Expanding the Knowledge Acquisition Bottleneck for Intelligent Tutoring Systems
(Preface to the IJAIED Special Issue on Authoring Systems for Intelligent Tutoring Systems)

Tom Murray
Computer Science Department
University of Massachusetts, Amherst, MA
tmurray@cs.umass.edu

The first conference workshop on authoring tools for intelligent tutoring systems (ITSs) was held at the AI and Education conference in 1995. Since then a number of workshops and symposia on ITS authoring, cost-effective production, and reusability have been offered and well attended (see "Related workshops and symposia" at the end of this article). The field has come a long way in the last decade, and the time is right to take stock of where we are. Submissions to this special issue were numerous enough to require that it be organized into two parts. We expect to have Part II of this special issue published within a year. In Part II this introductory paper will be expanded into a more complete overview of the state of the art in ITS authoring tools.

ITSs go beyond traditional computer based instruction (CBI) in trying to build models of subject matter expertise, instructional expertise, and / or diagnostic expertise. This is done so that the content and feedback can be "intelligently" composed and sequenced, adapting to the student and the subject matter. Part I of this special issue includes five diverse attempts at simplifying ITS construction. Despite the diversity in approaches, we can see substantial agreement concerning the problems and challenges we face in our attempts to make advanced technology learning systems more cost effective. In various ways the authors in this special issue begin by making the same arguments: intelligent tutoring systems can be powerful and effective learning environments; they are very expensive and difficult to build, and therefore authoring tools are needed to make them cost effective and to allow a wider range of people to build ITSs.

Do we know enough about ITSs to build authoring tools? Some (Youngblut, 1995) say that there are too few ITSs to make informed design decisions about ITS shells and authoring tools. There is certainly a grain of truth to this, but it is also true that so few ITSs exist for evaluation and generalization because they are so difficult and expensive to build. So we must continue to explore the various approaches to ITS authoring, and develop metrics to evaluate their success. Appropriate metrics for success include: the diversity (in terms of subject area and / or teaching style) of ITSs that a tool can be used to build, the cost effectiveness of using the tools to build ITSs, the depth or sophistication of the ITSs the tools can be used to build, and the ease with which the tools can be used. It is still to early to evaluate the overall effectiveness of ITS authoring tools, since most current data related to these metrics are either limited or anecdotal.
To bring highly interactive, flexible instructional systems into the mainstream there needs to be a paradigm shift in the way traditional computer-based instruction (CBI) authors perceive courseware. A major part of this is a shift from a story boarding paradigm for authoring to a "knowledge based" (or even "data based") paradigm. Fundamental to this shift is the separation of content, which resides in the knowledge base, from the instructional methods, which make decisions about how to select, sequence, and compose the content that the student sees. But in having this goal instructional designers are in new and difficult situation. Authors are being asked to represent their knowledge explicitly and modularly, as opposed to simply authoring what the student will see and enumerating the possible paths that the student can take. We are asking authors to create explicit representations of the content and the instructional strategy. The explicit representation of one's "knowledge" has long been a major bottleneck to AI work, and also to those who are trying to create computational models of instructional design theories. It can be an arduous and ambiguous task, but significant progress has been made, as is evidenced in the articles in this journal issue.

The knowledge acquisition bottleneck problem is particularly acute in intelligent tutors which incorporate rule-based representations of subject matter expertise—i.e. those which require the author to build an expert system, a task that usually involves substantial programming and/or knowledge engineering skills. Model tracing tutors are in this category. They incorporate production rules to represent problem solving behaviors. Blessing's paper (in this issue) describes the use of "programming by example" to decrease the skill threshold for authoring domain knowledge in model tracing tutors. In his Demonstr8 system the author simply manipulates objects on the same interface that the student will use, to demonstrate how to solve a specific problem. The system then uses analogy-based AI techniques to generalize from this specific example to the general solution. For example, the author can demonstrate how "carrying" is done for a specific addition problem, and the authoring system will generate a rule for how carrying is done in the general case. This general rule becomes part of the tutor's expert system for arithmetic skills.

Successful use of programming by example requires that the system have a rather deep and complete understanding of the theory underlying the generalized task that is to be learned. Doing this in the domain of arithmetic, the learning domain used in Blessing's research, is easier than in most domains. As Blessing notes, more research is needed to learn whether the powerful "programming by demonstration" method has wide applicability.

The ability to simulate processes and phenomena that are impractical for students to experience directly is one of the most persuasive reasons to use computers for instruction. Intelligent tutors that incorporate simulations must include the rules or processes that drive the simulation and the rules or steps which comprise expert problem solving for the subject area. The RIDES authoring system (Munro et al. in this issue) provides tools for easily building graphical representations of a physical device, and for authoring constraints and cause-effect events which define the device's behavior. Students can interact with the simulation and observe the effects of their actions. RIDES is most often used to build tutors that teach students how to operate mechanical or electrical devices. RIDES exploits the structural information contained in the device definition (the names and descriptions of components, and their relationship to each other) to simplify the authoring of the domain expertise. Once the device simulation has been authored, the system already knows the names of parts and how they affect each other. Thus the authoring system can automatically generate instruction, including feedback and evaluation, regarding component identification and interrelation. To generate instruction about operating a device, RIDES uses
an example-based technique similar to that employed in Demonstr8 (see above). The author simply uses the same interface that the student would see to manipulate device components, for example flipping switches to turn on components and adjusting controls according to the value observed on a meter. The authoring tools record each procedural step as the author does it, and can use this information to generate a tutorial about the recorded procedure. The student is given feedback on the correctness of his steps, and can ask for hints in performing the next step.

This type of example-based authoring, though powerful, is less sophisticated than that used in Demonstr8 because the procedure demonstrated by the author is the procedure that the student will be learning, whereas in Demonstr8 the system must generalize the authorâs procedure to use it for tutoring. Along similar lines, the instructional methods used in RIDES are fairly straightforward and uncomplicated in comparison to those used in the other authoring tools we describe here. The flip side of these comparisons is that, though RIDES has less "intelligence" or knowledge depth in its domain expertise and instructional strategies than most of the other systems, it is also the most robust and, by far, the most extensively used. RIDES (and its predecessors) has been used to build dozens of tutors in a number of domains, and it has been used as the software foundation for a number of more advanced systems which add on to its capabilities. One of these is the DIAG system (Towne in this issue).

RIDES focuses on instructing about device operation and functionality, whereas DIAG adds capabilities to RIDES for teaching about the diagnosis of equipment failures. DIAG simulates various equipment faults and guides the student through the diagnosis and repair of these faults. Equipment diagnosis is a much more complex task than equipment operation, and assisting an expert in articulating diagnostic expertise is much more difficult. In complex equipment the number of possible fault states, including multiple faults, is extremely high, and the problem of inferring the interpretations of every possible fault indication (e.g. a meterâs value or the on/off state of a light) is intractable. DIAG applies fuzzy set theory to the expertâs qualitative judgments about the relationship between diagnostic signals and equipment faults to construct a fault table that can be used to diagnose system faults based on symptomatic indicators. Thus the problem of representing the factors and relationships involved in every possible failure scenario, of which there could be millions, is reduced to answering a few hundred questions such as "How often will a failure in Unit 3 produce a Low reading on Display 30? Choose one: very often, often, unlikely, rarely." As in the RIDES system, information that is entered to describes the device and its behavior is used to automatically generate instruction about the device. Faults are randomly generated, students work at diagnosis, and the system generates hints and feedback, all automatically.

Intelligent tutoring systems have been described as having three kinds of expertise: knowledge of how to solve problems in the domain, knowledge of how to teach, and knowledge about how to diagnose the studentâs understanding. Enabling these areas of "intelligence" to a significant depth in any given domain is difficult, and with the current state of the art most ITSs excel at one or two of these areas and contain simplified models of the other area(s). Enabling these types of intelligence using authoring tools is even more difficult, since the knowledge must be represented in a more general way if it is to apply to the whole class of ITSs that the authoring tool is used to build. Parallel to the situation with ITSs, the current state of the art for ITS authoring tools is that they tend to excel at supporting users in authoring sophisticated capabilities in limited areas. The other areas are simplified by incorporating simplistic models or by automating a part of the system so that authors donât have to worry about it. RIDES and DIAG excel in the area of subject matter representation. The REDEEM system (Ainsworth & Major in this issue) focuses on the representation of instructional expertise.
The REDEEM system is perhaps the most "usable" of the ITS authoring tools described in the set of papers in this issue. While the other tools aim for usability, they still require a fair degree of either knowledge acquisition skill or domain expertise to use. REDEEM comes the closest to a system that one could imagine practicing school teachers using to build intelligent tutors. What has been "traded" to result in higher usability? First, the system only deals with sequencing content and learner activities, it does not deal with generating them. The content consists of pre-made interactive multimedia pages authored in Tool Book (a multimedia production tool), which are imported into REDEEM. Questions, answers, and hints (all text-based) related to these pages are authored in REDEEM. Second, the user's authoring task is limited to categorization and selection tasks; there is no need to design knowledge structures, procedures, or rules. Buttons and sliders are used to categorize tutorial "pages" and interactions according to difficulty, generality, familiarity, among others. Additional declarative information is entered such as which pages are prerequisites of a given page.

REDEEM contains a "black box" expert system of rules comprising a generic instructional strategy. Ainsworth and Major justify the use of one preset, flexible, general instructional strategy by noting that the parameters affecting instruction are generally agreed upon, but that no theory exists which can automatically determine the values of those parameters for a given instructional setting. Thus the authoring process is reduced to specifying these parameters. The default sequencing strategy has rules for how to use the parameters of the content pages to sequence the pages. The student's mastery of the questions associated with the content pages is also used to influence the sequencing. "Meta-strategies" can be authored to modify the default sequencing. This is done with easy to use sliders and buttons. For example, the author can define a sequencing strategy which prefers general to specific instructional pages, or which interleaves difficult and easy problems. The underlying expert system is general enough to encompass all of the parametric possibilities. One might wonder whether the benefits of "intelligent" tutoring can be realized from an authoring system that is so easy to use. REDEEM produces tutors with a high degree of flexibility in the content sequencing. The tutors adjust to the student's progress and the pedagogical characteristics of each page of content, a degree of adaptability far beyond that achieved in traditional multimedia instructional systems. It should be noted however, that even with REDEEM's level of usability, it was non-trivial for the average classroom teacher, particularly at the level of creating meta-strategies.

The instructional strategies embedded in REDEEM were derived from published psychological studies and interviews with teachers. It is generally agreed that since computer based learning is so novel compared with standard methods of instruction, we need additional quantitative evaluations of computer- based learning to create more effective ITS teaching strategies. ITS authoring tools can make it much easier to carry out the necessary studies, since they can represent alternative instructional strategies, and because learner data can be automatically gathered and analyzed.

Instructional design theory is yet another source on which to base ITS teaching strategies. For many years developers of intelligent tutoring systems seemed to ignore findings and principles from the field of instructional design (ID) theory (Merrill 1983, Gagne 1985). This may have been due to ID theory's focus on learning basic types of knowledge such as facts, procedures, concepts, and principles, whereas those designing intelligent tutors and learning environments were more interested in higher order thinking skills such as problem solving and metacognitive skills. Also, the most successful applications of ID
theory are based on a reductionist method of decomposing knowledge and behavior into its smallest units, and teaching those units independently using simplified tasks. ITS developers, for the most part, wanted to work with more complex and complete behaviors, a task that seemed ripe for the application of AI techniques. It was those working in the area of ITS authoring tools that brought ID theory into the ITS fold. This was because, though the most notable early successes in intelligent tutoring came from incorporating expert systems that modeled expert performance and student behavior, creating authoring systems for expert systems was intractable in any general way. Authoring tools must capitalize on aspects of intelligent tutoring that can use consistent representational schemes across many domains. It is easier to develop a generic formalism for describing instructional methods and for prerequisite-based curriculum than for arbitrary subject matter tasks. So most authoring tool research focussed on the representation of instructional strategies and pedagogical/curriculum knowledge. Instructional design theory has much to offer in this area. The key contributions stem from what has been called the “Gagne hypothesis:” that there are different types of knowledge (or “learned capabilities”) and that there are different methods appropriate for promoting the learning of each type. Various theories have been proposed which incorporate the three concerns of the Gagne hypothesis: 1) distinguishing types of knowledge and skill (learned information that presumably exists in oneâs head); 2) relating this knowledge (which canât be directly measured) to concrete, measurable behavioral objectives; and 3) prescribing the types of instructional conditions that are most likely to promote the learning of each type of knowledge or skill. Instructional design theories have more elaborate and, from an instructional point of view, practical taxonomies and prescriptions than those traditionally seen in either AI or psychology.

The IRIS Shell (Arruarte et al. in this issue) integrates elements of instructional design theories from several sources, including the theories of Merrill, Gagne, and Bloom (Merrill 1983, Gagne 1985, Bloom 1956). It incorporates a least-commitment incremental planning architecture to plan the specific instructional events to reach the stated instructional objectives of a tutorial. Opportunistic re-planning adjusts the plan to unexpected student states. The end goal of the system is to facilitate teachers in developing intelligent tutors, but it is too early to tell how much training will be needed to allow teachers to comprehend the sophisticated representational formalism. So far the IRIS tools facilitate easy creation of instructional objects and data entry for the object attributes, but do not support visualization of the overall knowledge structures.

IRIS is one of a class of ITS authoring tools which use what I will call "pedagogy oriented domain modeling." They are oriented around a semantic network of topics or instructional objectives, and include pedagogically relevant topic properties, such as knowledge type and difficulty. This information, along with inter-topic relationships such as "prerequisite" and "part" are used to build a representation of the subject matter from a pedagogical view point. Other systems, including LEAP, Demonstr8, RIDES, and DIAG, use "performance oriented domain modeling." These systems are oriented around the representation of domain procedures or rules. Because domain expertise is not very generic, these systems are less general (more specialized to particular domain types) and conversely more powerful in their ability to support authoring for a specific domain type. Pedagogy-oriented systems tend to have different representational schemes and different teaching strategies for several different knowledge types, including facts, concepts and simple procedures. Performance-oriented systems tend to incorporate more complex procedures, but have simplistic instructional strategies. Pedagogy-oriented systems have simple overlay student models, while performance-oriented systems can have more fine-grained student models. Eventually, the benefits of both types of systems will be combined, but the challenges of developing good authoring tools has constrained developers to focus their efforts, as mentioned above. XAIDA (Redfield 1996), for example, straddles the two categories. It uses simple performance-type knowledge but spans a number of knowledge types.
How generic should an authoring tool be? Bell (1998) believes that teachers need a high degree of scaffolding to be able to author instructionally effective ITSs, and that such detailed support is possible only when a specific task model and instructional paradigm are provided. Thus he argues that ITS authoring tools should be domain specific (or "special purpose"). This is a sound argument, but it must be balanced by the concern that a plethora of different authoring tools may not be an attractive solution to institutions wanting to build a variety of intelligent tutors. Though one-size-fits-all authoring tools are likely to be too open ended to be useful to all but the most trained authors, more research is needed to discover whether a manageable set of special purpose tools will satisfy the demand.

The major challenge to those creating ITS authoring tools is not simply to automate or streamline what instructional designers already do, but to provide tools which facilitate a reconceptualization of both content and on-line instruction, so that they can build more powerful learning environments. The papers in this special issue demonstrate how authoring tools can assist knowledge acquisition in several ways.

- First, authoring tools can make knowledge/data entry more efficient, especially as compared to creating learning environments via straight programming, by structuring the knowledge/data entry task, providing GUI fill-in forms, and by facilitating knowledge representation that is reusable and modular.
- Second, authoring systems can help the author articulate implicit knowledge (and then debug the knowledge which has been made explicit) by using prompts, visualization aids, consistency checking tools, and try-it-out capabilities (e.g. the designer can test run a tutorial to see if it behaves the way she wants it to).
- Third, the authoring system can include knowledge that the expert may not have, as in the case of authoring systems that contain pre-made instructional strategies or constrain the knowledge base to adhere to good instructional principles.
- Finally, authoring tools can actually create new knowledge beyond what the expert, or even those building the authoring tool itself, could reasonably be expected to know or deduce.

Ten years ago ITS authoring was a vague concept. Today there are a number of tools used to build tutors in a variety of domains, and these systems help us articulate the tradeoffs, opportunities, and limitations in designing these authoring environments. We are learning what types of limits need to be placed on generality and power in order to author an effective ITS. We have evidence that ITSs can be authored with no more effort than that given to authoring traditional CAI systems.

¹Compare the standard declarative/procedural or short vs. long term memory categories to Merrillâs component display theory, which categorizes knowledge into about 16 categories along two dimensions: content type and performance level

¹ Other authoring tools which use pedagogy oriented domain modeling are: REDEEM (in this issue), Eon (Murray 1998), CREAM (Nkambou et al. 1996), IDE (Russell et al. 1988) and GTE (Van Marcke 1992). A complete description of the Eon ITS authoring system developed by myself and others at UMass can be
found in Murray 1998. Since that paper pretty much tells the whole story to date, I did not submit a paper on the Eon tools to this special issue.

Acknowledgements

Thanks to those who reviewed articles for this issue, especially Charles Bloom who helped get it off the ground. Special thanks to my co-guest editor Stephen Blessing, who was instrumental in ushering articles through second and third revisions.

References

[Other than the papers appearing in this special issue.]


Redfield, C.L., "Demonstration of the eXperimental Advanced Instructional Design Advisor," Third International Conference on Intelligent Tutoring Systems, Montreal, Quebec, Canada, June 12-14, 1996


Related workshops and symposia


