knowledge units "**topics**" and the concrete media "**contents.**"
- Provide separate data objects to store the individual user (and tutor) events that happen during presentations, for example: "hint given", "correct answer" "poured water into the flask." KBT-MM calls these objects "**events**".
- Represent topics and their relationships in a semantic network, which we call a "topic network." Allow for a customizable vocabulary of **topic types and link types**.
- Provide methods for assigning **pedagogically relevant attributes**, such as difficulty, importance, and is-qualitative, to topics. The vocabulary of these attributes should be customizable.
- Provide a method for an **additional level of structure** inside of topic objects, as this will facilitate knowledge representation in most domain. In KBT-MM we do this by allowing topic to have any number of "**topic levels**" within them. The number and names of the topic levels should be customizable. Example topic levels are: introduction, examples, easy-level, practice, review.
- Provide methods for associating domain knowledge objects and media content objects to instructionally relevant categories or purposes, e.g. "explanation," "summary," "hint," "difficulty," so that teaching strategies can be designed using building blocks at the **pedagogical level** of abstraction. The KBT-MM design does this through a number of features, including customizable topic levels.
- Provide a method for defining the **objectives and context** of particular tutorial sessions. In KBT-MM we do this with **Lesson** objects, which, at the minimum, specify goal topics and the default teaching strategy for a particular assignment of the ITS to a group of students.
- KBT-MM specifies a "five **layered decision model**" that shows the logical and control relationships between the major objects in the curriculum model: Lessons, Topics, Topic Levels, Presentations, and Events.

4.2 Student Model

The KBT-MM specifies the following for the Student Model representational framework:

- The student model is an **overlay** on the five types of objects in the decision architecture. The types of values that can be assigned to each object (e.g. mastery, number of times given, certainty) is customizable.
- KBT-MM does not constrain the **inference methods** that are used to assign values to curriculum overlay objects, but specifies that the value of objects at each level is a function of the values at the next lower level.
- Define topic types for **buggy knowledge** or misconceptions to allow these to be recognized, diagnosed, and dealt with. The student model accumulates evidence for this "incorrect" knowledge using methods parallel to its regular topic diagnostic methods.

4.3 Teaching Model

The KBT-MM specifies the following for the Teaching Model representational framework:

- Allow the representation of instructional content and instructional **strategies separately**.
- Allow content to be generated and **sequenced dynamically**.
- Allow authors to create generic and **reusable teaching strategies** that can be used with different instructional content.
- Allow instructional strategies to be predicated upon **pedagogically relevant characteristics** of the content (e.g. whether a topic is a fact or a concept; whether a topic level is difficult or easy).
- Instruction decisions, if conceptualized as **rules**, have the objects, properties, and ontology vocabulary as described above available for their **consequents** (for example, running, showing, of "giving" a topic, lesson, presentation, hint, etc.). Similarly, the properties of the student model and of the curriculum model are available for rule **antecedents** (for example, predicating a tutorial action on mastery of a topic, the type of topic, or how many hints were given).

4.4 Authoring Tools

Finally, I summarize our recommendations regarding authoring tools that reify the KBT-MM representational framework.

- Provide **visual reification** for the concepts and structure of the underlying representational framework. This relieves working memory load and assists long term memory by providing memory cues. As a general rule, if some aspect of the representational framework is too complex to create clear visualization, then the framework may be too complex to expect non-programmers to be authors. Multiple views or multiple representations of the same structure are often needed to convey the full form and function of a feature.

- Provide tools that make it easy to **browse, search for, and reference all content** objects. Include tools that make it easy to navigate among the objects and tools.
- Anchor usability on **familiar authoring paradigms**, and facilitate evolution to more powerful paradigms. It is useful to have a look and feel similar to off-the shelf CAI authoring tools, and to provide features which allow a smooth transition from traditional authoring paradigms to authoring more powerful intelligent tutors. (For example, Eon's interaction editor and flowline editor have many surface similarities to off-the-shelf multimedia authoring tools.)
- Provide features powerful enough to author **generative content** and to create new presentations on the fly based on parameters of the current state. Similarly, provide productivity tools that capitalize on repetitive or **template-like content** (as in Eon's Presentation Content Editor). Other useful features include scripting languages, the ability to assign interface component attributes to methods or expressions, and the ability to have attribute values reference each other.
- Provide **WYSIWYG tools** that allow easy movement between authoring content and test-running the tutorial to facilitate rapid build-and-test iterations. If interface template screens are used, the tools need to help the author comprehend the distinction between the fixed items of a tutorial screen and the database-driven variable items.
- Facilitate an **opportunistic design** process. ITS authoring tools should allow for top down (starting with the abstract curriculum structure), bottom up (starting with specific screens and content), and opportunistic (switching between top down and bottom up as needed) design of ITSs.
- Allow for **interface extensibility**. Little is said in this paper about how authoring tools should support the design of ITS interfaces, but regardless of the method used, it is recommend that the system allow plug-ins or APIs such that interfaces of arbitrary complexity can be incorporated, and the author is not limited to building all learning environments from scratch using the primitive graphical objects available in the authoring tool. (An example is the "custom widgets" feature in Eon.)
- Provide tools to create the Ontology first, and subsequently the interface should **reify the Ontology**. Once the Ontology for a tutor is defined, the forms, menu selections, etc. of the authoring tool should reflect the vocabulary of the ontology. For example, if the Ontology defines the types of relationships allowed between topics then the tool that allows topics to be linked together should show exactly this list of relationships.
- Create **special purpose authoring** tools for increased usability and productivity. Generic authoring tools having Ontology objects can be used as "meta-authoring tools" to build special purpose authoring tools for particular classes of intelligent tutors.

5. ISSUES AND LESSONS LEARNED IN REPRESENTING PEDAGOGICAL KNOWLEDGE

This Section covers important issues that we have encountered when working with instructors trying to explicitly represent pedagogical knowledge for intelligent tutors. 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 definite structure and properties. The tension between the organic nature of knowledge and our need to modularize it leads to a number of unavoidable issues for ITS knowledge representation, which I discuss below, along with how we dealt with these issues in our research using the Eon system.

5.1 How Generic Can Teaching Strategies Be?

In Chapter 17 Section 5.3 I discuss limitations to what we can expect authors to be able to conceptualise and perform. In addition There are also limits to what we can reasonably expect in the level of sophistication of tutoring strategies. In our design for KBT-MM we are aiming for a particular level of generality in teaching strategies. To illustrated, consider the progression of hypothetical teaching strategies or rules from very specific to very general, where each item is intended to be an abstraction or reason encompassing the previous one:

1. If button #1 on screen #5 is pushed, then go to screen # 12.
2. If question-12 is answered wrong, then give explanation-5.
3. If the student gets a question wrong twice, then give a canned explanation.
4. If the student is very confused, then give an additional level of feedback.
5. Give students several opportunities to think about each situation so that they may learn from their mistakes, then scaffold feedback of increasing levels of specificity.
6. Learning happens through an active process of concept formation while trying to account for new information within in the context of previous knowledge.

This progression of hypothetical ITS tutoring rules goes from the trivial to the impossible. The first two items illustrate the low-level coupling of state testing and action found in (non-intelligent) CAI. The third item illustrates a type of tutorial reasoning that is typical of today's intelligent tutors. A tutor using this rule must keep a record of the student's behavior, but the reason why the rule is applicable is not explicit. The fourth item, though less common, is well within the state of the art for ITSs. A tutor using this rule must have abstract models of the student's mental state and the tutoring process. A diagnostic strategy must infer the level of "confusion" from student behavior (such as number of times asking for help), and the appropriate interpretation of "feedback" must be inferred based on the current state of the tutorial session. The fifth item states a pedagogical belief or strategy, and represents the principle behind the previous rule. It could be operationalized

in a limited way but is not precise enough to be part of a robust ITS (with today's technology). The final item is based on a theory---a psychological assumption. It represents the reason for the previous principle and the purpose for the rule above it. Representing and using knowledge at this level of abstraction is clearly out of the reach of current technology.

5.2 Topic Modularity and Interdependence

When 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 the emergent knowledge that comes from understanding topics together. Reigeluth (1983) and Lesgold (1988) refer to this as the "glue" in a curriculum (Lesgold also refers to this as "non-linearities" between topics).

The issue of curricular "glue" is not as much a natural property of the content, as an emergent phenomena that happens when instructional designers break up the content or domain knowledge into discrete chunks. 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, 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. In Eon we can define a topic type called Synthesizer. One possible strategy using synthesizers relates two topics the student has learned: after a topic is taught check whether a synthesizer connects it with another mastered topic, and if so, teach the synthesis material. Another possible strategy connects new information with existing information before the new information is presented: before a topic is taught, check whether a synthesizer connects it with an already known topic and, if so, present the synthesis material.

Lesson objects are another mechanism for dealing with curricular "glue". 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, moderate, and difficult) and allow prerequisite links such as Familiarity, Easy Prerequisite, and Moderate Prerequisite. Thus 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. This method simulates spiral teaching, in which the same topics are taught from successively more difficult perspectives (see (Murray & Woolf 1992) for more discussion of spiral teaching).

5.3 Knowledge Structure and Complexity

In order to create structured and purposeful instruction the instructional designer organizes her understanding of a domain into common "epistemic forms" (Collins & Ferguson 1993) such as hierarchies, tables, networks, scripts, and schemas, and as complex nested constructs using these basic forms. Halff (1988) notes that "the sequence of exercises and examples should reflect the structure of the (knowledge) being taught and should thereby help the student induce the target (knowledge)". ITS authoring must be supported with tools that allow clear visualization of the knowledge structures, and more esoteric or complex structures are difficult to comprehend and author. In KBT-MM we have devised a system that accommodates many but not all possibilities.

Having Topic Levels within the topics allows a significant increase in representational power over strictly network representations, without a large increase in complexity. For example, we have simulated Merrill's Performance Content Matrix (Merrill 1983) 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). If we tried to do this with topic networks 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. Levels of mastery (e.g. easy, difficult) can be encoded using topic levels, 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.

Additional analysis of a domain seems to always lead to additional complexity. For example, in our study of authoring the Statics Tutor we discovered numerous different perspectives on the material (Murray 1991 pg. 218), most of which we did not attempt to implement. For example, Newton's' Third Law can be taught by presenting questions which progress from existence (does a force exist here?), to direction (what direction is the force?), to relative magnitude, to quantitative questions. This material can also be taught by showing a progression of example situations using different surface features, e.g. with objects hanging, falling, rolling, colliding, etc. It can also be taught by dealing with first horizontal forces, then vertical, then both, then rotational forces. Each of these methods has some instructional merit. Some are within the capacity of the topic-network and topic-level to represent, and some would require a more complex scheme if the representational formalism is to clearly reflect the structure of the domain knowledge.

5.4 Non-Independence of Content and Strategies

Unfortunately, it would seem that little in ITS authoring is as simple "as advertised." One of the fundamental characteristics of ITSs, and what characterizes ITS authoring tools from CAI authoring tools, is the separation of content from teaching strategies--the separate representation of what to teach/present from how and when to teach/present it. However, it turns out that it is epistemologically impossible to completely separate them. In fact, the *meaning* of most of the conceptual objects used to define content depends inextricably on the strategy that is implemented to use those objects. For instance: what is the meaning of "prerequisite?" At a vague level the term specifies that one thing should be known before another is taught. But at a more precise level, for a particular ITS, the meaning is most strongly related to what a prerequisite relationship causes to happen. Are prerequisites always taught first? Are the parts of a subtopic also prerequisites of the topic? If you have to "know" all prerequisites before being taught a topic, do you have to know them completely, or only at an introductory level? The answers to these questions are found in the teaching strategies (and perhaps in the student model rules). One *might* be able to design the content knowledge base without making any assumptions about teaching strategies, but more likely, the decision about whether one topic should really be called a "prerequisite" of another will depend on knowing the answer to these types of questions. As another example, consider the decision of whether to classify a topic as a "concept". At a vague level we may have a meaning for "concept," but at a deeper level whether we classify a topic as a concept or something else is closely related to how the system will behave as a result of that categorization.

All this does not change our vision for effective authoring tools, but is offered as a note of caution to designers. Content can still be authored to be flexible and re-usable. But every ITS design group must engage in an ongoing process of grounding the meaning of the objects and terms that they use. An authoring tool, by its very nature or through accompanying documentation, can suggest meanings or ranges of interpretation for these terms, but this does not alleviate the necessity of meaning-grounding within each design team. It means that designing content ontologies and tutoring strategies may need to be an iterative process, and that when decisions are made about ITS conceptual building blocks, that the designer should note what this reveals about assumptions related to teaching strategies.

6. CONCLUSIONS

Authoring tools can have a variety of purposes and intended users, and their design must account for tradeoffs among four overall goals: usability, power, flexibility, and generality. This paper describes a reference model or generic framework (KBT-MM), plus a set of guidelines and lessons learned, with the goal of supporting the creation and use of ITS authoring tools that maximize among these four goals (especially for pedagogy-oriented tutors). KBT-MM is in a sense an attempt to answer the question: is there a common underlying framework that could be used to describe all or most ITSs or ITS authoring systems? In terms of the systems

described in this book it does so with only moderate success, because of the diversity of systems represented. However I believe that it does capture the essential or fundamental elements of the systems that fall within our definition of knowledge-based tutors. If compared to any particular knowledge based tutor authoring tool one might say that KBT-MM is fine as far as it goes, but leaves out all of the most interesting parts. This is precisely because what one find most interesting about a particular system are the parts that are particularly innovative or distinctive. The KBT-MM is offered as a reference model which could be used to compare systems and as a starting point for the design of new systems.

The issues involved in building an ITS can be subtle, and a trained knowledge engineer may always be needed on the ITS design team. But with appropriate representational formalisms and tools that visually reify the conceptual structures involved, learning how to be a good ITS knowledge engineer can be made accessible to many more people, not just to computer programmers and AI scientists. Also, once a trained person gets the primary structures set up, an instructional designer with much less training can continue to fill in the content.

Future plans for this line of work include working with researchers who work in the area of performance-oriented systems, especially model-tracing tutors, to extend or combine the principles outlined in this paper to that domain. I have also applied many of these principles to the design of an authoring tool for adaptive hypermedia, as described in (Murray 2002).

7. REFERENCES

Anderson, J. (1983). The Architecture of Cognition. Harvard Univ. Press: Cambridge, MA.

Ausubel, D.P. (1960).The use of advanced organizers in the learning and retentions of meaningful verbal material. J. of Educational Psychology 51, 267-272.

Betz, F. Suthers, D., Wheeler, T. (1997). Architecture Abstraction Hierarchy Reference Model. IEEE Learning Technology Standards Committee draft document.

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.

Collins, A. & Ferguson, W. (1993). Epistemic Forms and Epistemic Games: Structures and Strategies to Guide Inquiry. Educational Psychologist, 28(1), 25-42.Collins, A.M. & Loftus, E.F. (1975). A spreading activation theory of semantic processing. Psychological Review 82(6), 407-428.

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

Ginsburg, H. & Opper, S. (1979). *Piaget's Theory of Intellectual Development*. Prentice-Hall: Englewood Cliffs, NJ.

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.

Hoffman, R. (1987). "The Problem of Extracting the Knowledge ofExperts From the Perspective of Experimental Psychology." *AI Magazine*, pp. 53-67, Summer 1987.

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.

Mizoguchi, R., Sinitsa, K., Ikeda, M. (1996). Knowledge Engineering of Educational Systems for Authoring System Design. In *Proceedings. of EuroAIED-96,* pp. 593-600. Lisbon.

Murray, T. (1991). *Facilitating Teacher Participation in Intelligent Computer Tutor Design: Tools and Design Methods.* Ed.D. Dissertation, Univ. of Massachusetts, Computer Science Tech. Report 91-95.

Murray, T. (2002). MetaLinks: Authoring and Affordances for Conceptual and Narrative Flow in Adaptive Hyperbooks. International Journal of Artificial Intelligence in Education, Vol. 13.

Murray, T. & Woolf, B. (1992). Tools for Teacher Participation in ITS Design. In Frasson, Gauthier, & McCalla (Eds.) Intelligent Tutoring Systems, Second Int. Conf. , Springer Verlag, New York, pp. 593-600.

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

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

Schoening, J. & Wheeler, T. (1997). Standards--The key to educational reform. IEEE Computer, March 1997, pp. 116-117.

Van Marcke, K. (1992). Instructional Expertise. In Frasson, C., Gauthier, G., & McCalla, G.I. (Eds.) *Procs. of Intelligent Tutoring Systems* '92. Springer Verlag, Berlin.

VanLehn, K. (1987). "Learning One Subprocedure per Lesson," *Artificial Intelligence*, Vol. 31.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems.* Los Altos, CA: Morgan Kaufmann.

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.

Winne P.H., 1991. Project DOCENT: Design for a Teacher's Consultant. In Goodyear (Ed.), *Teaching Knowledge and Intelligent Tutoring.* Norwood, NJ: Ablex.

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.